2D Tensor Field Segmentation*

Cornelia Auer[1], Jaya Sreevalsan-Nair[2], Valentin Zobel[3], and Ingrid Hotz[4]

1 **Zuse Institut Berlin**
Takustrasse 7, 14195 Berlin, Germany
`auer@zib.de`
2 **IIIT – Bangalore, Electronics City, Hosur Road, Bangalore, 560100, India**
`jnair@iiitb.ac.in`
3 **Zuse Institut Berlin**
Takustrasse 7, 14195 Berlin, Germany
`zobel@zib.de`
4 **Zuse Institut Berlin**
Takustrasse 7, 14195 Berlin, Germany
`hotz@zib.de`

Abstract

We present a topology-based segmentation as means for visualizing 2D symmetric tensor fields. The segmentation uses directional as well as eigenvalue characteristics of the underlying field to delineate cells of similar (or dissimilar) behavior in the tensor field. A special feature of the resulting cells is that their shape expresses the tensor behavior inside the cells and thus also can be considered as a kind of glyph representation. This allows a qualitative comprehension of important structures of the field. The resulting higher-level abstraction of the field provides valuable analysis. The extraction of the integral topological skeleton using both major and minor eigenvector fields serves as a structural pre-segmentation and renders all directional structures in the field. The resulting curvilinear cells are bounded by tensorlines and already delineate regions of equivalent eigenvector behavior. This pre-segmentation is further adaptively refined to achieve a segmentation reflecting regions of similar eigenvalue and eigenvector characteristics. Cell refinement involves both subdivision and merging of cells achieving a predetermined resolution, accuracy and uniformity of the segmentation. The buildingblocks of the approach can be intuitively customized to meet the demands or different applications. Application to tensor fields from numerical stress simulations demonstrates the effectiveness of our method.

1998 ACM Subject Classification I.3.3 [Computer Graphics]: Picture/Image Generation

Keywords and phrases Tensorfield visualization, surface topology

Digital Object Identifier 10.4230/DFU.Vol2.SciViz.2011.17

1 Introduction

Tensor fields occur in engineering and scientific simulations, either as intermediate product or as final result. Mostly, the analysis of the resulting data is based on scalar fields derived from these tensor fields. Since this approach often is insufficient to understand the entire physical process, there is an increasing interest in the analysis of tensor data itself. The wealth of information contained in tensor data, however, induces high data complexity making visualization, analysis, and finally understanding of the data a challenging problem. In

* This work was supportet by the DFG Emmy Noether Research Project.

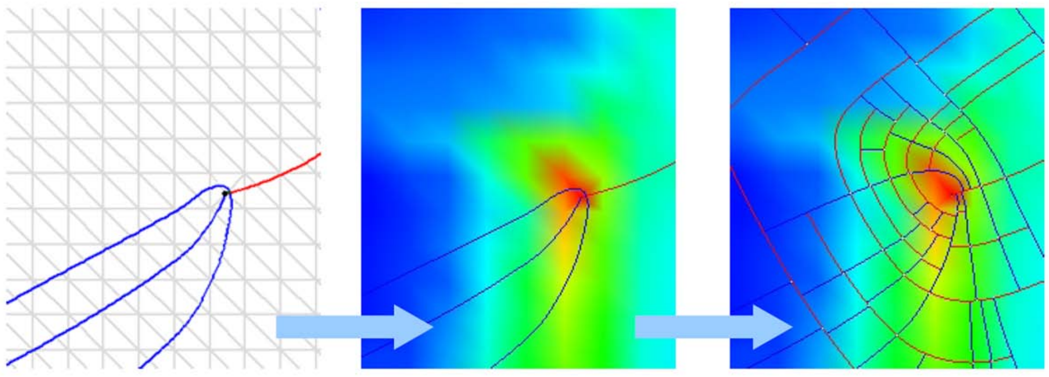

Figure 1 (a) Close up: Extraction of integral topological skeleton for pre-segmentation into cells of equivalent eigenvector behavior. (b) Color coding of scalar field reflecting the eigenvalue fields. (c) Adaptive segmentation into regions of similar eigenvector and eigenvalue behavior.

addition, researchers often do not exactly know what they are looking for, before getting used to the data. This lack of specific questions limits the use of feature extraction methods to reduce complexity. Thus, the goal is to provide an overview of the data without missing important details and without overwhelming the observer at the same time.

An essential step to reduce the amount of information is a segmentation that separates the field into regions of similar characteristic behavior. This higher level of abstraction allows a top-down exploration of the given dataset. Additionally the segmentation can be considered as a basis for visualization techniques using textures and glyphs.

This paper proposes a 2D tensor field segmentation guided by eigenvector and eigenvalue characteristics. Since the tensor is uniquely defined by these invariants, the segmentation gives the domain experts insight into the tensor field as a whole. The segmentation process consists of two steps:

- *Extraction of the integral topological graph considering both eigenvector fields* to provide a pre-segmentation, explained in Section 4. Thereby, the aim is not to show the correct topological structure but to use it as a basic frame, see Figure 1(a).
- *An adaptive segmentation workflow using the eigenvalue fields* to coarsen and subdivide the initial segmentation, explained in Section 5. The workflow consists of a set of building blocks, which can be flexibly combined to meet specific needs of the user or application, see Figure 1(c).

The adaptive refinement process of the segmentation is guided by the definition of a scalar invariant as similarity or dissimilarity measure, see Figure 1(b). Depending on the application, a variety of scalar invariants can be used. For example, anisotropy and maximum shear stress reflect the relation of eigenvalues and are of high importance in many applications. A generalization of the notion of anisotropy to non-positive definite fields allows us to extend our analysis to all tensor fields. In our approach, several dissimilarity measures can be flexibly applied either by themselves or as combinations.

The resulting segments themselves in the final segmentation can be considered as visualization glyphs in form of tiles. They are bounded by tensor lines which allows immediate interpretation of the eigenvector behavior within and color coding renders the eigenvalue characteristics in the segments. The geometry of the resulting segments is represented explicitly and offers statistical enquiry of properties inside each cell.

Section 5 explaines the basic concept by the implementation of a focus+context visualization. The segmentation is adapted to the demanded accuracy concerning eigenvector and -value similarity and to the given resolution of the displayed domain. Finally the extraction of *degenerate regions* in the field demonstrates the flexibility of the concept. Building blocks of the basic approach are adapted to a strategy capturing these often numerical unstable entities in tensor fields. The directional expression is weak, the only interest in these regions is in the isotropic behavior of the eigenvalues.
There has been a lot of recent research in extraction, simplification and visualization of tensor field topology, on which this work builds. Although, these methods extract valuable structural information from the eigenvector fields, they ignore the importance of the eigenvalues. The gap caused by the lack of interpretation of features in the eigenvalue field leads to decreased use of tensor topology in analysis. This work bridges this gap by providing a complete interpretation of the tensor field using the features present in both, the eigenvector as well as the eigenvalue fields. Its effectiveness is demonstrated using data sets from structural engineering, see Section 6.

2 Related Work

Still, the most common analysis methods for tensor fields are built on derived scalar fields. While this approach is often helpful it is not always sufficient. Due to the demand for tools representing the entire tensor information a variety of visualization methods have been developed. Since tensor fields have very application specific characteristics, these methods often are designed for concrete applications. Most efforts have been put into tensors from diffusion tensor imaging (DTI) and mechanical engineering applications, which is the focus of our work. In this area existing methods can roughly be classified into glyph-, texture- and topology-based methods.

Glyphs represent a direct visualization approach displaying tensor values in selected points. Related research issues are focused on the definition and placement of glyphs. Glyphs that are commonly used are ellipsoids, Haber glyphs [8], or superquadrics [12]. Different placement strategies are used to maximize the information displayed per image [7, 13]. A representation of tensor values on one-dimensional lines are hyperstreamlines. They are strongly related to streamline methods used for vector fields. They were introduced by Delmarcelle and Hesselink [6] and have been utilized in a geo-mechanical context by Jeremic et al. [11]. While glyphs are appropriate for characterizing single tensors, they are limited to low resolution and fail to give insight into the structure of the entire field. A more continuous view onto 2D fields can be obtained using tensor splats [3], or textures based on line integral convolution [9, 24].

For DTI , a lot effort has been put into tensor field segmentation, mostly with the goal of brain segmentation. Extending methods from image segmentation and clustering, the central research topic is the definition of an appropriate dissimilarity measure for tensors. Proposed methods range from active contours [21] and level sets [26] to graph-cut algorithms [23, 27]. Used metrics are the angular difference between principle eigenvector directions, or standard metrics considering the entire tensor, like the Euclidean or Frobenius distance. Recently, Wang et al. [21] introduced a distance measure from information theory designed for Gaussian distributions. Although, it is a good representation of the diffusion tensor characteristics, it is limited to positive definite tensors. A segmentation designed for meshes based on the curvature tensor was introduced by Lavoue et al. [15]. Vertices are clustered according to their principal curvature values using a k-means classification. The boundaries of resulting cells tend to be parallel to lines of minimum curvature but do not exactly represent the principal directions.

Methods concerned with the segmentation of general tensor fields are based on tensor field topology. They concentrate on the structure of the eigenvector fields neglecting the scalar entities. The idea of using topological methods to analyse the structures of 2D tensor fields goes back to Delmarcelle [5] and Lavin et al. [14] and builds the basis for the method proposed in this paper. They have introduced the topological skeleton consisting of degenerate points and connecting tensor lines as central features. Following this work, much effort has been put in simplifying and tracking of the resulting structures [19]. Alliez et al. [2] have proposed an application to curvature tensors for polygonal remeshing of surfaces. Zheng et al. [25] have initiated work in 3D tensor topology. Their analysis shows that in three dimensions degenerate features form one-dimensional structures. An eigenvector-based interpolation as basis for the topology extraction is proposed in [10]. An integral topological skeleton using both eigenvector fields has been used for a directional field segmentation in [17].

3 Basics and Notations

This section summarizes the basics for this paper. Definitions and notations are restricted to 2D tensor fields of second order, since they are the topic of this work. For a more complete and formal definition of tensors we refer readers to [22].

3.1 Tensors and Tensor Field

A tensor is a type of geometrical entity that generalizes the concept of scalars, vectors, and linear operators in a coordinate-independent fashion. With respect to a given basis of $I\!R^2$, a tensor $\mathbf{T}$ can be expressed by a 2×2-dimensional matrix of real numbers. $\mathbf{T}$ is called symmetric if for any coordinate basis, the corresponding matrix is symmetric. A tensor field over some domain $D \subset I\!R^2$ assigns to every point $P \in D$ a tensor $\mathbf{T}(P)$. In the rest of the paper, we will refer to symmetric 2D tensors of second order as tensor.

A tensor $\mathbf{T}$ is fully represented by its *eigenvalues* λ, μ and corresponding *eigenvectors* $\overset{\leftrightarrow}{v}$ and $\overset{\leftrightarrow}{w}$, implied by the eigenvalue equations $\mathbf{T}\cdot \overset{\leftrightarrow}{v}= \lambda\cdot \overset{\leftrightarrow}{v}$ and $\mathbf{T}\cdot \overset{\leftrightarrow}{w}= \mu\cdot \overset{\leftrightarrow}{w}$. The names λ and μ are assigned in a way, such that always $\lambda \geq \mu$. Since the multiplication of an eigenvector by any non-zero scalar yields an additional eigenvector, eigenvectors should be considered without norm and orientation, which distinguishes them from classical vectors. We use $\overset{\leftrightarrow}{v}$ and $\overset{\leftrightarrow}{w}$ when referring to eigenvectors to allude to the fact that they are bidirectional. We use $\mathbf{v}$ and $\mathbf{w}$ when referring to vectors representing normalized eigenvectors with an arbitrarily but fixed direction, e.g., using the unmodified results of the numerical computation used to generate them. The direction of $\mathbf{w}$ is defined in such a way that $\mathbf{v}$ and $\mathbf{w}$ form a right-handed system. Eigenvalues are computed by solving the characteristic equation, which is a quadratic equation in λ: $|\mathbf{T}-\lambda\mathbf{I}| = 0$, where $\mathbf{I}$ is identity matrix. For symmetric tensors, the eigenvalues are real, and the eigenvectors are mutually orthogonal. The eigenvector corresponding to the larger eigenvalue is called major eigenvector. Analogously, the eigenvector corresponding to the smaller eigenvalue is denoted as minor eigenvector. If both eigenvalues are positive, the tensor is called positive definite. Examples for positive definite tensor fields are diffusion tensor fields. Stress and strain tensor fields are in general not positive definite.
Integrating the eigenvector fields results in two orthogonal families of continuous curves. These curves are called *major* (red) and *minor* (blue) tensorlines according to the eigenvector field integrated. These tensorlines are used to bound the cells of the segmentation.

3.2 Tensor Field Topology

Similar to vector fields the structure of eigenvector fields is represented by their field topology. It defines a skeleton consisting of distinguished points (degenerate points), and connecting edges (separatrices). In this work, the topology of both eigenvector fields is considered as one integral tensor field topology. In the following, we shortly resume the basics of tensor field topology, concentrating on the aspects that we need later on. A more detailed discussion on this topic is given in [5, 17, 20].

Degenerate points – definition At most points in a tensor field, both eigenvectors are defined uniquely; each assigned to one eigenvalue. However, this is not the case for points with identical eigenvalues, that is $\lambda = \mu$. These points are called *degenerate* or *isotropic points.* This means the tensor is proportional to the identity matrix and all vectors are eigenvectors. They are the only location where tensorlines of the same color can intersect. Mostly, degenerate points appear as isolated points but also degenerate features of higher order are possible. These are degenerate lines and triangles. Degenerate points in tensor fields are equivalent to critical points in vector fields. However, due to orientation indeterminacy of tensorlines, these points exhibit structures that are different from the structures seen in vector and scalar field topologies, respectively.

Degenerate point – classification The field behavior in the vicinity of degenerated entities is characterized by a number of specific sectors. These sectors are separated by distinguished tensorlines which enter the degenerate point radially. The following behavior is possible, see Figure 2:

- A *hyperbolic sector* is bounded by one red and one blue radial line: Tensorlines in this sector approach, sweep past the degenerate point and leave the sector through one bounding radial line.
- A *parabolic or radial sector* is bounded by two radial lines of the same color: In this sector tensorlines of this color, start from the degenerate point and then diverge. Tensorlines of the opposite color enter and leave the sector through the bounding lines.
- An *elliptic sector* is bounded by one red and one blue radial line: Tensorlines in this sector start from the degenerate point, and leave the sector through one of the bounding lines.

A criterion to classify the sectors is the rotation angle of the eigenvectors $\Delta\alpha$, in comparison to the opening angle of a sector $\Delta\Theta$, for more details we refer to [17, 18].

$$\Delta\alpha = \begin{cases} \Delta\Theta & \text{radial, concentric} \\ \Delta\Theta - \pi/2 & \text{hyperbolic} \\ \Delta\Theta + \pi/2 & \text{elliptic} \end{cases} \tag{1}$$

Separatrices Radial tensorlines bounding hyperbolic sectors are called *separatrices.* They constitute the edges of the topological graph. The graph defined by the two eigenvector field builds the basis for the following segmentation algorithm.

3.3 Interpolation

Usually, tensor datasets represent a discretized tensor field, given on uniform or non-uniform grids which we store on a triangular mesh. For the extraction of topology and the integration of tensorlines, a linear eigenvector and eigenvalue interpolation is used [10]. The insertion of new vertices in degenerate points makes sure that a consistent eigenvector interpolation is

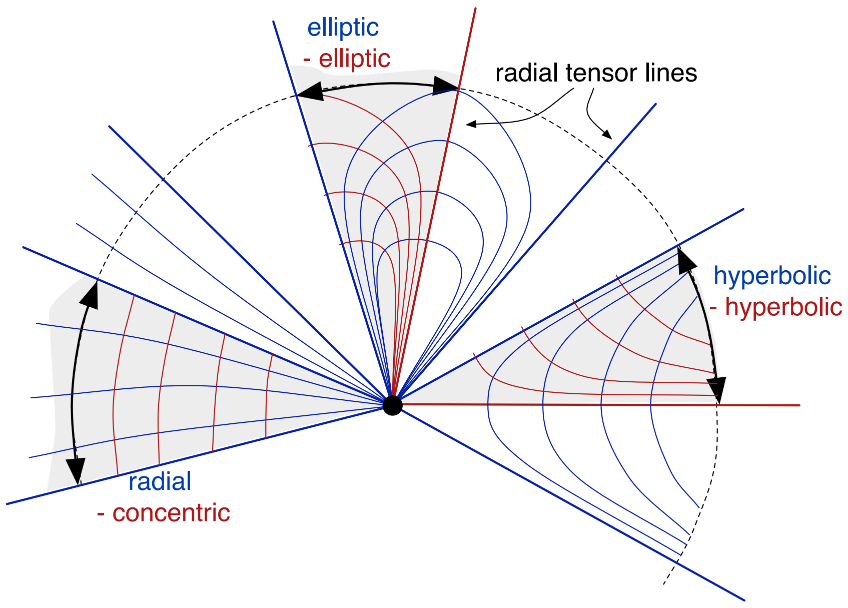

Figure 2 The neighborhood of a degenerate point is characterized by a number of sectors with specific behavior.

possible inside each triangle. The eigenvectors at these points are set to zero. The problem of eigenvector orientation is resolved by introducing edge labels, which encode the relative orientation of the calculated eigenvectors **v** and **w** in adjacent vertices. After computing these edge labels once, simple vector interpolation can be performed inside the triangles. The additional vertices together with re-triangulation lead to an increased number of degenerate entities of higher dimensionality when compared to linear interpolation of tensor components. This interpolation method has been chosen for performance reasons. It minimizes the number of eigenvector computations and makes an exact integration of the tensorlines possible. The interpolation can be easily replaced with any other consistent tensor interpolation, leading to slightly different results.

4 Initial Segmentation

In this section, we describe the steps leading to the initial segmentation of tensor fields using the topological skeleton. The degenerate points and intersections of the separatrices in both major and minor eigenvector fields define the vertices of the cells and the separatrices form the edges.

4.1 Topology Extraction

We extract the topology of the tensorfield using linear interpolation of eigenvector fields, given in Section 3.3. Alternatively, one can use other interpolation schema, e.g. [19]. Here, we summarize the steps taken for the topology extraction, which is described in detail in [17].

Location of degenerate points – The product of the edge labels, introduced for the interpolation, offers a simple criterion to detect triangles that contain degenerate points. The location of an interior degenerate point only depends on the eigenvector directions in the vertices

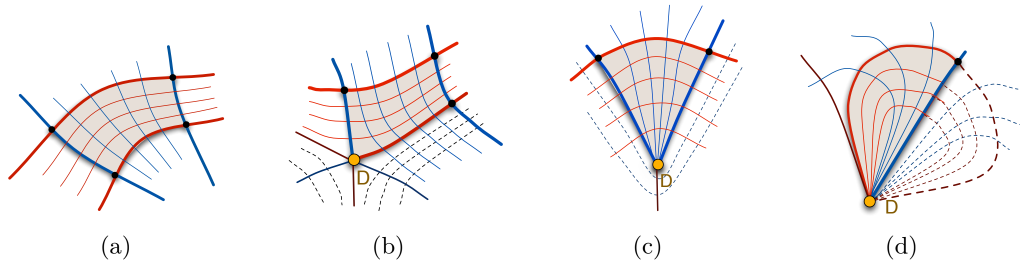

Figure 3 Cells defined by the topological skeleton: (a) regular cell without any degenerate points, (b) hyperbolic sector, (c) parabolic sector, and (d) elliptic sector.

$P_i, i = 1, 2, 3$ of the triangle. It is defined as the intersection of connections of the vertices and their so-called opposite points O_i. These are points on triangle edges e_i opposite to P_i, in which the eigenvector directions $\mathbf{v}_{oi}$ and $\mathbf{w}_{oi}$ are orthogonal to the eigenvector directions $\mathbf{v}_i$ and $\mathbf{w}_i$.

Determination and classification of radial directions – The neighborhood of a degenerate point is characterized by segments separated by radial tensorlines. For linear eigenvector interpolation, the radial tensorlines are straight lines defined by their intersection with the edges of the triangle. For the skeleton computation, only the radial lines which are boundaries of hyperbolic sectors are relevant. The classification is computed using Equation 1.

Non-isolated degenerate points – Degenerate entities like degenerate lines, polylines and triangle, can be treated similar to isolated degenerate points. The eigenvector field inside triangles adjacent to degenerate lines or triangles is constant. Thus, from a structural point of view, it is enough to consider the vertices of the polylines line as degenerate points. The sectors can be classified using the same angle criteria as for isolated degenerate points. While these configurations are rather rare in tensor fields for the component-based interpolation, they appear frequently for the eigenvector-based interpolation.

Separatrix computation and termination conditions – To complete the topological skeleton, relevant radial directions are integrated. The tensorline evaluation is done triangle-wise using 4th order Runge-Kutta. Eigenvector-based interpolation also allows an exact integration [16]. Following termination conditions are implemented to obtain a cleaner skeleton: (a) The separatrix leaves the domain. (b) The separatrix gets close to a degenerate point, line or triangle. (c) The separatrix describes a circle or spiral and passes itself closely in parallel integration direction. Separatrices are stored as polylines.

4.2 Cell Generation

After computing the topological skeletons for the major and minor eigenvector fields the intersections of the red and blue separatrices define the cells of the pre-segmentation. To increase the efficiency of these computations, every triangle keeps track of all separatrices passing through it. Thus, only the triangles that contain at least one red and one blue line are considered to compute the intersections. The vertices of the resulting curvilinear cells are either red-blue intersection points, degenerate points, or intersections of tensorlines with the boundary. They exhibit one of the following basic structures, see Figure 3:

1. Cells without a degenerate point are quadrangular with two red and two blue tensor lines

as boundary, in an alternating order. All red tensor lines passing through this segment enter at one blue boundary and leave the cell at the opposite boundary. All intersection angles are orthogonal.

2. Cells with one degenerate vertex lying in a hyperbolic sector are quadrangular. The angle at the degenerate point is in general not orthogonal.
3. Cells having a degenerate point in one vertex, lying in a parabolic segment, degenerate to a triangular shape.
4. In elliptic sectors, cells with either two or three vertices are possible.
5. Cells containing degenerate lines as edges can exhibit all kinds of complicated structures.

The edges of the cell are segments of the separatrices and hence are represented as polylines. The edges are ordered in counterclockwise orientation of the cell, and stored in a doubly-linked list, for efficiency in finding neighbors to the cell and adjacent edges in a cell. Each edge is represented using a half-edge data structure.

4.2.1 Half-edge Data Structure

A half-edge data structure [1, 4] is an edge-centered data-structure that maintains spatial information of vertices, edges and cells. Each edge is shared by two cells. An edge can also be considered as two opposite directed half-edges, called twins. Each half-edge stores its start point, the end point of a half-edge however is determined indirectly by referencing to the start point of the twin. The prime advantage of using this data structure is that a half-edge and its corresponding cell share a one-to-one relationship. Consequently, neighbor-searches and an iteration through the cells become very efficient.

Half-edge twins always belong to the same separatrix, except in the cases when the edges are part of either boundaries or degenerate lines. As separatrices are represented by polylines, the half-edge data structure is represented by polylines. Our implementation using CGAL [1] additionally has to support irregularities in the cell layout, namely T-junctions or hanging nodes, where twins have an n:m relation, such that they share common points of the separatrix polyline but do not share the same start and end points, see Figure 4(a). To resolve the issue of continuities in irregularities, our half-edge data-structure is modified as follows: (a) We store pointers to points representing the current edge; (b) Two sets of twins are supported for each edge - (i) *a geometric twin*: a single edge to identify the geometrical limits of the edge and (ii) *neighboring twins*: an array of twins to identify all neighboring cells in case of hanging nodes. The geometric twin of an edge is the flipped image of the edge with respect to its starting and end points, which would ideally be the twin but necessarily need not exist in the topological skeleton. The neighboring twins of an edge is the segmented set of the first twin, which are the edges that actually exist in the skeleton. In the absence of hanging nodes, the second set is a singleton set of the first twin.

4.2.2 Creating Cells from Topological Skeleton

The actual cell creation process involves physically creating the half-edges from the topological skeleton, and using them to build the curvilinear cells. Starting with a single cell as a seed cell, its neighborhood is grown to find the entire set of cells. Convex cells can be found by a strict rotation angle criteria at the vertices in a counterclockwise orientation. The remaining non-convex cells in the vicinity of degenerate lines or triangles are found by implementing a greedy walk of finding consecutive half-edges that are not associated with any cells. Consecutive half-edges are all that have the current half-edge's end point as start point.

5 Adaptive Segmentation Workflow

The segmentation resulting from the topology already decomposes the domain in regions where the eigenvector fields have a qualitatively similar behavior, but it does not yet fulfil all our criteria for a good segmentation. To represent the entire tensor information also the scalar invariants based on the eigenvalues have to be considered. This is achieved by adaptively modifying the cells, characterized by a specified degree of similarity with respect to eigenvalue behavior.

The segmentation strategy on the initial cell structure builds on two basic operations:

- **Coarsening**: Cells that do not exhibit enough structural information on their own get merged with adjacent cells.
- **Subdivision**: Cells which exceed the defined criteria of similarity are subdivided by new tensorlines.

Due to divergence and convergence of tensor lines, adaptive segmentation inevitably causes occurrences of hanging nodes in the edges. To keep these irregularities to a minimum, a growing strategy where we continue to merge or subdivide on consecutive cells as long as possible and necessary, see Figures 4(b,c). Algorithms for coarsening and subdivision operations are described in Section 5.3.

To guide the modification process by eigenvalue behavior one or more scalar fields are derived from the initial tensor field, which directly render the eigenvalue behavior, see Section 5.1. The degree of similarity and the need of modification is represented by weight functions defined on the edges of the cells, see Section 5.2. These edge-weights evaluate the derived scalar field but also reflect geometric properties of the cells. Combined with data dependent thresholds the edge-weights serve as decision basis whether cells have to be merged or subdivided. Further the edge-weights help to steer the modification by importance, as their values directly offer prioritisation to achieve a smooth segmentation. Using edge-weights for decision-making is an efficient choice as edges are one-dimensional structures, on which the weights are computed.

To extend the capabilities of the adaptive segmentation process, it is designed to have a high degree of flexibility to customize the workflow. For example using this segmentation as preprocessing step for glyph placement would have different demands than using it for texture mapping. We define the variables for customizing the workflow of the adaptive refinement as:

- **Operations:** The operations of the refinement, namely coarsening and subdivision, are the modules of the workflow. They can be repeated and the workflow customized by choosing the number and the order of implementation of the operations.
- **Control Parameters:** The control parameters of the refinement are the chosen edge-weights, the considered scalar fields and the thresholds defined by the demanded accuracy and resolution. The choice of these control parameters impact the priority queues used for the implementation of the operations.

The remainder of this section first presents the operations and control parameters of the approach. Then the basic workflow is demonstrated by calculating a segmentation of the tensor field. The segmentation follows accuracy towards tensor invariant similarity as wells as geometric criteria. Implemented as focus + context visualization, the refinement process is stopped for cells whose size falls below a value proportional to the resolution of the displayed domain. This provides an overview of the field, on demand the user can specify a focus

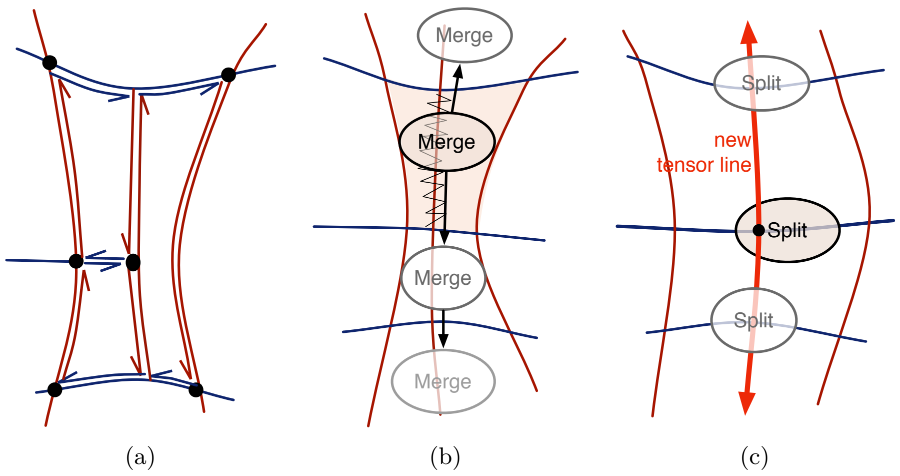

Figure 4 (a) T-junction or hanging node, where neighboring twins in half-edge structure have an n:m relation. Adaptive refinement operations: Recursive strategy for avoiding hanging nodes in (b) coarsening of cells, and (c) subdivision of cells started by insertion of new tensor line.

region to view further detail, see Section 5.5. The flexibility of the approach is finally shown by the extraction of degenerate regions, see Section 5.6.

5.1 Choice of Scalar Field

Several scalar fields can be considered as basis for the refinement depending on the specific application. We can either use both eigenvalue fields, an anisotropy value, the maximum shear stress (which is related to the anisotropy) or other tensor invariants.
Maximum shear stress S is defined as

$$S = |\lambda - \mu| \tag{2}$$

For the anisotropy we propose a generalized notion of fractional anisotropy FA^*, which can also be used for non positive definite tensors.

$$FA^* = \sqrt{\frac{(\lambda - \mu)^2}{\lambda^2 + \mu^2 + A^2}} \tag{3}$$

The positive constant A is added to the denominator, which eliminates the discontinuity close to zero for $\lambda, \mu \in IR$. This results in low anisotropy values for tensors with eigenvalues of different sign but small absolute value. The values of FA^* range from 0 to 1.

5.2 Edge-weight Definition and Thresholds

A set of pre-defined functions as weights assigned to the cell edges is provided. It consists of geometric measures representing the current cell size and shape as well as similarity measures for scalar fields derived from the tensor field. This set can be extended by user-defined functions. The dominant use of geometric edge-weights favours a more uniform segmentation, whereas the scalar field based weights lead to a higher adaptivity towards accuracy in eigenvalue similarity. In the following two weights of each class are proposed. These weights can be arbitrarily combined.

Let e be an edge of a cell consisting of k segments $(\mathbf{x}_i, \mathbf{x}_{i+1})$, by the virtue of being part of a polyline, where $\mathbf{x}_i$ is the position of the ith point on the edge. Further, let s be a scalar function defined along the edge and $s_i = s(\mathbf{x}_i)$.

- *Variance of scalar values $s(\mathbf{x}_i)$ on the edge $w_v(e)$:*
  $w_v(e) = \frac{\sum_{i=1}^{k}(s_{x_i}-\bar{s})^2\|\mathbf{x}_{i+1}-\mathbf{x}_i\|}{\sum_{i=1}^{k}\|\mathbf{x}_{i+1}-\mathbf{x}_i\|}$, where $\bar{s}$ is the mean of s along e.
- *Absolute difference of minimal and maximal scalar value along the edge $w_d(e) = abs(s_{min} - s_{max})$.*
- *Edge length $w_l(e) = \sum_{i=1}^{k}\|\mathbf{x}_{i+1} - \mathbf{x}_i\|$.*
- *Change of eigenvector direction along the edge $w_c(e)$:*
  $w_c(e) = \sum_{i=1}^{k}|\angle(\mathbf{v}_i, \mathbf{v}_{i+1})|$, v_i is the major eigenvector at position x_i.

We chose the proposed edge-weights to be as intuitive and universal as possible, independent of the various ranges that appear in different datasets. *Variance* is a commonly known statistical quantity and is a similarity measure which is robust to smaller perturbations of scalar values along an edge, such as noise. *Difference of minimal and maximal scalar value* in turn is strict towards any changes of scalar values along an edge and directly renders the absolute difference of scalar values appearing on an edge. *Edge length* can be used to adjust the size of the segmented cells to optimize perceivability by the user. The eigenvector directions are already well represented by the cell shape and tensor line boundaries, however if uniformity of the cells is required *change of eigenvector direction* represents the curvature of the cell boundaries and is therefore an appropriate measure.
In Section 5.5 we give a preset of thresholds for these weights, which are calculated as percentages of the given ranges in the field. These presets are universal and lead to stable results of good quality, which experiments with different datasets showed, see Section 6. However they can be intuitively strictened or loosened for different visualization purposes with immediate interpretation.

5.3 Refinement Operations

If not noted differently the refinement operations always respect the chosen thresholds towards the edge-weights. For example if a minimum edge length for the coarsening operation is specified - no edge subdivision is performed if one of the new edges would fall below the minimum edge length.

5.3.1 Coarsening

The main goal of the coarsening operation is to get rid of small cells that do not carry enough structural information on their own. Coarsening operation involves merges of cell pairs. Merging a pair of cells requires the merge of up to two pairs of edges and removal of the common edge of the cells. For this operation, we build an edge-weight based priority queue of pairs of cells that can be merged. We use queues to follow the FIFO (*first in, first out*) order, ascending or descending priority is fixed by minimum or maximum thresholds respectively.

Merge Prerequisites – Based on the geometry layout, two cells can only be merged if they share *a common edge* that can be deleted to join these cells. Technically, a common edge between two cells means that one of the cells has an edge whose geometric twin is an edge of the second cell. Edges containing hanging nodes cannot be common edges.

Priority Queue and Sorting – For the coarsening operation a priority queue of pairs of adjacent cells that can be merged is maintained. A multi-pass sort is performed based on the chosen edge-weights. Not only the smallest cells should be merged first but also for each cell two neighboring cells are candidates for merge (except for boundary cells). For the example of the segmentation workflow in Section 5.5 the priority queue is first sorted by minimum edge length of the edges involved and then by maximum edge length of the edges to be merged.

Algorithm

- Check adjacent cells for if they can be merged and sort these into the priority queue, based on the chosen edge-weight prioritization.
- While the priority queue is not empty, the pair with the highest priority is merged.
- Update the data structure by merging the appropriate edges of the pair cells, deleting the common edge and creating a new cell from the new edges.
- Update the priority queue with the new merged cell.
- If contiguous cell pairs in direction of the deleted common edge are to be merged move them on top of the queue to accomplish the recursive workflow. See Figure 4(b).

5.3.2 Subdivision

Single cells are subdivided by starting a new tensor line of opposite color on one of its edges that has to be subdivided. Similar to the coarsening operation an edge-weight based priority queue implemented as FIFO.

Start Point of Subdividing Tensor line – Two possibilities for the start point of the new tensor lines are provided. The first option favours the generation of equally sized cells, and starts the tensor line in the midpoint of the edge. The second option starts the tensor line between the extrema of the scalar values on the edge. This choice is more adapted to the data and guarantees to decrease the edge-weight when subdividing. There are no technical prerequisites to subdivide a cell.

Priority Functions and Sorting – Differently from the coarsening operation, a priority queue for edges is used rather than cells. Again the priority queue can be sorted according to multiple edge-weights. The growing strategy in subdividing consecutive cells is implemented by integrating a subdividing tensor line as long as possible and necessary, see termination conditions (a,b) in the algorithm below. No explicit prioritisation has to be done.

Algorithm

- Sort edges to be subdivided into priority queue, based on chosen edge-weights.
- While the priority queue is not empty, pop the top edge and start a subdividing tensor line of opposite color.
- Integrate tensor line until one of the following termination conditions is reached:
  (a) It intersects an edge, which is not in the priority queue and therefore should not be subdivided.
  (b) It intersects an edge and it's subdivision would generate edges violating fixed edge weight thresholds, for example minimum edge length.
  (c) It fulfils one of the termination conditions described in Section 4.1.
- Subdivide all cells corresponding to edges intersected by the new tensor line, as shown in Figure 4(c). Update data structure by subdividing intersected edges, generating new

edges along the tensor line, and finally generating new subdivided cells using the new edges.

– Update priority queue by deleting the original edges intersected by the new tensor line, and adding and sorting the newly generated edges if they are candidates for further subdivision.

5.4 Customized Workflow of Adaptive Refinement

The possibility to customize the workflow gives a high degree of flexibility in obtaining various analyses of the same dataset. Essentially the workflow consists of modules for operations, which are influenced by the control parameters and strategies adopted for implementation. Variations in the workflow are achieved by changing the number and order of the modules, by adjusting the thresholds used for each operation, and by deciding on the strategies to be used for the control flow of the modules. Strategies include the choice of appropriate edge-weights and scalar fields and choice of position of starting a new tensor line for subdivision of edges.

Table 1 Table to summarize the options when configuring for the segmentation process.

Basic operations	coarsening	subdivide
Edge-weight prioritisation	geometric	scalar field
Error measure for edge-weight definition	variance	max difference
Tensor line seeding	middle of edge	between max and min
Level of detail	resolution	accuracy

For a domain expert the flexibility of the approach ranges from using the presets with the scalar field of his choice over strictening or loosening thresholds to mixing and matching the implemented components to his needs. Developers can extend the basic set by implementing new elements, as e.g. edge-weights or tensorline seeding.

5.5 Workflow: Basic Segmentation

This workflow delivers a focus + context visualization, calculating an initial context segmentation which can be browsed in detail by selecting a focus region. The field is segmented in regions of similar tensor invariant behavior. We chose FA^* (see Section 5.1) as scalar field to render the eigenvalue characteristics. For all operations the same edge-weights and thresholds are used.

The default parameters are, if not differently noted:
(a) Geometric edge-weight steering resolution is the edge length w_l with minimum edge length threshold fixed to $\varphi_l = 1\%$ of the displayed domain range. By selecting a focus region the minimum edge length is automatically adjusted and the segmentation is refined displaying further details.
(b) Scalar field edge-weight steering accuracy is the absolute difference of minimum and maximum scalar value w_d, with its threshold set to 10 % of the scalar value range, which is for FA^* as scalar field φ_d= 0.1.
Resolution and accuracy are the basic level of detail parameters for focus + context visualizations, where the geometric edge-weight has higher priority than the accuracy edge-weight. This means for edge lengths smaller than the fixed minimum edge length a merge is performed even if the merged edge exceeds the given accuracy threshold.

The following operations composite the workflow

1. First coarsening: in merging as many similar cells as possible cleans up the pre-segmentation , especially very small cells are removed. The priority queue is first sorted by minimum edge length of the edges involved and then by maximum edge length of the edges to be merged. Merging of small cells with rather large cells favours the goal of a smooth segmentation.
2. Subdivision: the cells are refined to the pre-defined accuracy, unless this violates the resolution criterion. The tensor line seeding is between the extremal points. Experiments showed that the best strategy for a smooth segmentation is to do a 2-pass sort of the priority queue first based on maximum edge-length, the second pass based on maximum scalar edge weight.
3. Final coarsening: cells with highest similarity are merged. The priority queue is sorted first by minimum edge length of the edges involved and then in ascending order by a pre-calculated scalar edge weight w_d of the edges to be merged.

For a chosen focus step 2 and 3 are repeated with adjusted geometric edge weight, the accuracy edge weight remains. As the cells are given as explicit entities, cells exceeding the accuracy edge weight can be highlighted on demand.
This workflow and thresholds can be used in any tensor field segmentation as stable presets. Results for using *variance* as scalar field edge-weight are given in Section 6.

5.6 Workflow: Degenerate Regions

In the direct vicinity of degenerate entities tensors are almost isotropic and thus directional behavior is not strongly expressed. Such areas are collectively represented as *degenerate regions*, following the paradigm of the coarsening operation, that all cells are combined which do not exhibit enough structural information on their own. These degenerate regions are "grown" from the degenerated entities using two passes of the subdivision operation. FA^* is used as scalar field.

1. First subdivision: In the first pass only edges, called *degenerate edges*, emerging from a degenerate entity are considered for subdivision using a specific edge-weight, which is the maximum anisotropy occurring on the edge. The start point for the subdividing tensor line is the point on the edge with anisotropy below a fixed threshold $\varphi = 0.05$ and with maximum distance from the degenerate entity. prioritisation is done by the least anisotropy value of the start points.
2. Second subdivision: As it cannot be guaranteed that the subdividing tensor lines will intersect other degenerate edges at points with anisotropy below φ, a second subdivision pass is performed, where all edges participating in the intermediate degenerate regions from the first subdivision step are considered.

It should be noted that the generation of degenerate cells may induce subdivision of neighboring cells that may result in small not well shaped cells, which may not get merged in a later step, as seen in Figure 9. It is still of benefit to extract degenerate regions, as the weak expression of direction in such areas can result in numerical instabilities.

6 Results and Discussion

We tested our algorithm on three datasets from structural engineering, which are finite element simulations of forces acting upon solid blocks resulting in stress tensor data. These

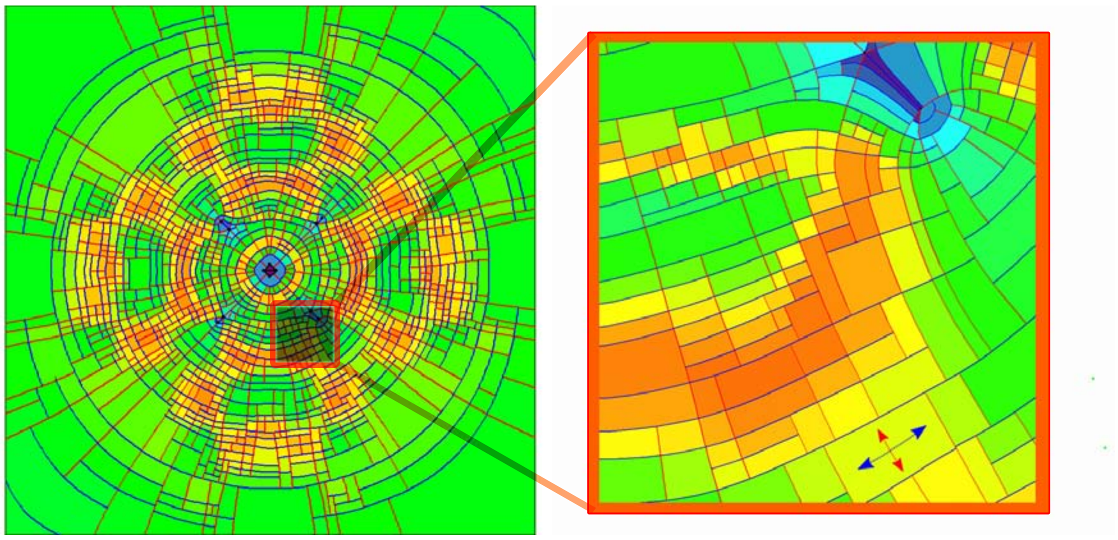

Figure 5 Focus + context visualization.

are simulations of one and two forces applied to the top of a solid block and of multiple forces applied to a notched block, the latter using hp-adaptive finite elements. In this section, we will refer to them as *one point load* (1PL), *two point load* (2PL), and *notched block* (NB), respectively.

Figure 5 shows the focus + context implementation proposed in Section 5.5 on the 1PL dataset. The right image shows the selected focus further refined with the resolution threshold, minimum edge length, adapted to 1% of the displayed domain range, the accuracy threshold remains. In this visualization the cells delineate regions of similar eigenvalue behavior, the color coding of the cells renders the relation of the eigenvalues. The two arrows in the lower right corner of the focus image indicate exemplary how the eigenvector behavior can be interpreted from the cell boundaries.

We conducted further analysis on thresholds, specific control parameters and strategies. To evaluate the quality of the overview segmentation we used the following methods:

- *Image representation of local error*: The error is sampled in an image of resolution $u \times v$ where each pixel (i, j) is mapped to a point (x, y) in the tensor field, and is assigned a color from a red-shaded colormap mapping to error value $err(i, j) = \|s(x, y) - \bar{s}_{cell(x,y)}\|$, where $cell(x, y)$ is the cell containing (x, y) and $\bar{s}_{cell(x,y)}$ is the average scalar value on the edges of cell $cell(x, y)$. Results are shown in Figure 8(a) and 7.
- *Average error:* While computing the image representation of the local error, we calculate the average error for the field as $\oslash Err = \frac{\sum_{i,j} err(i,j)}{u*v}$.
- *Number of cells needed for pre-defined quality:* We aim to have as few cells as possible in an adaptively segmented field, which makes the number of cells needed to achieve a segmentation of predefined quality an important criterion.

The first example examines level of detail according to accuracy, see Figure 6. In Figure 6(b) φ_d is twice as strict as for Figure 6(c). Using superimposition of the original scalar field on segmentation result of the one point load dataset, Figure 6(a) demonstrates that the segmentation and original scalar field match.

The second example focuses on resolution as level of detail. In Figure 7(b) the minimum

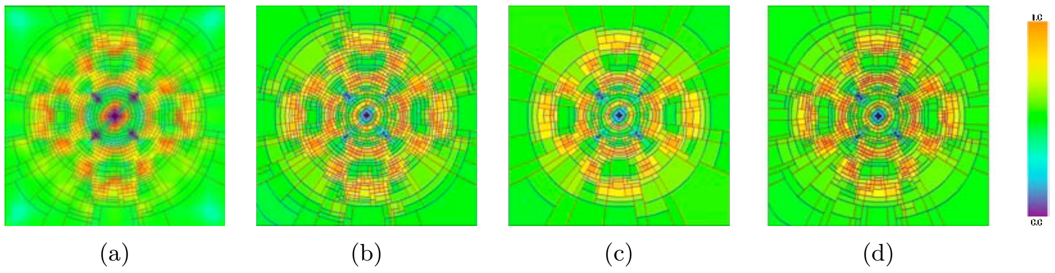

Figure 6 Adaptive segmentation of a slice in the one point load dataset. (a) Superimposition of scalar field on adaptive segmentation of tensor field. Choice of threshold for edge-weight φ_d of value (b) 0.1 and (c) 0.2 for customizing the workflow for segmentation. (d) Segmentation with edge-weight variation and $\varphi_v = 0.015$

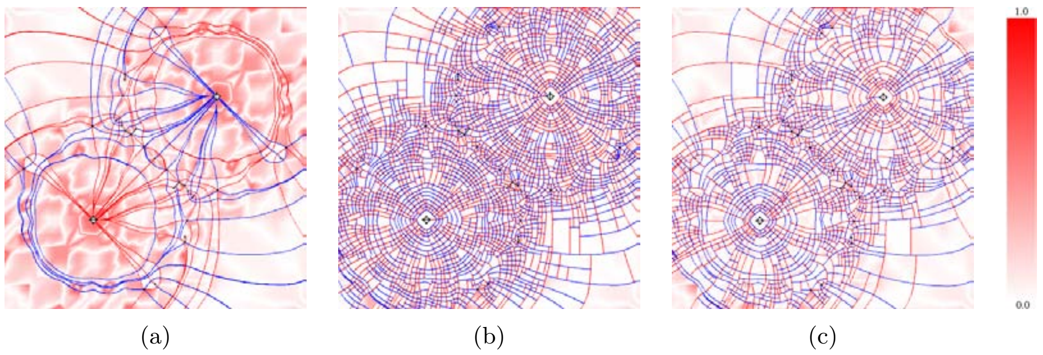

Figure 7 Image representation of error for a specific slice of two point load dataset. (a) Initial segmentation of tensor field. (b) Adaptive segmentation using w_d as edge-weight. (c) Adaptive segmentation using double threshold for minimum edge length.

edge length φ_l is twice as strict as for Figure 7(c). All images are superimpositions of error images and the segmented cell boundaries. Figure 7(a) shows the pre-segmentation, and 7(b,c) the results for the different levels of detail.

As differences in images can be hard to perceive we display results of further evalutation in tables, associated figures are denoted. The first block in Table 2 is used as reference for the subsequent evaluation, it lists the values for the overview segmentation of Section 5.5.

Test 1: Comparison of edge-weights, variance w_v vs. absolute difference of extremal values of scalar field w_d:

As threshold for w_v 0.1% of the scalar value range is used, given by $\varphi_v = 0.001$. The middle block in table 2 and Figure 6(d) demonstrate that w_v leads to fewer cells at a higher mean error, compared to w_d used in the basic workflow. Variance is a criterion that regards mean values and ignores smaller variations along an edge. Thus fewer cells have to be subdivided, see Figure 6(d). The smoothing effect of the variance-based edge-weight can be extensively used to achieve desired results. w_d is an edge-weight that is rather strict and regards any change in the scalar field along the edge, which results in more cells, but higher accuracy.

Test 2: Comparison of strategies in choice of start point of tensor lines for subdivision operation: midpoint of an edge vs. point between extrema of the scalar field along an edge. Using the midpoint of an edge is a simple, straightforward technique, which leads to a slightly

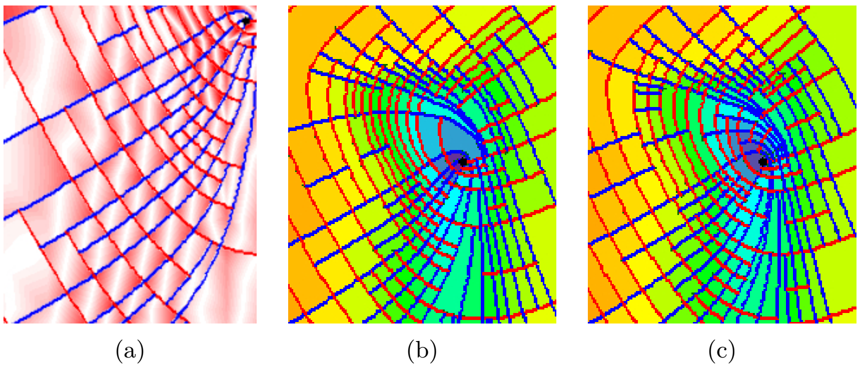

Figure 8 Close up of adaptive segmentation of a slice in the notched block dataset: (a) Image representation of error shows how the edge-weights only reflect the behavior of the scalar field on the edges of the cell but not in its interior. Results of the choice of start point of tensor lines for the subdivision operation, at (b) the midpoint of edge and (c) the midpoint of extrema of scalar value along edge.

Table 2 Results for tests on control parameters and strategies 5.5.

Data	**2PL**	**1PL**	**NB**
Basic workflow:			
#Cells	2830	1424	420
$\oslash$*Err*	4.67e-2	3.38e-2	4.02e-2
Ref. Figure	Fig. 7(b)	Fig. 6(b)	Fig. 8(a,c)
Accuracy edge weight *variance*:			
#Cells	2497	1129	285
$\oslash$*Err*	5.039e-2	3.64e-2	4.93e-2
Ref. Figure		Fig. 6(d)	
Start point for subdividing tensor line in the middle:			
#Cells	2870	1414	402
$\oslash$*Err*	4.75e-2	3.38e-2	4.02e-2
Ref. Figure			Fig. 8(b)

higher mean error, as shown in Table 2 last block. Starting the subdividing tensor lines between the minimum and maximum scalar values guarantees decrease in the edge-weights. This leads to qualitatively higher subdivisions as shown in Figure 8(b,c).

7 Conclusions

We have shown that the presented segmentation approach is able to generate segmentations aligned to the tensor field with low error measures. The weights and strategies can be chosen dependent on the demands of the specific application and allow a wide variety of data representation. The results shown in this paper are based on the anisotropy of the tensor field, and we can extend our work to other scalar fields used individually or as a combination of several fields. The main challenge for a segmentation based on tensor lines is the fact that feature lines resulting from the scalar field are in general not aligned with the eigenvector

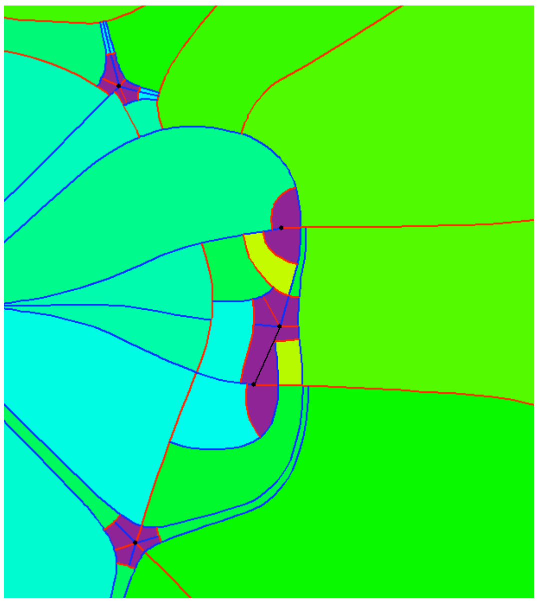

Figure 9 Close up: degenerate regions marked in violet.

field and thus they can only be approximated by a step function (see Figure 8(a)).
The strength of the application lies in its flexibility - restricting the modification process to geometric edge-weights delivers a uniform segmentation, whereas steering it by accuracy edge-weights generates a highly adaptive segmentation. There is a variety of options to customize the segmentation and use it for example for tensorline seeding or as preprocessing step for glyph placement or texture mapping.
During the design of the algorithm at many points decisions were made to balance its efficiency and accuracy. One example is the choice to define weights only on basis of the scalar field along cell edges, which is very efficient. But it cannot be guaranteed that the error contained in a cell is always well represented by the error along its one-dimensional edges. An integrated analysis of all cell edges, which leads to use of cell-weights as opposed to edge-weights is one of the interesting options for the future.

References

1 Cgal, Computational Geometry Algorithms Library. http://www.cgal.org.
2 Pierre Alliez, David Cohen-Steiner, Oliver Devillers, Bruno Levy, and Mathieu Desbrun. Anisotropic polygonal remeshing. *SIGGRAPH 03*, 22(3):485–493, 2003.
3 A. Bhalerao and C.-F. Westin. Tensor splats: Visualising tensor fields by texture mapped volume rendering. In *6th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 294–901, 2003.
4 M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry, Algorithms and Applications.* Springer, 2nd edition, 1998.
5 Thierry Delmarcelle. *The Visualization of Second-order Tensor Fields.* PhD thesis, Stanford University, 1994.
6 Thierry Delmarcelle and Lambertus Hesselink. Visualization of second order tensor fields and matrix data. *IEEE Computer Graphics & Applications*, pages 25–33, 1993.
7 Louis Feng, Ingrid Hotz, Bernd Hamann, and Kenneth Joy. Anisotropic noise samples. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):342–354, 2008.

8 Robert B. Haber. Visualization techiques for engineering mechanics. *Comp. Systems in Engineering*, 1(1):37–50, 1990.

9 Ingrid Hotz, Louis Feng, Hans Hagen, Bernd Hamann, Boris Jeremic, and Kenneth I. Joy. Physically based methods for tensor field visualization. In *VIS '04: Proceedings of IEEE Visualization*, pages 123–130, 2004.

10 Ingrid Hotz, Jaya Sreevalsan-Nair, and Bernd Hamann. Tensor field reconstruction based on eigenvector and eigenvalue interpolation. In Hans Hagen, editor, *Scientific Visualization: Challenges for the Future*, 2008.

11 B. Jeremic, Gerik Scheuermann, Jan Frey, Zhaohui Yang, Bernd Hamann, Kenneth I. Joy, and Hans Hagen. Tensor visualization in computational geomechanics. *Int. Journal for Numerical and Analytical Methods in Geomechanics*, 26:925–944, 2002.

12 Gordon Kindlmann. Superquadric tensor glyphs. In *Proceeding of The Joint Eurographics - IEEE TCVG Symposium on Visualization*, pages 147–154, 2004.

13 Gordon Kindlmann and Carl-Fredrik Westin. Diffusion tensor visualization with glyph packing. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1329–1336, 2006.

14 Yingmei Lavin, Rajesh Batra, Lambertus Hesselink, and Yuval Levy. The topology of symmetric tensor fields. *AIAA 13th Computational Fluid Dynamics Conference,*, page 2084, 1997.

15 Guillaume Lavoue, Florent Dupont, and Atilla Baskurt. A new cad mesh segmentation method, based on curvature tensor analysis. *Computer-Aided Design*, 37(10):975–987, 2005.

16 Gregory M. Nielson and Il-Hong Jung. Tools for computing tangent curves for linearly varying vector fields over tetrahedral domains. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):360–372, 1999.

17 Jaya Sreevalsan-Nair, Cornelia Auer, Bernd Hamann, and Ingrid Hotz. Eigenvector-based interpolation and segmentation of 2d tensor fields. In *submitted to TopoInVis 09*.

18 Xavier Tricoche. *Vector and Tensor Field Topology Simplification, Tracking and Visualization*. PhD thesis, University of Kaiserslautern, 2002.

19 Xavier Tricoche, Gerik Scheuermann, Hans Hagen, and Stefan Clauss. Vector and tensor field topology simplification on irregular grids. In *VisSym '01: Proceedings of the symposium on Data Visualization 2001*, pages 107–116, 2001.

20 Xavier Tricoche, X. Zheng, and Alex Pang. Visualizing the topology of second order, time-varying two-dimensional tensor fields. In *Visualization and Image Processing of Tensor Fields*, pages 225–240, 2005.

21 Zhizhou Wang. *Diffusion Tensor Field Restoration and Segmentation*. PhD thesis, University of Florida, 2004.

22 Joachim Weickart and Hans Hagen, editors. *Visualization and Processing of Tensor Fields*. Mathematics and Visualization. Springer, 2006.

23 Y. Weldeselassie and G. Hamarneh. Dt-mri segmentation using graph cuts. In *Medical Imaging 2007: Image Processing. SPIE*, 2007.

24 Xiaoqiang Zheng and Alex Pang. Hyperlic. In *Proceedings of IEEE Visualization 2003*, pages 249–256, 2003.

25 Xiaoqiang Zheng and Alex Pang. Topological lines in 3d tensor fields. In *Proceedings of IEEE Visualization 2004*, 2004.

26 Leonid Zhukov, K. Museth, D. Breen, Ross Whitaker, and Alan Barr. Level set modeling and segmentation of dt-mri brain data. *Journal Electronic Imaging*, 12(1):125–133, 2003.

27 Ulas Ziyan, David Tuch, and Carl-Fredrik Westin. Segmentation of thalamic nuclei from dti using spectral clustering. In *Ninth International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI'06)*, pages 807–814, 2006.