Crease Surfaces: From Theory to Extraction and Application to Diffusion Tensor MRI

Thomas Schultz, Holger Theisel, and Hans-Peter Seidel

Abstract—Crease surfaces are two-dimensional manifolds along which a scalar field assumes a local maximum (ridge) or a local minimum (valley) in a constrained space. Unlike isosurfaces, they are able to capture extremal structures in the data. Creases have a long tradition in image processing and computer vision, and have recently become a popular tool for visualization. When extracting crease surfaces, degeneracies of the Hessian (i.e., lines along which two eigenvalues are equal) have so far been ignored. We show that these loci, however, have two important consequences for the topology of crease surfaces: First, creases are bounded not only by a side constraint on eigenvalue sign, but also by Hessian degeneracies. Second, crease surfaces are not, in general, orientable. We describe an efficient algorithm for the extraction of crease surfaces which takes these insights into account and demonstrate that it produces more accurate results than previous approaches. Finally, we show that diffusion tensor magnetic resonance imaging (DT-MRI) stream surfaces, which were previously used for the analysis of planar regions in diffusion tensor MRI data, are mathematically ill-defined. As an example application of our method, creases in a measure of planarity are presented as a viable substitute.

Index Terms—Height crease, ridge surface, valley surface, tensor topology, DT-MRI stream surface.

1 Introduction

Local extrema are characteristic structures of scalar fields, and are relevant in a wide variety of applications. However, typical data sets assume unconstrained local extrema only in isolated points. For cases in which higher dimensional extremal features are more appropriate, there exist crease definitions, which generalize local extrema: Ridges generalize local maxima, while valleys generalize local minima.

The crease definition considered in our work has been introduced to visual computing by Haralick [13], who suggested the use of creases to capture highlight and shadow lines in natural images. Haralick defines creases as lines in a 2D image along which the first directional derivative, taken in a direction which extremizes the second directional derivative, changes sign. This is known as the "height crease definition," since it is motivated by treating the intensity profile of an image as a height field and is closely related to the notion of ridges and valleys in surface topography.

Different crease definitions have been proposed and there has been some dispute over which is the "correct" one [19]. After a theoretical analysis and visual comparison of results, Eberly et al. [9] conclude that height creases are most suitable for digital image analysis. Their reformulation of Haralick's definition, which generalizes it to arbitrary d-dimensional creases in n-dimensional images, will be presented in Section 2.1.

• T. Schultz and H.-P. Seidel are with MPI Informatik, Department 4—Computer Graphics, Campus E1 4, 66123 Saarbruecken, Germany. E-mail: {schultz, hpseidel}@mpi-inf.mpg.de.
• H. Theisel is with the University of Magdeburg, Otto-von-Guericke-Universitaet Magdeburg, Fakultaet fuer Informatik, AG Visual Computing, Universitaetsplatz 2, 39106 Magdeburg, Germany. E-mail: theisel@isg.cs.uni-magdeburg.de.

Manuscript received 17 Dec. 2008; revised 30 Mar. 2009; accepted 10 Apr. 2009; published online 16 Apr. 2009.
Recommended for acceptance by M. Chen.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2008-12-0207.
Digital Object Identifier no. 10.1109/TVCG.2009.46.

In the present work, we concentrate on 2D creases in 3D space. This was motivated by the fact that crease surfaces have recently received increasing attention as tools for visualization and computer graphics [16], [32], [37], [27], but have so far not been treated thoroughly from the methodological side. In particular, Hessian degeneracies, which have been neglected in previous algorithms for crease surface extraction, have important consequences for the topology of creases. Based on this insight, we propose an efficient algorithm which produces more accurate representations of crease surfaces.

The remainder of this paper is organized as follows: After providing a formal definition of height creases and discussing related work (Section 2), we describe our theoretical results on crease surface topology (Section 3). We present our novel algorithm for crease surface extraction (Section 4) and demonstrate a clear improvement over the state of the art (Section 5). Finally, we show that diffusion tensor magnetic resonance imaging (DT-MRI) stream surfaces [44] are ill-defined and propose crease surfaces as a replacement to analyze planar regions in diffusion tensor MRI visualization (Section 6), before we conclude the paper (Section 7).

2 Related Work

2.1 Definition of Height Creases

The formal definition of height creases given in this section follows the idea of Eberly et al. [9], but adopts the simplified notation used in [17]. Assume a C^2 scalar field $f : \mathbb{R}^n \rightarrow \mathbb{R}$. Let $\mathbf{g} = \nabla f$ be its gradient and $\mathbf{H}$ be its Hessian with eigenvectors $\mathbf{e}_i$ and eigenvalues λ_i, $i \in \{1, 2, \ldots, n\}$, sorted such that $\lambda_1 \geq \cdots \geq \lambda_n$. Then, a d-dimensional height ridge is given by the conditions

$$\forall_{d<i\leq n} \quad \mathbf{g} \cdot \mathbf{e}_i = 0 \quad \wedge \quad \lambda_i < 0. \tag{1}$$

Intuitively, this means that f attains a local maximum in the $n-d$ directions of strongest convexity. For ridge surfaces in $\mathbb{R}^3$, this definition simplifies to

$$\mathbf{g}\cdot\mathbf{e}_3 = 0 \quad \wedge \quad \lambda_3 < 0. \tag{2}$$

The valleys of f are exactly the ridges of $-f$, so they need not be discussed separately.

2.2 Crease Extraction

Crease lines have been studied extensively by Pizer et al. in the context of medial core extraction [29], which generalizes the Blum medial axis analysis of binary objects [4] to finding the core of objects in gray-scale images. Pizer et al. employ a medial function, which yields high values at the center of an object, and they extract its core as ridges in medial function values. Lindeberg [20], whose formalism differs slightly from the presentation above, and Damon [6] carefully investigated the behavior of height creases in Gaussian scale space.

Crease lines have also been used for finding vortex cores in vector field visualization, for example, by Miura and Kida [23] and by Sahner et al. [33], and for the extraction of characteristic lines in symmetric tensor fields by Tricoche et al. [40]. In this context, the parallel vector approach by Peikert and Roth [26] and the feature flow fields by Theisel and Seidel [39] provide popular algorithms for crease line extraction.

For crease surfaces, Furst et al. have proposed the "marching cores" algorithm [12], which addresses the problem of finding 2D creases in a 4D $(3\text{D}+\text{scale})$ space. In their "marching ridges" method [11], Furst and Pizer even extend this to the extraction of one and two-dimensional creases from spaces of arbitrary dimension. To deal with such high complexity, they make simplifying assumptions, such as that the boundary of each face is only intersected twice by a ridge. Intersections are found as changes of sign in $\mathbf{g}\cdot\mathbf{e}_i$, after imposing a local orientation on $\mathbf{e}_i$ via a principal component analysis (PCA).

Crease surfaces should not be confused with *surface creases,* which are lines of extremal curvature on general surfaces [3]. Methods for surface crease detection take a surface as input, and produce salient lines on it as output. Therefore, they are not applicable to our problem of finding surfaces from sampled volume data.

So far, the application of ridge surfaces in visualization has been restricted to single-scale analysis, so it has been sufficient to find crease surfaces in 3D space. To find skeletal structures in data from diffusion tensor MRI, Kindlmann et al. [16] have extracted ridge surfaces as isosurfaces of $\mathbf{g}\cdot\mathbf{e}_i = 0$, using the marching cubes algorithm [21] after imposing a per-cell orientation on $\mathbf{e}_i$ by tracking eigenvectors along subsampled cell edges.

Sadlo and Peikert [32] have used marching cubes on an adaptive grid to extract ridge surfaces which separate regions of different flow behavior in unsteady vector fields, using the original rule from [11] to orient eigenvectors. Another recent work on vector field visualization by Sahner et al. [34] has used crease surfaces, but employed a different, watershed-based definition. In computer graphics, Süßmuth and Greiner [37] used marching cubes to reconstruct surface meshes from noisy point clouds by extracting ridges of point cloud density. They use the height

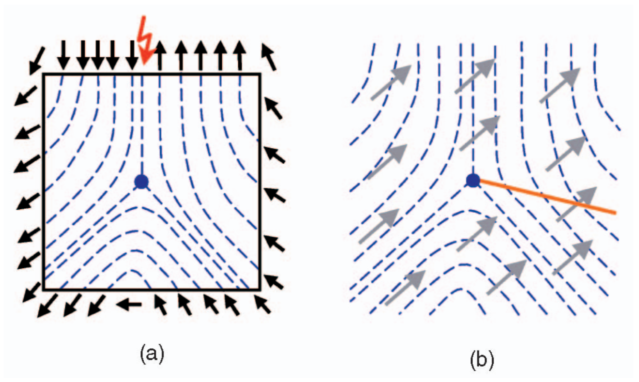


Fig. 1. (a) Orienting eigenvectors can lead to a contradiction. (b) Lines along which $\mathbf{g}\perp\mathbf{e}_i$ end in degeneracies.

crease definition, but do not provide details on their way of orienting eigenvectors.

In the following section, we will discuss differences between the topology of isosurfaces and crease surfaces, which show that the marching cubes case table is inappropriate for crease extraction. This will lead us to a novel algorithm which is specialized for the extraction of 2D creases from 3D fields.

3 On the Topology of Crease Surfaces

3.1 Terminology

This paper discusses "generic" (or "structurally stable") properties of crease surfaces. A formal definition of genericity is given in [6]. In practice, generic properties are the ones which we can expect to meet under general conditions and which remain stable under small perturbations.

3.2 Degenerate Lines as Boundaries

Unit eigenvectors are only defined up to their sign. However, previous algorithms for the extraction of crease surfaces rely on a locally consistent sign of the involved eigenvector, so prior work has suggested different ways to impose a local orientation on it [25], [11], [16].

Unfortunately, orienting the eigenvector along the boundary of a cell face is, in fact, impossible when the Hessian has a degeneracy in the interior of the face, i.e., a point at which two eigenvalues are equal: Delmarcelle and Hesselink [7] have shown that the Poincaré indices of the three generic types of degenerate points in tensor fields are half-integers. This means that the eigenvector turns $\pm\frac{1}{2}$ times when traveling along a closed line around a degenerate point in counterclockwise direction. This is illustrated by Fig. 1a, which has a degenerate point (blue) at the center and indicates eigenvector directions in the plane around it by dashed blue lines. As indicated by the arrows, trying to impose a consistent sign along the cell boundary around the degeneracy (black) leads to a contradiction. This case is not rare in practice: Degenerate loci in symmetric 3D tensor fields generally form stable lines [6], [46].

In the context of crease surfaces, degenerate (also called semiumbilic) locations have traditionally been discussed as a source of numerical difficulty when imposing a local orientation on eigenvectors (e.g., [25]). Only recently, it has been pointed out that their presence implies that eigenvectors are not orientable in principle [27]. To the best of our knowledge, it has so far not been discussed that degenerate lines actually constitute one type of crease surface boundaries: Besides the obvious type of boundaries, which are caused by the side constraint on the eigenvalue ($\lambda_3 < 0$ or $\lambda_1 > 0$, respectively), ridge surfaces are bounded by type L degenerate lines ($\lambda_2 = \lambda_3$), and valley surfaces are bounded by type P lines ($\lambda_1 = \lambda_2$).

For crease lines in 2D, this insight follows directly from the Poincaré index of the degenerate point: Along the boundary of an ϵ-environment around it, the eigenvector turns $\pm\frac{1}{2}$ times, while changes in the gradient can be neglected for sufficiently small ϵ. Thus, both vectors are orthogonal (i.e., the crease intersects the boundary) exactly once—the crease ends inside of it. Fig. 1b illustrates this: Along the crease (orange), gradient vectors (gray) are orthogonal to the eigenvectors (blue). Behind the degenerate point, both vectors are parallel, so the crease ends.

This argument carries over to crease surfaces in 3D by projecting the gradient vector to the eigenplane of the repeated eigenvalue (the part outside the eigenplane is orthogonal to the relevant eigenvector anyway) and observing that generic 3D degenerate points behave just like 2D degeneracies within that plane (as shown in [47]). This also clarifies that, in general, crease surfaces do not branch, since this would require degeneracies with index $\pm\frac{3}{2} + n, n \in \mathbb{N}_0$, which are not structurally stable in 3D.

Extracting the skeleton of a bifurcating structure as a crease surface typically does *not* result in a nonmanifold sheet. Rather, one part of the surface ends shortly before meeting the other one. In our experience, it is exactly this case in which degenerate lines occur as crease surface boundaries most frequently in practice.

Note that corresponding results for crease lines in Gaussian scale space have been obtained in the context of medial cores by Damon [6], in a work which has not found adequate attention in the visualization community: Among other things, Damon proved that degenerate loci of symmetric 3×3 matrices form stable lines in 3D (cf. Propositions 8.1 and 9.1 in his work), which was later rediscovered by Zheng and Pang [46].

3.3 Nonorientability of Creases

The fact that $\mathbf{g} \cdot \mathbf{e}_i = 0$ defines a surface with boundary even before considering any further constraints introduces the possibility that crease surfaces may not be orientable, i.e., it may not, in general, be possible to assign a normal vector field with consistent sign to a crease surface. This problem has been encountered by previous authors [17], [32], but so far, it has not been discussed whether it is a true property of creases or merely a numerical artifact of existing extraction techniques. Also, examples of nonorientable creases have not been published so far.

Peikert and Sadlo [27] propose to extract crease surfaces as subsets of the zero isosurface of a scalar measure

$$d = \det(\mathbf{g}|\mathbf{Hg}|\mathbf{HHg}), \tag{3}$$

which first appears in a work by Süßmuth and Greiner [37]. The fact that creases can be expressed as a filtered isosurface suggests that they are orientable. However, the scalar field d changes sign in an ϵ-band around the crease not only in normal direction, but also along the surface. More precisely, $d = 0$ not only when $\mathbf{g}$ is orthogonal to the selected eigenvector ($\mathbf{e}_3$ for ridges, $\mathbf{e}_1$ for valleys), but to an arbitrary eigenvector. Along parallel vector lines $\mathbf{g} \parallel \mathbf{e}_i, \mathbf{g}$ is orthogonal to *both* remaining eigenvectors $\mathbf{e}_j$ ($j \neq i$), so in these places, the zero isosurface of d self-intersects—the sign of d changes along the crease.

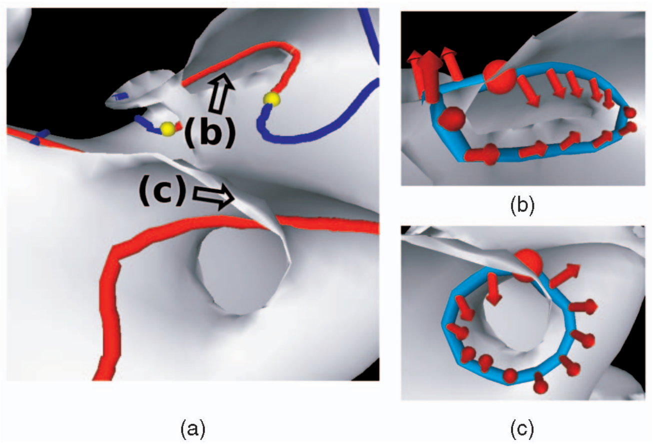


Fig. 2. (a) Cutting this nonorientable ridge surface along the red and blue lines yields orientable pieces. (b) and (c) Two nonorientable paths are shown in detail.

Fig. 2 presents a ridge surface from a real-world MRI data set. In two places, marked by arrows, it shows surface pieces which are homeomorphic to the Möbius strip. This establishes the fact that nonorientability is, in fact, a true property of creases. To give a better visual impression of the nonorientability, Figs. 2b and 2c show closed paths (light blue) along which the normal (red) cannot be oriented—at the points marked by a red ball, a contradiction occurs. In Fig. 2a, the parallel vector lines $\mathbf{g} \parallel \mathbf{e}_1$ are shown in red, $\mathbf{g} \parallel \mathbf{e}_2$ in blue, and type P degeneracies are shown as yellow spheres. This illustrates that cutting the crease along these lines would result in pieces that could be oriented using the scalar d from (3).

4 Extraction of Crease Surfaces

4.1 Basic Idea

From the observations made in the previous section, it follows that marching cubes are not suitable for the extraction of crease surfaces. Since isosurfaces are closed, marching cubes only consider cases in which the boundary of each cell face is intersected an even number of times. If there really is an odd number of intersections (because the crease ends inside the cell), applying the marching cubes case table will either add spurious triangles or create a hole. In existing algorithms, as used in [16] and [32], either of these options happens at random.

Peikert and Sadlo [27] have proposed to solve this problem by using marching cubes to extract a superset of creases and filtering out irrelevant parts afterward. This is theoretically appealing, but unfortunately, it is infeasible in practice, since the marching cubes algorithm cannot handle the self-intersections which occur in the zero isosurface of their scalar field d. In fact, it follows from the nonorientability of crease surfaces that it is generally not possible to close them in $\mathbb{R}^3$ without introducing self-intersections.

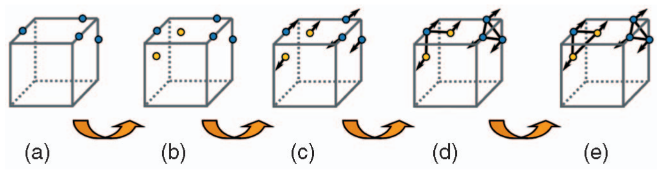


Fig. 3. Our algorithm creates the mesh per cell, by (a) and (b) finding intersection points, (c) estimating normals, and (d) and (e) connecting the points based on them.

The algorithm we propose instead is also based on cell marching, but does not rely on the marching cubes case table to determine topology. Fig. 3 gives an overview of our pipeline: We first extract individual intersections of cell edges with the crease surface (Fig. 3a) and of cell faces with the degenerate lines which bound the crease (Fig. 3b). We then estimate surface normals at these points (Fig. 3c) and use them to select the most likely topology, both on the faces (Fig. 3d) and within the cell (Fig. 3e). Taken together, this leads to closed polygons over the boundary of each cell, which can be triangulated to form the final mesh.

Without showing results, Eberly [8] has proposed a similar strategy to extract 2D creases from 3D fields. However, he assumes that the intersections of crease surfaces with cell faces can be described as the zero contour of a bilinear function. Like marching cubes, this does not allow creases to terminate within a cell.

To keep the notation simple, we will restrict our discussion to ridge surfaces. Valleys are obtained by straightforward analogies or by extracting ridges of $-f$.

4.2 Finding Edge Intersections

Our algorithm makes extensive use of a differentiable symmetric tensor field $\mathbf{T}(\mathbf{x})$ which is derived from the Hessian field $\mathbf{H}(\mathbf{x})$ and has $\mathbf{g}(\mathbf{x})$ as an eigenvector to eigenvalue 1 if and only if $\mathbf{x}$ is a point on the ridge. Its eigenvectors $\mathbf{e}'_i$ and eigenvalues λ'_i are defined from those of the Hessian matrix ($\mathbf{e}_i$ and λ_i) as

$$\mathbf{e}'_i := \mathbf{e}_i, \quad \lambda'_1 := 1, \quad \lambda'_2 := 1,$$
$$\lambda'_3 := \begin{cases} 0, & \text{if } \lambda_2 - \lambda_3 > \theta \\ \left(1 - \dfrac{\lambda_2 - \lambda_3}{\theta}\right)^2, & \text{else.} \end{cases} \tag{4}$$

This definition makes sure that $\mathbf{T}$ is a differentiable function of $\mathbf{H}$ and that it remains well-defined as $(\lambda_2 - \lambda_3) \to 0$ and $\mathbf{e}_3$ becomes ill-conditioned. We assume that this starts to play a role when $(\lambda_2 - \lambda_3)$ drops below a threshold θ, which we empirically fixed at 0.5 percent of the dynamic range in our data. It is reasonable to count such loci as being on the ridge regardless of $\mathbf{g}$—as we have shown in Section 3.2, they form one type of surface boundary.

To detect intersections of the ridge with cell edges, we consider the vector

$$\mathbf{h}(\mathbf{x}) := \mathbf{T}(\mathbf{x})\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{x}), \tag{5}$$

which is zero if and only if $\mathbf{x}$ is a point on the ridge. Otherwise, $\mathbf{h}$ indicates the direction in which the gradient $\mathbf{g}$ moves when being projected onto the eigenplane of $\mathbf{T}$ (Fig. 4a). Let $\mathbf{h}_1$ and $\mathbf{h}_2$ be the respective vectors at the endpoints of an edge. We assume that the ridge intersects the edge if $\mathbf{h}_1 \cdot \mathbf{h}_2 < 0$, i.e., if the gradient has changed from one side of the eigenplane to the other (Fig. 4b). An estimate of the point of intersection is given by the relative magnitudes of $\mathbf{h}_1$ and $\mathbf{h}_2$. This way of finding edge intersections does not require to orient eigenvectors.

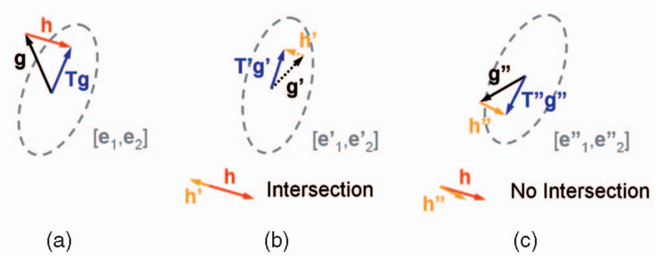


Fig. 4. An intersection of $\mathbf{g}$ with the plane spanned by $\mathbf{e}_1$ and $\mathbf{e}_2$ is detected by considering the difference vector $\mathbf{h}$ that results from projecting $\mathbf{g}$ to the plane.

It may appear even easier to locate edge intersections by bracketing zero crossings in the scalar d from (3). Unfortunately, d often has very close pairs of zero crossings, of which only one indicates a ridge and which are difficult to find in practice. This happens near parallel vector lines $\mathbf{g} \parallel \mathbf{e}_i$, for the reasons mentioned in Section 3.3.

4.3 Extracting the Boundary

To find the end points of the ridge on the cell faces, we localize type L Hessian degeneracies via the gradient descent proposed by Zheng and Pang [46]. We have augmented it with an Armijo step-size selection [1] to improve its convergence properties and repeat it from different starting positions on the face in cases where it runs into local minima. To save computations, this is only done on faces whose boundary is intersected an odd number of times.

Each ridge that enters a face should either leave it again or end in a type L degeneracy. It is important to ensure this algorithmically to achieve a consistent final triangulation. If no degeneracy is found, this typically means that we have missed an edge intersection. In fact, edges along which $\mathbf{T}$ varies strongly may be intersected multiple times. To handle this, we bisect an edge if the values $\mathbf{T}_1$ and $\mathbf{T}_2$ at its endpoints differ too much. The exact condition used in our current implementation is

$$\frac{\mathrm{tr}(\mathbf{T}_1^{\mathrm{T}}\mathbf{T}_2)}{\sqrt{\mathrm{tr}(\mathbf{T}_1^{\mathrm{T}}\mathbf{T}_1)} \cdot \sqrt{\mathrm{tr}(\mathbf{T}_2^{\mathrm{T}}\mathbf{T}_2)}} < \Theta, \tag{6}$$

where $\mathbf{T}^{\mathrm{T}}$ denotes transpose of $\mathbf{T}$, tr is matrix trace, and Θ is increased iteratively while the total number of intersections is odd and no degeneracy has been found.

4.4 Estimating Normals

The fact that $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ for all points $\mathbf{x}$ on the surface allows us to compute the surface normal at $\mathbf{x}$. Directional derivatives of $\mathbf{h}$ tangential to the surface are $\mathbf{0}$, so the Jacobian $\nabla\mathbf{h}$ has rank one, with the only nonzero eigenvector in normal direction. The fact that the normal computed this way is only defined up to sign is not a limitation, since the ridge surface is nonorientable anyway.

Since $\nabla\mathbf{h} = \nabla\mathbf{T}\mathbf{g} + \mathbf{T}\nabla\mathbf{g} - \nabla\mathbf{g}$ involves the gradient of $\mathbf{T}$, which is, in turn, defined in terms of the Hessian, normal

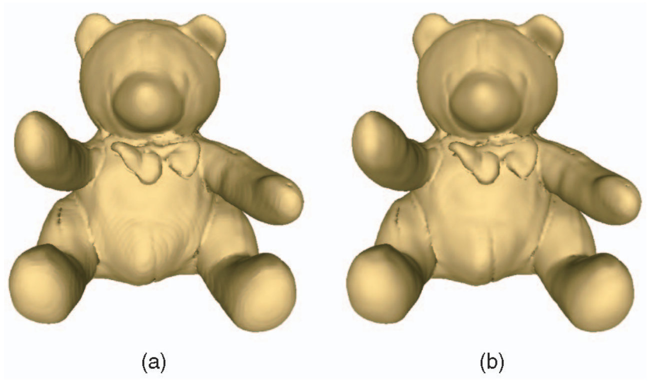


Fig. 5. (b) Despite the use of third derivatives, our analytic surface normals appear smooth. For comparison, (a) shows normals estimated from the mesh, as in [38].

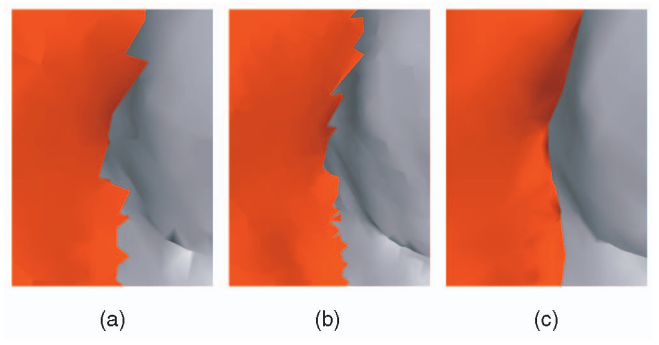


Fig. 6. Unlike (a) marching cubes and (b) marching cubes at double resolution, (c) our method creates a smooth representation of crease surfaces that terminate at degenerate lines.

estimation assumes that f is at least C^3 continuous. Despite dealing with third derivatives and applying some computational simplifications (detailed in Section 4.6), we found that the normals obtained this way are of reasonable quality and can be used both for estimating local topology and for rendering (cf. Fig. 5).

4.5 Generating the Mesh

Our theoretical analysis in Section 3 revealed two facts: First, crease surfaces end at degenerate lines; second, they may not be orientable. To reflect these insights, per-cell processing has to concentrate on finding the lines that bound the surface. Orientability, on the other hand, is a global property, and will result automatically if we make the correct per-cell decisions.

Consistency of the final mesh is guaranteed by estimating the connectivity per face, and sharing the results between adjacent cells. Moreover, the boundary points are connected pairwise per cell, and thus, form continuous lines in the final mesh. As an example, Fig. 6 illustrates the case discussed in Section 3.2, in which one part of a crease (red) terminates just before it would meet another one (gray) in a nonmanifold configuration. Since marching cubes does not allow the surface to terminate within a cell, it produces a zig-zag edge (Fig. 6a), even when using a finer resolution (Fig. 6b). On the other hand, our method extracts a smooth degenerate line which bounds the surface.

To estimate per-face connectivity, we connect the extracted intersection points pairwise. Since the total number of points per face is low, we simply enumerate all possible pairings and exclude the ones that would lead to a self-intersection (2D line-line intersection test). Among the permissible options, we choose the one which agrees best with the computed normals, i.e., the one which minimizes the sum of absolute dot products of connection lines and normals at their end points.

On each face, we extracted a degeneracy if and only if its boundary was intersected an odd number of times. Since edges are shared between adjacent faces, this leads to an even number of degenerate points per cell, which are connected in a similar manner. In rare cases, a cell has more than two degenerate points. In that case, we use the quads defined by any pair of degenerate points and their respective neighbors on the face to check for self-intersections (3D triangle-triangle intersection tests after arbitrary subdivision).

After these steps, each cell contains a set of closed polygons (cf. Fig. 3e). Triangles are used as is, and quads are subdivided arbitrarily. We triangulate larger polygons via a triangle fan with an additional vertex at the barycenter.

4.6 Implementation

Trilinear interpolation is widely used for its computational efficiency. Since creases require C^2 continuity, more advanced interpolation becomes obligatory. Like Kindlmann et al. [17], we convolve the given sample points with a C^2 cubic B-spline kernel. However, we store the resulting values, gradients, and Hessians at each grid point and interpolate them trilinearly in between. A very similar approximation is made when using the Phong shading model [28], which interpolates surface position and normal independently.

We found that this approximation greatly speeds up the bisection of edges and the iterative search for degenerate points on faces, while the resulting changes to the crease are on the order of a small additive Gaussian perturbation of f. Even approximating third derivatives by taking finite differences in the trilinearly interpolated Hessian field did not introduce any notable artifacts in the resulting normals. Note that this choice is an implementation detail which could be changed without having to alter any part of the algorithm.

Our extraction algorithm only produces exact boundaries where the ridge ends at a degenerate line. The side constraint ($\lambda_3 < 0$) is taken into account by excluding cells for which no vertex meets the constraint. This causes zig-zag boundaries, which are straightened by triangle trimming in a postprocess. This choice was motivated by the fact that crease surfaces are typically filtered using application-specific rules anyway, so both tasks are easily combined. Moreover, it avoids complex special cases that would otherwise occur in the extraction when the two types of boundaries meet.

4.7 Rendering

On modern graphics hardware, it is straightforward to render nonoriented surfaces, simply by discarding the sign of the normal in the lighting computation. As an example, let $\mathbf{n}$ denote the surface normal, $\mathbf{l}$ the vector toward the light source. The diffuse term in the Phong shading model [28] usually involves $\max\{\mathbf{n} \cdot \mathbf{l}, 0\}$. To render nonoriented

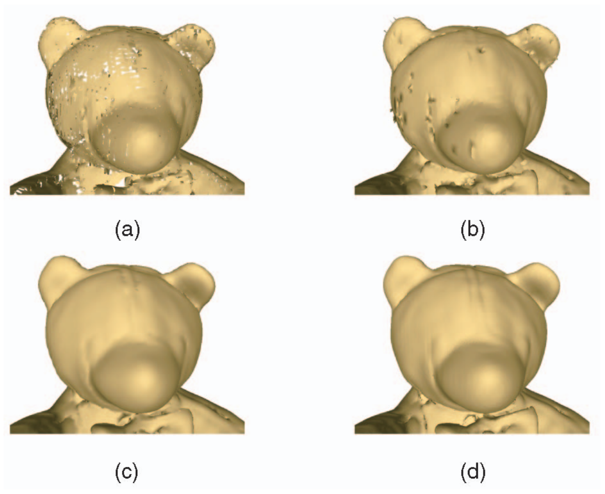


Fig. 7. At the original data resolution, (c) the proposed algorithm for crease extraction provides much better results than (a) marching cubes with eigenvector tracking and (b) marching cubes with PCA. For comparison, (d) shows results from eigenvector tracking at double resolution.

surfaces, we simply replace this expression with $|\mathbf{n} \cdot \mathbf{l}|$ in a vertex shader program [31].

5 Results

5.1 Setup and Qualitative Results

To validate our method, we extracted boundary ridges in a volume data set from a CT scan of a teddy bear. We chose this data set because the bear is composed from different materials, which makes it difficult to extract using simple isosurfacing. We resampled the data set to $118 \times 118 \times 105$ cells with isotropic edge length $l = 3\,\mathrm{mm}$. To detect the boundaries, we computed the gradient magnitude by convolution with the directional derivative of Gaussian kernels at $\sigma = 3.3\,\mathrm{mm}$. From the resulting data set, we then extracted height ridges at the data grid resolution using the proposed method, and compared them to results of marching cubes, using both eigenvector tracking (as in [17]) and PCA (as in [32]) as a preprocess.

All previous authors have found it necessary to filter out noise-related parts of creases. Like Haralick [13], we used a threshold on the ratio of gradient magnitude over λ_3 to restrict the ridge to its most salient part. Moreover, we put a threshold on absolute value and performed connected component analysis to remove a background object present in the data set. As shown in Fig. 7, the visual impression of our result (c) is clearly better than the ones from marching cubes at the same resolution.

Eigenvector tracking cannot process cells in which any edge is near a type L Hessian degeneracy. These skipped cells lead to the large number of small holes in (a). PCA processes all cells, but fails to find a consistent orientation in the presence of Hessian degeneracies or large eigenvector variations. This leads to the spikes and holes in (b). The degeneracies which cause these problems run through the same cells as the affected parts of the bear, but belong to the surfaces that end in the vicinity of the bear and are filtered out during postprocessing. Since the effect of Hessian degeneracies is not controlled in previous methods, they can affect any surface that intersects the cell. This problem can be mitigated by refining the extraction grid, which makes it less likely that a degeneracy of a "noise" ridge runs through the same cell as a legitimate ridge we would like to keep.

Previous authors have exploited this: Sadlo and Peikert [32] propose an adaptive refinement around the crease, and Kindlmann et al. [17] globally use a grid which is by factor 5 finer than the data grid. Indeed, at twice the original resolution, marching cubes with eigenvector tracking produce a result which looks comparable to ours (d). However, adaptive refinement cannot avoid ragged crease boundaries (cf. Fig. 6 and Table 3), leading to overtesselation and coming at considerable computational expense (cf. Table 1).

TABLE 1
Timings and Triangle Counts, Including Filtering

Method	Time (s)	# triangles
evec tracking	24+1	66,662
evec tracking (2× res.)	180+4	578,489
PCA	18+2	201,068
proposed method	38+3	246,040

TABLE 2
Mean Absolute and RMS (in Italics) Error in Face Position, as Measured by a Gradient Descent

Method	area	error (mm)
evec tracking	$0.21\,\mathrm{m}^2$	0.02 *(0.14)*
PCA	$0.49\,\mathrm{m}^2$	0.07 *(0.31)*
proposed method	$0.47\,\mathrm{m}^2$	0.02 *(0.13)*

5.2 Quantitative Results

Table 1 presents algorithm performance, in terms of consumed wall time (on a 2 GHz laptop) and generated geometry. It shows that the improved accuracy of our algorithm comes at moderate additional computational expense. In particular, it is more than four times faster than marching cubes on the refined grid, the only alternative that provides acceptable quality.

Moreover, we conducted two quantitative experiments which support the observations from the previous section. First, we evaluated the accuracy of the extracted surfaces by taking a large number of samples from the mesh ($1.5\,\mathrm{mm}^{-2}$, uniformly at random), and measuring the distance to the nearest point on the crease, as found by a gradient descent in the direction which minimizes the squared norm $\|\mathbf{h}\|^2$ of $\mathbf{h}$ from (5). This gradient descent is only used for evaluation, not during crease extraction. Table 2 lists the resulting average absolute and root-mean-square (RMS, in italics) distances. It clearly shows the increased error of the PCA result, which is due to erroneous triangles. The table also lists the total area of the bear, illustrating the fact that eigenvector tracking only reconstructs part of the surface.

In a second experiment, we considered the boundary components of the meshes and created a histogram of their length in terms of individual edges. The results in Table 3 show that marching cubes produce small holes in the surface. In particular, eigenvector tracking at both resolutions misses a large number of single triangles, due to skipped cells. Vertices in which more than two boundary edges meet are an

TABLE 3
Marching Cubes Produces a Large Number of Spurious Short Boundary Components on Crease Surfaces

	Bdy components of length				
Method	**3**	**4**	**5**	**6**	**> 6**
evec tracking	2158	59	408	69	353
evec tracking ($2\times$ res.)	2277	97	469	80	533
PCA	44	31	20	67	923
proposed method	0	10	0	0	136

indicator of spurious holes. In eigenvector tracking at the original resolution, more than 7 percent of all boundary vertices are affected. In marching cubes with PCA, it is slightly less than 1 percent; in our proposed method, such configurations do not occur by design. Note that many of the longer boundary components are a consequence of the fact that the crease also represents the stuffing of the bear.

For further validation, it would have been ideal to implicitly represent a known surface as a crease and to compare the mesh extracted from the resulting scalar field to the initial ground truth. Unfortunately, it is not obvious how to transform a surface to a well-defined height crease, and to the best of our knowledge, this topic has not been addressed in the literature.

6 Application to Diffusion Tensor MRI

Diffusion tensor magnetic resonance imaging is a medical imaging modality for noninvasive investigation of nerve fiber tracts in the human brain ([2], cf. [30] for a recent overview). In each voxel, DT-MRI estimates a diffusion tensor, a symmetric 3×3 matrix which models the Brownian motion of water molecules. Since this motion is restricted by nerve fibers, the main diffusion direction can be taken as an estimate of fiber direction in voxels where a single orientation prevails.

A standard way to visualize DT-MRI data is to integrate streamlines which are everywhere tangential to the principal eigenvector of the tensor field and are interpreted as estimated fiber pathways [24]. However, this method is inappropriate for regions where fiber tracts cross or fan out, since the diffusion tensor becomes planar, i.e., its larger two eigenvalues are similar in magnitude, and there is no single preferred direction.

To transfer the idea of streamlines to such areas, Zhang et al. [44] proposed to integrate stream surfaces, which are everywhere tangential to the plane spanned by the major and medium eigenvectors. They included the important caveat that this definition relies on the assumption that the Lie bracket of the involved eigenvector fields lies within their common plane. They considered it overly complex to verify that assumption, but stated that it would likely be fulfilled, since they did not experience problems in practice. Follow-up work [36], [41] has extended the original algorithm, but did not try to verify the condition on which it is founded. Currently, stream surfaces are frequently mentioned as a standard tool for DT-MRI visualization (cf. [45], [42], [30]).

In this section, we show that stream surfaces are, in fact, ill-defined: In typical DT-MRI data, there do not exist surfaces which are everywhere tangential to the plane

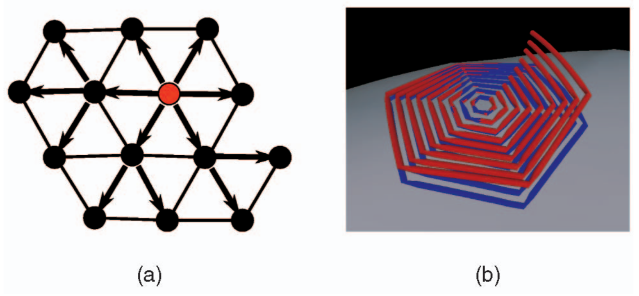


Fig. 8. (a) In stream surface extraction, vertices are added radially from the seed, breadth first. If the surface were well-defined, the algorithm would not depend on this particular order, and the red and blue lines in (b) would coincide.

spanned by the major and medium eigenvectors. Consequently, stream surfaces do not have a clear interpretation, since their shape is strongly influenced by arbitrary choices in their extraction.

In a second step, we use the notion of height creases to propose an alternative, well-defined set of surfaces to visualize planar regions of DT-MRI data, and employ our algorithm to extract them.

6.1 DT-MRI Stream Surfaces Are Ill-Defined

The algorithm in [44] extracts stream surfaces by growing a mesh of equilateral triangles from a given seed point. Zhang et al. perform the integration along the edges which are marked by arrows in Fig. 8a, but this choice is arbitrary. Their integrability condition in terms of the Lie bracket has an alternative formulation which is much easier to check in practice: If the surface resulting from their algorithm is well-defined, finding the position of a vertex by integrating along any other path in the mesh should produce the same result. In particular, integration along cycles should return to the initial position.

To test this, we integrated cycles along the n-rings $n \in \{1, \dots, 10\}$ around the seed point. Integration started at a vertex of the previously extracted stream surface and the first step was made in the direction of its neighbor on the n-ring in the counterclockwise direction. Further integration was carried out in the plane spanned by the minor eigenvector and the "incoming" vector of the previous step, using the exact rule from [44]. After each nth step, we turned the incoming vector 60 degree to the left within the current tangent plane.

If the rule for surface integration were well-defined, the resulting trajectory should coincide with the corresponding n-ring on the surface. Fig. 8b shows that this condition is violated: On the presented stream surface (gray), the n-rings are shown in blue. Our trajectories, which clearly depart from the surface, are red. Since we made conservative choices for step size (one-fifth of a cell edge length) and numerical integration scheme (fifth-order Runge-Kutta at 64-bit floating-point precision), such strong differences in such a small neighborhood cannot be explained by numerical errors. Also, integration was limited to a domain where the second eigenvalue was much larger than the third, so degeneracies have not played a role.

The algorithm in [44] expands the surface breadth-first, so adjacent vertices are integrated along similar paths. In our experience, the algorithm becomes unstable when this order is changed to depth-first, which should not be the case if the surface were well-defined. Moreover, when allowing for holes in the surface (as in [41]), highly deformed triangles occur when a boundary component is closed and vertices whose integration paths had departed for some time become adjacent again.

Effectively, stream surface integration tries to find a surface which is everywhere perpendicular to the minor eigenvector field. In computer vision, it is a well-studied problem that such surfaces only exist for vector fields whose derivatives obey a specific symmetry [10]: In shape from shading, an estimated normal field is used to infer surface geometry. In this context, there exist various strategies to deal with "nonintegrable" vector fields (cf. [5] and references therein). While it appears possible to avoid the above-mentioned algorithmic problems by adopting such methods, we feel that interpretation of the resulting surfaces would be unclear.

We would like to emphasize that our result does not affect stream surfaces in the sense in which they are traditionally defined in vector field visualization (i.e., as a surface which is traced out by a given seed line when advected along the field). Stream surfaces in this well-defined sense have been used for tensor field visualization by Jeremić et al. [14]. Sondershaus and Gumhold [36] even decide to call the "stream surfaces" in the context of DT-MRI "diffusion surfaces" to avoid confusion of these different definitions.

6.2 Planarity Ridges for DT-MRI Visualization

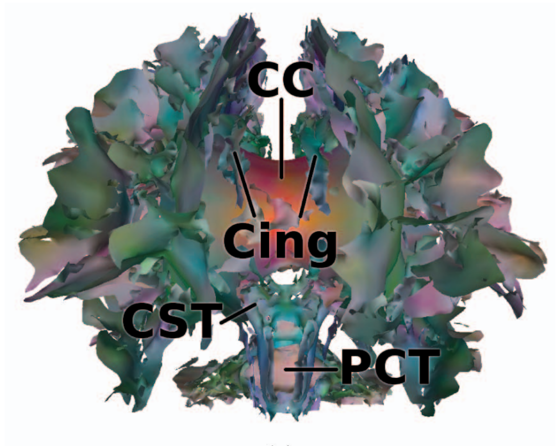
Our proposed substitute for stream surfaces in DT-MRI is closely related to the anisotropy creases which were used by Kindlmann et al. [16] to delineate the skeleton of white matter structures and have been shown to produce repeatable results over a range of subjects [17].

Kindlmann et al. extract ridges of fractional anisotropy (FA), a scalar measure which quantifies the overall directional dependance of diffusion. To investigate only planar regions, we replace FA with c_p, a specific measure of planarity introduced by Westin et al. [43]

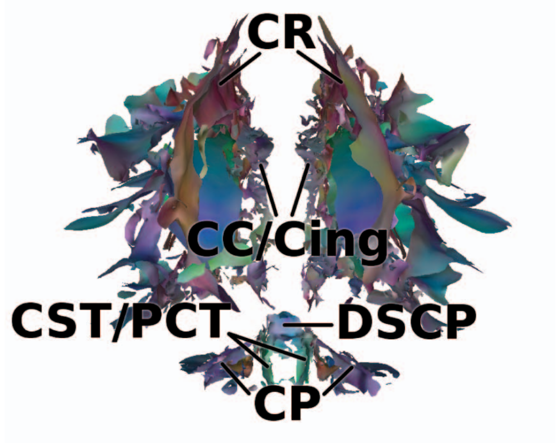
$$c_p = \frac{2(\lambda_2 - \lambda_3)}{\lambda_1 + \lambda_2 + \lambda_3}. \tag{7}$$

Extracting creases of c_p requires formulas for the first and second partial derivatives of c_p, which are derived in the appendix of this paper. To avoid problems at degeneracies, at which sorted eigenvalues may not be differentiable, we employ the regularized eigenvalue derivatives from [35].

Fig. 9 compares ridges in FA and c_p in a frontal view of an example data set (DT-MRI data with $93 \times 116 \times 93$ voxels, isotropic edge length $1.72\ \mathrm{mm}$). Gaussian prefiltering with $\sigma = 1.72\ \mathrm{mm}$ was used and the ridges were filtered to areas with $\mathrm{FA} > 0.2$ (Fig. 9a) and $c_p > 0.2$ (Fig. 9b), respectively. Note that different color schemes are used: Fig. 9a employs standard RGB-XYZ coloring of the major eigenvector, while Fig. 9b color codes the minor eigenvector (e.g., red denotes fanning perpendicular to the x-axis), since no principal direction may be defined in planar regions.



(a)



(b)

Fig. 9. Unlike ridges in FA (a), ridges in c_p (b) specifically illustrate the cores of planar regions. Therefore, they are a suitable replacement for the ill-defined stream surfaces.

As expected, ridges in c_p show the cores of planar regions: They capture the fanning in the *corona radiata* (CR) and the *cerebellar peduncles* (CP), the crossing at the *decussation of the superior cerebellar peduncle* (DSCP), and due to partial voluming, interfaces between *corpus callosum* and *cingulum* (CC/Cing), as well as between *corticospinal tract* and *pontine crossing tract* (CST/PCT). In comparison, the FA ridge (Fig. 9a) also includes structures with linear diffusion, like the CC, Cing, CST, and PCT. Unlike stream surfaces, planarity ridges cannot be integrated from arbitrary positions, which alleviates issues of seeding and culling. Their parameters are scale (amount of presmoothing) and a threshold for postfiltering.

Our novel algorithm for crease extraction facilitates the processing of full-brain DT-MRI scans at the original resolution: Fig. 9a was extracted on the original data grid within $26\ \mathrm{s}$, while Kindlmann et al. [17] report 6 minutes even after subsampling their data by a factor of two (i.e., to $48 \times 48 \times 28$ voxels), mainly due to the fact that they had to use an extremely fine extraction grid. Despite the fact that as a nonlinear function, FA has higher spatial frequency than the underlying tensor field itself [17], direct visual comparison between creases extracted from the approximation in Section 4.6 and ones from exact derivatives did not reveal any notable differences.

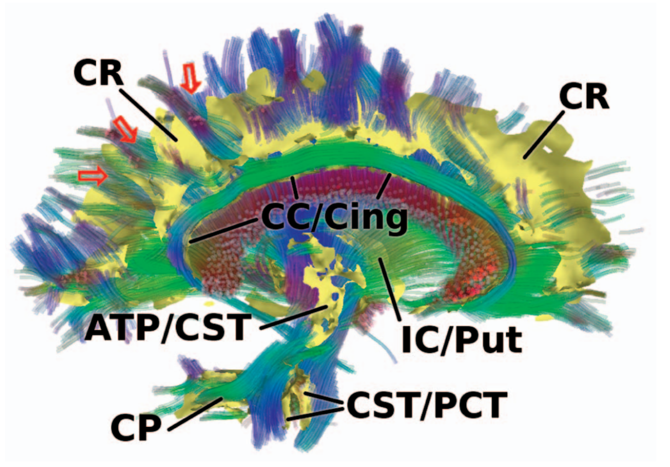


Fig. 10. Merging the planarity ridge with semitransparent streamlines made it easier to recognize the anatomical relevance of its components in this medial view.

6.3 Evaluation of Planarity Ridges

For evaluation, we presented our planarity ridges to a neuroscientist. In this process, we found it helpful to add further anatomical context. We added streamlines from fiber tracking [24], which we rendered semitransparently to mitigate problems with occlusion. Moreover, we seeded superquadric glyphs [15] on the surface.

Fig. 10 shows an overview of the left hemisphere, seen from a medial cutting plane. The fused rendering with the streamlines allowed our collaborator to confirm our annotations from Fig. 9b and, in addition, to identify planar regions corresponding to the interfaces between *anterior thalamic projections* and *corticospinal tract* (ATP/CST), as well as between *internal capsule* and *putamen* (IC/Put). Within the *precuneus*, the planarity ridge exhibits some characteristic dents (red arrows). In this region, the planarity is due to the fanning of the *corona radiata* (CR) and to its intersection with the SLF III, a component of the *superior longitudinal fasciculus*. It is weaker in places where fiber bundles run, in a more coherent manner, into one of the cortical gyri.

To get a more detailed view on a part of the planarity ridge, Fig. 11 presents a closeup of the right hemisphere, near the *lateral sulcus.* The annotated tracts in Fig. 11a are the *superior longitudinal fasciculus* (SLF) which intersects with the *transcallosal fibers* (TF) and the *short association fibers* (SF), the *subinsular white matter* (SI), the *inferior fronto-occipital fasciculus* (IFO) which intermingles with the *uncinate fasciculus* (Unc), as well as the *inferior longitudinal fasciculus* (ILF).

The superquadric glyphs in Fig. 11b confirm that the planarity ridges in this region capture the intersection of SLF with TF and SF, and the bifurcation of IFO and Unc. Moreover, a planar region exists in the *external/extreme capsule* (EC), where the tracts of the *subinsular white matter* (SI) originate. For our evaluation needs, we found it sufficient to place the glyphs via a simple stratified surface sampling. If desired, a more even distribution could be achieved by implementing glyph packing [18] on the surfaces.

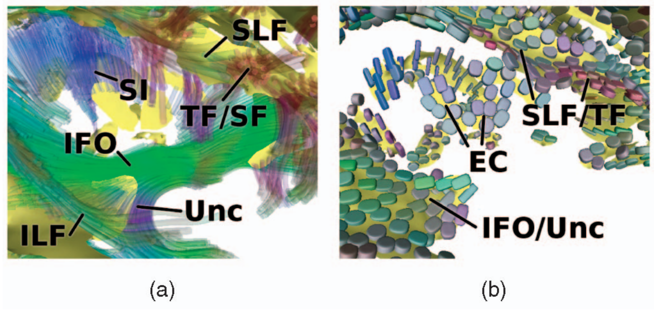


Fig. 11. (a) Besides using streamlines, (b) seeding superquadric glyphs on the surface helped to identify planarity ridges in this lateral closeup.

7 Conclusion

Crease lines have a long tradition in image processing and computer vision. We are convinced that once crease surfaces, which constitute their two-dimensional generalization, are fully understood and reliable numerical methods are available for their extraction, they will also offer a versatile visualization tool, both to capture boundaries which cannot be characterized as isosurfaces and to extract object cores or skeletal structures.

Our work has promoted this research goal by clarifying the topological properties of crease surfaces and proposing a novel algorithm for their extraction, which we have shown to be more reliable than existing methods. The transformed Hessian approach in Section 4.2 provides a unified framework for detection of crease surface intersections, estimation of surface normals, and for a gradient descent to the crease surface, without the need to orient eigenvectors.

A second contribution of our work is to demonstrate that the so-called stream surfaces, which have been considered as a standard tool for DT-MRI visualization, are mathematically ill-defined and should not be used. As a well-defined alternative, we propose planarity ridge surfaces for the visual analysis of planar regions. Our new algorithm is crucial for their extraction on full-brain data sets at original resolution.

Prior work has shown that visualizing creases at a single scale can already provide valuable insights [16], [32]. However, to fully harness the potential of creases, we would like to take their scale space behavior into account. This requires the extraction of surfaces from a four-dimensional space, which is an aspect beyond the scope of this work. However, understanding crease surfaces in 3D is a necessary first step toward that more complex goal.

Appendix

Partial Derivatives of c_p

To extract creases of c_p (7), we need to find its first and second partial derivatives with respect to the tensor field. Without loss of generality, we only consider $c_{p,x} := \frac{\partial c_p}{\partial x}$ and $c_{p,xy} := \frac{\partial^2 c_p}{\partial x \partial y}$. According to the quotient rule, they are given as

$$c_{p,x} = \frac{A}{B}, \quad c_{p,xy} = \frac{A_y B - A B_y}{B^2},$$

with

$$\begin{aligned}A &= 2\lambda_1\big(\lambda_{2,x} - \lambda_{3,x}\big) + 2\lambda_2\big(-\lambda_{1,x} - 2\lambda_{3,x}\big)\\ &\quad + 2\lambda_3\big(\lambda_{1,x} + 2\lambda_{2,x}\big),\\ A_y &= 2\lambda_1\big(\lambda_{2,xy} - \lambda_{3,xy}\big) + 2\lambda_2\big(-\lambda_{1,xy} - 2\lambda_{3,xy}\big)\\ &\quad + 2\lambda_3\big(\lambda_{1,xy} + 2\lambda_{2,xy}\big) + 2\lambda_{1,y}\big(\lambda_{2,x} - \lambda_{3,x}\big)\\ &\quad + 2\lambda_{2,y}\big(-\lambda_{1,x} - 2\lambda_{3,x}\big) + 2\lambda_{3,y}\big(\lambda_{1,x} + 2\lambda_{2,x}\big),\\ B &= (\lambda_1 + \lambda_2 + \lambda_3)^2,\\ B_y &= 2(\lambda_1 + \lambda_2 + \lambda_3)\big(\lambda_{1,y} + \lambda_{2,y} + \lambda_{3,y}\big).\end{aligned}$$

First partial eigenvalue derivatives $\lambda_{i,x}$ are found by rotating the corresponding tensor derivative $\mathbf{D}_x$ to the eigenframe of the original tensor $\mathbf{D}$ (cf. [35] for treatment of repeated eigenvalues). Second partial eigenvalue derivatives $\lambda_{i,xy}$ are given by rotating the second partial tensor derivative $\mathbf{D}_{xy}$ to the same frame, but additionally require a correction based on first *eigenvector* derivatives $\mathbf{e}_{i,x}$. Let $\mathbf{I}$ denote the identity matrix and $\mathbf{T}^+$ the Moore-Penrose inverse of $\mathbf{T}$. Then, [22]

$$\begin{aligned}\lambda_{i,x} &= \mathbf{e}_i^{\mathrm{T}}\mathbf{D}_x\mathbf{e}_i,\\ \mathbf{e}_{i,x} &= (\lambda_i\mathbf{I} - \mathbf{D})^+\mathbf{D}_x\mathbf{e}_i,\\ \lambda_{i,xy} &= \mathbf{e}_i^{\mathrm{T}}\mathbf{D}_{xy}\mathbf{e}_i + \mathbf{e}_i^{\mathrm{T}}\mathbf{D}_x\mathbf{e}_{i,y} + \mathbf{e}_i^{\mathrm{T}}\mathbf{D}_y\mathbf{e}_{i,x}.\end{aligned}$$

Acknowledgments

The authors are grateful to Alfred Anwander, who is with the Max Planck Institute for Human Cognitive Brain Science, Leipzig, Germany, for providing the diffusion MRI data and helping to evaluate our results. They would like to thank Petra Specht for her help in creating Fig. 1. The teddy bear data set is courtesy of Bernd Tomandl (Division of Neuroradiology, University of Erlangen, Germany) and available from www.volvis.org. They used the software teem (teem.sf.net) both for its implementation of the marching cube-based crease extraction and as a framework for their own implementation. This work was partly funded by the Max Planck Center for Visual Computing and Communication (MPC-VCC).

References

[1] L. Armijo, "Minimization of Functions Having Lipschitz Continuous First Partial Derivatives," *Pacific J. Math.,* vol. 16, no. 1, pp. 1-3, 1966.

[2] P.J. Basser, J. Mattiello, and D.L. Bihan, "Estimation of the Effective Self-Diffusion Tensor from the NMR Spin Echo." *J. Magnetic Resonance,* vol. B, no. 103, pp. 247-254, 1994.

[3] A.G. Belyaev, E.V. Anoshkina, and T.L. Kunii, "Ridges, Ravines and Singularities," *Topological Modeling for Visualization,* A.T. Fomenko and T.L. Kunii, eds., chapter 18, Springer, 1997.

[4] H. Blum and R.N. Nagel, "Shape Description Using Weighted Symmetric Axis Features," *Pattern Recognition,* vol. 10, pp. 167-180, 1978.

[5] J.Y. Chang, K.M. Lee, and S.U. Lee, "Multiview Normal Field Integration Using Level Set Methods," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR),* pp. 1-8, 2007.

[6] J. Damon, "Generic Structure of Two-Dimensional Images under Gaussian Blurring," *SIAM J. Applied Math.,* vol. 59, no. 1, pp. 97-138, 1998.

[7] T. Delmarcelle and L. Hesselink, "The Topology of Symmetric, Second-Order Tensor Fields," *Proc. IEEE Conf. Visualization,* R.D. Bergeron and A.E. Kaufman, eds., pp. 140-147, 1994.

[8] D. Eberly, *Ridges in Image and Data Analysis, Computational Imaging and Vision,* vol. 7. Kluwer Academic Publishers. 1996.

[9] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach, "Ridges for Image Analysis," *J. Math. Imaging and Vision,* vol. 4, pp. 353-373, 1994.

[10] R.T. Frankot and R. Chellappa, "A Method for Enforcing Integrability in Shape from Shading Algorithms," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 10, no. 4, pp. 439-451, July 1988.

[11] J.D. Furst and S.M. Pizer, "Marching Ridges," *Proc. Int'l Assoc. Science and Technology for Development (IASTED) Int'l Conf. Signal and Image Processing,* pp. 22-26, 2001.

[12] J.D. Furst, S.M. Pizer, and D.H. Eberly, "Marching Cores: A Method for Extracting Cores from 3D Medical Images," *Proc. Workshop Math. Methods in Biomedical Image Analysis,* pp. 124-130, 1996.

[13] R.M. Haralick, "Ridges and Valleys on Digital Images," *Computer Vision, Graphics, and Image Processing,* vol. 22, pp. 28-38, 1983.

[14] B. Jeremić, G. Scheuermann, J. Frey, Z. Yang, B. Hamann, K.I. Joy, and H. Hagen, "Tensor Visualizations in Computational Geomechanics," *Int'l J. Numerical and Analytical Methods in Geomechanics,* vol. 26, pp. 925-944, 2002.

[15] G. Kindlmann, "Superquadric Tensor Glyphs," *Proc. Eurographics/IEEE Symp. Visualization (SymVis),* pp. 147-154, 2004.

[16] G. Kindlmann, X. Tricoche, and C.-F. Westin, "Anisotropy Creases Delineate White Matter Structure in Diffusion Tensor MRI," R. Larsen, M. Nielsen, and J. Sporring, eds., *Proc. Ninth Int'l Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI),* pp. 126-133, 2006.

[17] G. Kindlmann, X. Tricoche, and C.-F. Westin, "Delineating White Matter Structure in Diffusion Tensor MRI with Anisotropy Creases," *Medical Image Analysis,* vol. 11, no. 5, pp. 492-502, 2007.

[18] G. Kindlmann and C.-F. Westin, "Diffusion Tensor Visualization with Glyph Packing," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 1329-1335, Sept./Oct. 2006.

[19] J.J. Koenderink and A.J. van Doorn, "Local Features of Smooth Shapes: Ridges and Courses," *Geometric Methods in Computer Vision II,* B.C. Vemuri, ed., pp. 2-13, 1993.

[20] T. Lindeberg, "Edge Detection and Ridge Detection with Automatic Scale Selection," *Int'l J. Computer Vision,* vol. 30, no. 2, pp. 117-154, 1998.

[21] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Proc. ACM SIGGRAPH '87,* pp. 163-169, 1987.

[22] J.R. Magnus and H. Neudecker, *Matrix Differential Calculus with Applications in Statistics and Econometrics,* rev. ed. Wiley, 1998.

[23] H. Miura and S. Kida, "Identification of Tubular Vortices in Turbulence," *J. Physical Soc. Japan,* vol. 66, no. 5, pp. 1331-1334, 1997.

[24] S. Mori, B.J. Crain, V.P. Chacko, and P.C.M. van Zijl, "Three-Dimensional Tracking of Axonal Projections in the Brain by Magnetic Resonance Imaging," *Ann. Neurology,* vol. 45, no. 2, pp. 265-269, 1999.

[25] B.S. Morse, *Computation of Object Cores from Grey-Level Images,* PhD thesis, Univ. of North Carolina at Chapel Hill, 1994.

[26] R. Peikert and M. Roth, "The 'Parallel Vectors' Operator—A Vector Field Visualization Primitive," *Proc. IEEE Visualization Conf. '99,* D.S. Ebert, M. Gross, and B. Hamann, eds., pp. 263-270, 1999.

[27] R. Peikert and F. Sadlo, "Height Ridge Computation and Filtering for Visualization," *Proc. IEEE Pacific Visualization 2008,* I. Fujishiro, H. Li, and K.-L. Ma, eds., pp. 119-126, 2008.

[28] B.T. Phong, "Illumination for Computer Generated Pictures," *Comm. ACM,* vol. 18, no. 6, pp. 311-317, 1975.

[29] S.M. Pizer, C.A. Burbeck, J.M. Coggins, D.S. Fritsch, and B.S. Morse, "Object Shape Before Boundary Shape: Scale-Space Medial Axes," *J. Math. Imaging and Vision* vol. 4, pp. 303-313, 1994.

[30] B. Preim and D. Bartz, *Visualization in Medicine. Theory, Algorithms, and Applications.* Morgan Kaufmann, 2007.

[31] R.J. Rost, *OpenGL Shading Manual,* second ed. Addison-Wesley, 2006.

[32] F. Sadlo and R. Peikert, "Efficient Visualization of Lagrangian Coherent Structures by Filtered AMR Ridge Extraction," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 6, pp. 1456-1463, Nov. 2007.

[33] J. Sahner, T. Weinkauf, and H.-C. Hege, "Galilean Invariant Extraction and Iconic Representation of Vortex Core Lines," *Proc. Eurographics/IEEE Visualization and Graphics Technical Committee (VGTC) Symp. Visualization (EuroVis),* K. Brodlie, D. Duke, and K. Joy, eds., pp. 151-160, 2005.

[34] J. Sahner, T. Weinkauf, N. Teuber, and H.-C. Hege, “Vortex and Strain Skeletons in Eulerian and Lagrangian Frames,” *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 5, pp. 980-990, Sept./Oct. 2007.
[35] T. Schultz and H.-P. Seidel, “Using Eigenvalue Derivatives for Edge Detection in DT-MRI Data,” *Pattern Recognition,* G. Rigoll, ed., pp. 193-202, 2008.
[36] R. Sondershaus and S. Gumhold, “Meshing of Diffusion Surfaces for Point-Based Tensor Field Visualization,” *Proc. 12th Int’l Meshing Roundtable (IMR),* pp. 177-188, 2003.
[37] J. Süßmuth and G. Greiner, “Ridge Based Curve and Surface Reconstruction,” *Proc. Eurographics Symp. Geometry Processing,* A. Belyaev and M. Garland, eds., pp. 243-251, 2007.
[38] G. Taubin, “Estimating the Tensor of Curvature of a Surface from a Polyhedral Approximation,” *Proc. Fifth Int’l Conf. Computer Vision (ICCV),* pp. 902-907, 1995.
[39] H. Theisel and H.-P. Seidel, “Feature Flow Fields,” *Proc. Eurographics/IEEE-TCVG Symp. Visualization (VisSym),* G.-P. Bonneau, S. Hahmann, and C.D. Hansen, eds., pp. 141-148, 2003.
[40] X. Tricoche, G. Kindlmann, and C.-F. Westin, “Invariant Crease Lines for Topological and Structural Analysis of Tensor Fields,” *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 6, pp. 1627-1634, Nov./Dec. 2008.
[41] A. Vilanova, G. Berenschot, and C. van Pul, “DTI Visualization with Stream Surfaces and Evenly-Spaced Volume Seeding,” *Proc. Joint Eurographics-IEEE Technical Committee on Visualization and Graphics (TCVG) Symp. Visualization (VisSym),* O. Deussen, C. Hansen, D.A. Keim, and D. Saupe, eds., pp. 173-182, 2004.
[42] A. Vilanova, S. Zhang, G. Kindlmann, and D.H. Laidlaw, “An Introduction to Visualization of Diffusion Tensor Imaging and Its Applications,” *Visualization and Processing of Tensor Fields,* J. Weickert and H. Hagen, eds., pp. 121-153, Springer, 2006.
[43] C.-F. Westin, S. Peled, H. Gudbjartsson, R. Kikinis, and F.A. Jolesz, “Geometrical Diffusion Measures for MRI from Tensor Basis Analysis,” *Proc. Int’l Soc. Magnetic Resonance in Medicine,* p. 1742, 1997.
[44] S. Zhang, C. Demiralp, and D.H. Laidlaw, “Visualizing Diffusion Tensor MR Images Using Streamtubes and Streamsurfaces,” *IEEE Trans. Visualization and Computer Graphics,* vol. 9, no. 4, pp. 454-462, Oct.-Dec. 2003.
[45] S. Zhang, D.H. Laidlaw, and G. Kindlmann, “Diffusion Tensor MRI Visualization,” *The Visualization Handbook,* C.D. Hansen and C.R. Johnson, eds., pp. 327-340, Elsevier, 2005.
[46] X. Zheng and A. Pang, “Topological Lines in 3D Tensor Fields,” *Proc. IEEE Visualization,* pp. 313-320, 2004.
[47] X. Zheng, B. Parlett, and A. Pang, “Topological Structures of 3D Tensor Fields,” *Proc. IEEE Visualization Conf.,* pp. 551-558, 2005.

Thomas Schultz studied medical informatics during 2001-2003 at the University of Heidelberg, Germany, and computer science during 2003-2006 at Saarland University, Germany, where he received the diploma (MS) degree in 2006. Since then, he has been working toward the PhD degreee at the MPI Informatik, Germany. His current research interests focus on medical and tensor field visualization, and on the interface of visualization and computer vision.

Holger Theisel received the MS degree in 1994, the PhD degree in 1996, and the habilitation degrees in 2001 from the University of Rostock, Germany, where he studied computer science from 1989-1994 and was a research and teaching assistant during 1995-2001. He spent 12 months from 1994/1995 as a visiting scholar at Arizona State University, and six months as a guest lecturer at ICIMAF Havana, Cuba. From 2002 to 2006, he was a member of the Computer Graphics Group at MPI Informatik Saarbrücken, Germany. In 2006/2007, he was a professor for computer graphics at Bielefeld University, Germany. Since October 2007, he is a professor for visual computing at the University of Magdeburg. His research interests focus on flow and volume visualization, as well as on CAGD, geometry processing, and information visualization.

Hans-Peter Seidel is the scientific director and chair of the Computer Graphics Group at the Max-Planck-Institut (MPI) Informatik and a professor of computer science at Saarland University. He has published some 200 technical papers in the field and has lectured widely. He has received grants from a wide range of organizations, including the German National Science Foundation (DFG), the German Federal Government (BMBF), the European Community (EU), NATO, and the German-Israel Foundation (GIF). In 2003, he received the “Leibniz Preis,” the most prestigious German research award, from the German Research Foundation (DFG). He is the first computer graphics researcher to receive such an award.