Eurographics Conference on Visualization (EuroVis) 2018
J. Heer, H. Leitte, and T. Ropinski
(Guest Editors)

Volume 37 (2018), Number 3

Core Lines in 3D Second-Order Tensor Fields

T. Oster, C. Rössl and H. Theisel

University of Magdeburg, Germany

Abstract

Vortices are important features in vector fields that show a swirling behavior around a common core. The concept of a vortex core line describes the center of this swirling behavior. In this work, we examine the extension of this concept to 3D second-order tensor fields. Here, a behavior similar to vortices in vector fields can be observed for trajectories of the eigenvectors. Vortex core lines in vector fields were defined by Sujudi and Haimes to be the locations where stream lines are parallel to an eigenvector of the Jacobian. We show that a similar criterion applied to the eigenvector trajectories of a tensor field yields structurally stable lines that we call tensor core lines*. We provide a formal definition of these structures and examine their mathematical properties. We also present a numerical algorithm for extracting tensor core lines in piecewise linear tensor fields. We find all intersections of tensor core lines with the faces of a dataset using a simple and robust root finding algorithm. Applying this algorithm to tensor fields obtained from structural mechanics simulations shows that it is able to effectively detect and visualize regions of rotational or hyperbolic behavior of eigenvector trajectories.*

CCS Concepts

•***Human-centered computing*** → ***Scientific visualization;***

1. Introduction

Feature extraction is one of the most successful standard approaches in Scientific Visualization. Features describe important and relevant properties of a field. Their extraction and representation promises an effective visualization of even complex data sets. For scalar and vector fields, a variety of features have been proposed. The most prominent ones for vector fields are topological features and vortices. For the extraction of vortex core lines, a standard approach is the one by Sujudi and Haimes [SH95]. It finds all locations in a piecewise linear vector field where the Jacobian has complex eigenvalues and the streamlines are parallel to its single real eigenvector. It is known that these lines can be characterized in terms of the parallel vectors (PV) operator [PR99], which has also been used to extract ridge and valley lines, and separation and attachment lines. Locations where the flow is parallel to an eigenvector of a Jacobian with three real eigenvalues also form stable lines. These hyperbolic trajectories are the centers of simultaneous converging and diverging behavior of the vector field and can be used to extract Lagrangian coherent structures [MSE13, MBES16]. Roth and Peikert [RP98] showed that vortex core lines can equivalently be described as locations where the curvature of streamlines locally vanishes.

In this work, we examine three-dimensional, second-order tensor fields, i.e., functions that map a matrix in $\mathbb{R}^{3\times3}$ to each location in a three-dimensional domain. Tensor fields arise from a variety of applications in physics. Examples are stress and strain-, deformation-, and diffusion tensors. In second-order tensor fields, the analog of stream lines are eigenvector trajectories. These are lines that are tangential to an eigenvector of the tensor field everywhere along their path. Eigenvector trajectories can show a behavior similar to vortices in vector fields. For example, solid objects subject to a torque show swirling eigenvector trajectories of the stress tensor field (see Figure 1). A number of topological visualization methods have been proposed for second-order tensor fields. However, there are no approaches to extract core lines of tensor fields similar to

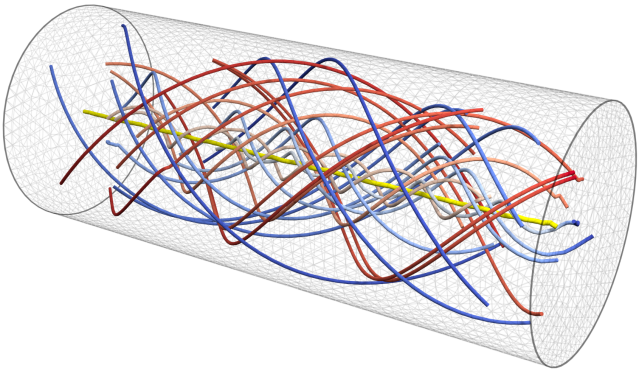

Figure 1: *Eigenvector trajectories in a stress tensor field induced by applying a torque to a cylindrical shaft. Trajectories of both major (blue) and minor (red) eigenvectors show a swirling behavior around a common core line (yellow).*

Sujudi/Haimes and the PV operator for vector fields. Such an approach would be a valuable tool for quickly identifying regions with swirling behavior of eigenvector trajectories, such as induced by torque, that would otherwise be tedious to find. In this paper, we introduce the concept of *tensor core lines* as all locations in the 3D domain of a tensor field where at least one eigenvector trajectory has a vanishing curvature. Considering the observation made by Roth and Peikert [RP98], this is the direct extension of Sujudi/Haimes to tensor fields. In particular, we make the following contributions:

- We give a rigorous definition of tensor core lines and show that the definition gives indeed structurally stable line structures.
- We provide a numerical algorithm for the extraction of tensor core lines in piecewise linear tensor fields.
- We introduce a filter criterion based on numerical stability to separate significant and insignificant tensor core lines.
- We show tensor core lines in mechanical stress tensor fields, interpret them and compare them with degenerate lines where two eigenvalues are equal.

Relation of Tensor Core Lines to the Parallel Vectors Operator

At first glance, tensor core lines seem to be a straightforward extension of the classical PV operator: given a tensor field, we consider all eigenvector fields as vector fields and apply the PV operator to them. Such a naive approach cannot give well-defined and stable results for the following reasons:

- Undefined length and orientation of eigenvectors:
  An eigenvector of a matrix is in fact not a vector but a linear subspace. To express the field of eigenvectors as a vector field, heuristic assumptions about the length and orientation of the vectors are necessary. Applying such assumptions globally can not always give results that are free of discontinuities.
- Existence of multiple eigenvectors:
  Regions with three real eigenvectors require a decision on which of them to use for the PV operator – a decision that is particularly non-unique in near-isotropic regions (i.e., where the difference between two real eigenvalues is small).
- Discontinuities in eigenvectors:
  A small change of a tensor does not necessarily result in a small change of the eigenvectors. In fact, in near-isotropic regions, a small change of the tensor may result in a very large change of the eigenvector. Moreover, in regions of transition between real and imaginary eigenvalues (i.e., in neighborhoods containing both tensors with all real eigenvalues and tensors with complex eigenvalues), a small change of the tensor can result in a sudden appearance or disappearance of real eigenvectors.

Some of the problems mentioned above could be tackled by local heuristics (for instance, the orientations of the eigenvectors in a local neighborhood could be chosen as consistent as possible). However, especially the last point shows that eigenvectors show a fundamentally different behavior than normal vector fields for which the PV operator is designed. New algorithms for the extraction of tensor core lines are therefore necessary.

2. Related Work

The basis of this work is the extractor for centers of swirling flow in vector fields first described by Sujudi and Haimes [SH95]. In their original paper, they examine the Jacobian $\nabla\mathbf{v}$ of a vector field $\mathbf{v}$. They search for vortex core lines only in areas where $\nabla\mathbf{v}$ has two complex eigenvalues, i.e., where the flow shows locally swirling behavior. The location of a vortex core line is then defined as the set of points where the velocity is parallel to the single real eigenvector. These are the locations where the swirling in the orthogonal plane vanishes and only a motion along the eigenvector direction remains. Note that vortex core lines defined in this manner are generally not stream lines of the vector field. When applying this method to a piecewise linear vector field, the result is a set of straight line segments. Since the derivative $\nabla\mathbf{v}$ is constant within each cell, these segments do not generally form closed lines. Single line segments may appear due to noise, but if multiple line segments form a visually coherent line, it is a strong indicator for vortical behavior in the flow. The advantage of this vortex core line extractor is its inherent locality, which makes it well-suited for parallelization and avoids expensive line integration.

Parallel Vectors

Peikert and Roth [PR99] showed that the vortex core lines described by Sujudi and Haimes are locations where the acceleration $\nabla\mathbf{v}\cdot\mathbf{v}$ is parallel to the vector field, and identified this as one application of a concept they called the *parallel vectors operator*. This concept had been applied in a lot of other contexts before, such as ridge detection in scalar fields [Har83] and extraction of attachment/separation lines in flows [KHL99].

An overview of a number of algorithms for computing parallel vectors can be found in Martin Roth's PhD thesis [Rot00]. A common approach is to first determine intersection points of the PV lines with the cell faces of a data set. The resulting points are then connected to lines. If there are exactly two intersections with the faces of a cell, they can be trivially connected with a line. In case of a higher number of intersections, different heuristics have been employed, the simplest one being to subdivide the cell until there are again only two intersections.

Another class of algorithms employs some form of line tracing to follow a parallel vector line starting from a seed point that has been found by one of the aforementioned face intersection methods. Banks and Singer [BS95] and Miura and Kida [MK97] employed a predictor-corrector scheme, following the lines in small steps and minimizing the angle between the vector fields in an orthogonal plane after each step. Sukharev et al. [SZP06] presented a mix between both approaches by first finding intersection points on a (potentially coarser) grid, and then following the analytical tangent of the PV line to connect them. A similar approach is taken by Theisel et al. [TS03], where the PV line is reformulated as a stream line in a *feature flow field* that is derived from the original vector fields. Given a starting point on the PV line, it can be traced like a stream line with any standard ODE solver. The *PVsolve* framework introduced by van Gelder and Pang [vGP09] is a generalized approach for finding PV lines in vector fields of different dimensionality. Their approach views the tracing of PV lines as a root-finding problem

and presents an algorithm that does not suffer from accumulating errors. Weinkauf et al. [WTvGP10] presented stable feature flow fields as an extension to feature flow fields that similarly eliminates accumulating errors.

Most PV algorithms assume they operate on piecewise linear or piecewise bi-/trilinear data in two- or three-dimensional space. However, there have been a number of publications dealing with higher-order data or using higher-order methods in some form. Roth and Peikert [RP98] introduced a method for finding strongly curved vortex core lines by using higher-order derivatives of the flow field. Bauer and Peikert [BP02] used a scale-space technique to filter out small-scale structures and irrelevant large-scale ones when computing vortex cores. Pagot et al. [POS*11] presented an algorithm for extracting PV lines from data represented by higher-order basis functions. Due to its generality, the already mentioned *PVsolve* framework [vGP09] allows for extracting parallel vector surfaces in time-dependent vector fields. Similarly, Theisel et al. [TSW*05] extracted PV surfaces in unsteady flows and applied it to vortex core line tracking. All these methods work on vector fields, while our approach deals with second-order tensor fields.

Tensor Field Visualization

Tensor fields (of second order) occur in a variety of different scientific contexts. Some examples are stress and strain tensors in mechanical engineering applications, and diffusion tensors occuring in diffusion tensor imaging (DTI), a special magnetic resonance imaging (MRI) modality used to visualize fiber tracts, e.g., in the human brain. Tensor field visualization methods can be roughly classified into three different categories: glyph-based, line-/surface-based and topology-based.

Glyph-based methods for tensor field visualization place small geometric objects in space to represent certain characteristics of the local tensor. Diffusion tensors, which are symmetric and positive definite, have been visualized by Kindlmann [Kin04] using superquadric glyphs aligned with the eigenvector directions. By explicitly encoding eigenvalue signs in geometry, Schultz and Kindlmann [SK10] extended this to indefinite tensors. Gerrits et al. [GRT17] additionally incorporated complex eigenvalue information into the glyph design to visualize general second-order tensors in 2D and 3D. A specialized glyph for stress and strain tensors was introduced by Hashash et al. [HYW*03]. Kindlmann and Westin [KW06] addressed the problem of clutter by optimizing glyph placement to increase information density while minimizing occlusion. Glyphs for comparative visualization of two different diffusion tensors were used on medical data by Zhang et al. [ZSL*16].

A simple extension of streamlines for visualizing symmetric tensor fields in 3D are hyperstreamlines, first introduced by Delmarcelle and Hesselink [DH93]. These hyperstreamlines follow one of the eigenvectors of the tensor field, while their cross-section is a cross shape or ellipse aligned with the other two eigenvectors and scaled by the corresponding eigenvalues. The problem of undefined integration direction in near-isotropic areas was adressed by Weinstein et al. with a new concept called tensorlines [WKL99]. These lines behave like hyperstreamlines, but try to preserve direction in areas of equal eigenvalues by incorporating local context information. As an extension to this concept, Jeremić et al. introduced hyperstream-surfaces [JSF*02], which are formed analogous to hyperstreamlines by using a line instead of a point as the seed structure.

The topology of symmetric 2D and 3D tensor fields was studied by Delmarcelle [DH94] and Hesselink [HLL97]. They characterized the topology of a tensor field by degenerate structures where two or more eigenvalues are equal. Zheng and Pang [ZP04] build on top of this work and provide numerical algorithms for extracting the topological skeleton in practice. Similar to the approach we take for extracting tensor core lines, their initial algorithm is based on finding the roots of a number of discriminant functions on the cell faces of a dataset using a bisection algorithm, and then connecting them to lines. Zheng et al. later introduced alternative approaches that are based on solving a system of equations on each face, and on tracing the degenerate lines using their analytical tangent [ZPP05]. As Schulz et al. [STS07] showed, these features are very sensitive to noise. A more stable approach suitable to noisy data such as obtained from DTI scans was therefore proposed by Tricoche et al. [TKW08]. The notion of tensor topology was recently extended to surfaces of neutral and traceless tensors by Palacios et al. [PYW*16], who also propose an improved algorithm for extracting degenerate lines. Assymetric tensors were studied in 2D by Zheng and Pang [ZP05] and in the context of flow visualization by Zhang et al. [ZYLL09].

3. Notation

Throughout the paper, we use the following notation:

$\mathbf{T}$	Matrix in $\mathbb{R}^{3\times3}$
$\mathbf{v}$	Column vector in $\mathbb{R}^3$
v_i	Components of a vector
$\begin{pmatrix}\mathbf{a} & \mathbf{b} & \mathbf{c}\end{pmatrix}$	Block matrix of multiple matrices/vectors
$\mathbf{T}_{x_1}, \mathbf{v}_{x_2}, \ldots$	Partial derivatives of a Matrix/vector in $x_1, x_2, \ldots$
∇	Nabla operator $\left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \frac{\partial}{\partial x_3}, \ldots\right)^{\mathrm{T}}$
$\nabla_{\mathbf{r}}\mathbf{F}$	Directional derivative of $\mathbf{F}$ along a vector $\mathbf{r}$
$\mathbf{v} \parallel \mathbf{u}$	The PV operator applied to $\mathbf{v}$ and $\mathbf{u}$

4. Tensor Core Lines

We define tensor core lines as the locations where eigenvector trajectories have a locally vanishing curvature. The intuition for this is similar to the intuition for the vortex core line extractor by Sujudi and Haimes: In a region where eigenvector trajectories show a swirling behavior, there must be a center of rotation, where the swirl vanishes. Similarly, an equivalent to hyperbolic trajectories [MSE13] for eigenvectors can be imagined. Note that like vortex core lines, tensor core lines are generally not eigenvector trajectories of the tensor field. In this section, we provide a formal definition of tensor core lines and examine their mathematical properties.

Let $\mathbf{T} : \mathbb{R}^3 \to \mathbb{R}^{3\times3}$ be a 3D second-order tensor field that may or may not be symmetric. We want to find locations where the direction of a real eigenvector of $\mathbf{T}$ does not change when moving along the eigenvector direction, i.e., where the curvature of an eigenvector trajectory vanishes. A vector $\mathbf{r} \in \mathbb{R}^3$ is an eigenvector of $\mathbf{T}$ if $\mathbf{T}\,\mathbf{r} = \lambda\mathbf{r}$ for eigenvalue $\lambda \in \mathbb{R}$ and $\mathbf{r} \neq \mathbf{0}$. To observe the change of eigenvector direction when moving along $\mathbf{r}$, we need to consider the

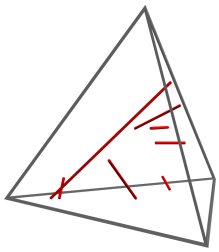

Figure 2: *Example of seven distinct core lines in a random linear tensor field.*

derivative of $\mathbf{T}$ in this direction. The directional derivative of $\mathbf{T}$ along $\mathbf{r}$ is the linear combination of three component-wise derivatives

$$\nabla_{\mathbf{r}}\mathbf{T}(\mathbf{x}) = \nabla\mathbf{T}(\mathbf{x})\,\mathbf{r} = \mathbf{T}_{x_1}(\mathbf{x})\,r_1 + \mathbf{T}_{x_2}(\mathbf{x})\,r_2 + \mathbf{T}_{x_3}(\mathbf{x})\,r_3$$

for $\mathbf{x} = (x_1, x_2, x_3)^{\mathrm{T}}$ and $\mathbf{r} = (r_1, r_2, r_3)^{\mathrm{T}}$. Given $\nabla_{\mathbf{r}}\mathbf{T}$ we can approximate the behavior of $\mathbf{T}$ along $\mathbf{r}$ as

$$\mathbf{T}(\mathbf{x} + h\,\mathbf{r}) = \mathbf{T}(\mathbf{x}) + h\,\nabla_{\mathbf{r}}\mathbf{T}(\mathbf{x}), \tag{1}$$

with $h \in \mathbb{R}$. For our zero-curvature requirement to be fulfilled, the eigenvector direction must not change when moving along $\mathbf{r}$, i.e.,

$$\mathbf{T}(\mathbf{x} + h\,\mathbf{r})\,\mathbf{r} = \mathbf{T}(\mathbf{x})\,\mathbf{r} + h\,\nabla_{\mathbf{r}}\mathbf{T}(\mathbf{x})\,\mathbf{r} = \mu\mathbf{r}, \tag{2}$$

for $\mu \in \mathbb{R}$. This means that $\mathbf{r}$ is only scaled by a factor μ. If we substitute $\mathbf{T}(\mathbf{x})\,\mathbf{r} = \lambda\mathbf{r}$, we get

$$\lambda\mathbf{r} + h\,\nabla_{\mathbf{r}}\mathbf{T}(\mathbf{x})\,\mathbf{r} = \mu\mathbf{r}$$

$$\nabla_{\mathbf{r}}\mathbf{T}(\mathbf{x})\,\mathbf{r} = \frac{\mu - \lambda}{h}\,\mathbf{r},$$

i.e., $\mathbf{r}$ is also an eigenvector of $\nabla_{\mathbf{r}}\mathbf{T}(\mathbf{x})$. With this, a tensor core line is an isolated line of positions $\mathbf{x}$ where

$$\lambda\mathbf{T}(\mathbf{x})\,\mathbf{r} = \mu\nabla_{\mathbf{r}}\mathbf{T}(\mathbf{x})\,\mathbf{r} = \mathbf{r},$$

for $\mathbf{r} \neq \mathbf{0}$ and $\lambda, \mu \in \mathbb{R}$. In terms of the PV operator, this equation has the form

$$\mathbf{r} \parallel \mathbf{T}\,\mathbf{r} \parallel \nabla_{\mathbf{r}}\mathbf{T}\,\mathbf{r}. \tag{3}$$

We now examine the mathematical properties of tensor core lines.

Lemma 1 In a C^1-continuous tensor field, tensor core lines are structurally stable line structures.

Here, structural stability refers to the property that adding noise slightly perturbs the structures but does not destroy them. To show Lemma 1, we consider a local approximation of a real eigenvector field in the neighborhood of a point as a normalized vector field. Then the fact that PV lines give line structures [PR99] shows the lemma. Note that although such a consideration of an eigenvector field as a normalized vector field is locally possible, it does not apply globally in a consistent way, and therefore does not provide a strategy to extract tensor core lines. We also mention that in real datasets, tensor cores may build surfaces or even parts of spatial structures. This is due to shape symmetries often observed in artificial data produced by humans. Even though these structures are unstable (adding noise destroys the surfaces to many lines), our extraction algorithm has to deal with them.

Lemma 2 For a linear tensor field, tensor core lines are straight lines.

This follows from the fact that for linear tensor fields, the linear approximation in equation (1) describes the whole data set exactly: if equation (2) holds for a small h, it holds for all h (i.e., on a whole straight line) as well. Figure 2 shows an example of a random linear tensor field containing 7 isolated tensor core lines.

5. Extracting Tensor Core Lines from Piecewise Linear Data

We assume tetrahedral partition of the three-dimensional domain. Each tetrahedron supports a linear piece if the given tensor field has a constant tensor for each vertex. The gradient field $\nabla\mathbf{T}$ consists of constant pieces per tetrahedron.

In general, tensor core structures are line features (see section 4). We extract them by first bounding the search space to individual tetrahedral cells and then to the cell boundary. The two-dimensional search on triangles reduces the extraction generally to finding point features: the intersection of core lines with the cell boundary. We will apply a root finding algorithm for feature extraction.

This discretization of the field and also the restriction to a local two-dimensional search space is applied similarly by other methods, e.g., the Sujudi and Haimes extractor [SH95] or the extractor for degenerate lines in tensor fields by Zheng and Pang [ZP04].

5.1. General Algorithm

A tensor core line consists of all locations $\mathbf{x}$ where $\mathbf{T}(\mathbf{x})\,\mathbf{r} \parallel \mathbf{r}$ and $\nabla_{\mathbf{r}}\mathbf{T}(\mathbf{x})\,\mathbf{r} \parallel \mathbf{r}$, see equation (3). This can be expressed as solutions to

$$\begin{aligned}(\mathbf{T}(\mathbf{x})\,\mathbf{r}) \times \mathbf{r} &= \mathbf{0}\\ (\nabla_{\mathbf{r}}\mathbf{T}(\mathbf{x})\,\mathbf{r}) \times \mathbf{r} &= \mathbf{0}\,.\end{aligned} \tag{4}$$

This system consists of six polynomial equations in the unknown variables $\mathbf{x}$ and $\mathbf{r} \neq \mathbf{0}$.

The polynomials are of maximum degree 1 in $\mathbf{x}$ and 3 in $\mathbf{r}$. Note that for a linear tensor field, $\nabla_{\mathbf{r}}\mathbf{T}(\mathbf{x}) = \nabla_{\mathbf{r}}\mathbf{T}$ is constant and thus independent of $\mathbf{x}$. We parameterize directions $\mathbf{r}$ such that they can be defined w.r.t. triangles. With the restriction of the local search space to triangles that bound tetrahedra, the solution of the polynomial system is equivalent to finding isolated (real) roots, i.e., points where all six polynomials simultaneously become zero.

We construct a bisection algorithm that uses the Bernstein-Bézier form (subsection 5.2) of the polynomials to exclude the presence of roots within subtriangles. Within the search space (subsection 5.3) a recursive subdivision (subsection 5.4) approximates the locations of roots in $\mathbf{x}$-$\mathbf{r}$-space with an arbitrary user-defined precision. The local feature extraction is followed by a stage that clusters solutions and then connects feature points to lines within cells (subsection 5.5) Finally, we use a filtering criterion for removing unstable solutions (subsection 5.6).

5.2. Polynomial System in Bernstein-Bézier Form

We consider bivariate polynomials $p : \mathbb{R}^2 \to \mathbb{R}$ of degree n, which are evaluated in a triangular domain $\Delta \subset \mathbb{R}^2$. Let indices $i, j, k \geq 0$. We write $p(\mathbf{u})$ in Bernstein-Bézier form as

$$p(\mathbf{u}) \;=\; \sum_{i+j+k=n} B^n_{ijk}(\mathbf{u})\, b_{ijk}\;, \quad B^n_{ijk}(\mathbf{u}) \;=\; \frac{n!}{i!\,j!\,k!}\, a_1^i\, a_2^j\, a_3^k$$

with the bivariate Bernstein polynomials B^n_{ijk} as basis and coefficients (or Bézier control points) b_{ijk}. The basis is defined w.r.t. the barycentric coordinates $a_\ell := a_\ell(\mathbf{u})$, the linear polynomials that satisfy

$$a_1\mathbf{u}_1 + a_2\mathbf{u}_2 + a_3\mathbf{u}_3 = \mathbf{u} \quad \text{and} \quad a_1 + a_2 + a_3 = 1$$

w.r.t. triangle Δ spanned by vertices $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ [HL93].

The Bernstein-Bézier representation has a number of remarkable properties. Important to our setting is the *convex hull property*: For any $\mathbf{u} \in \Delta$ – or equivalently all barycentric coordinates are nonnegative – the value $p(\mathbf{u})$ is bounded by the convex hull of the control points b_{ijk}. For scalar coefficients $b_{ijk} \in \mathbb{R}$ this means that if either all $b_{ijk} > 0$ or all $b_{ijk} < 0$ for $i + j + k = n$ then p *cannot* have a zero crossing (or real root) on Δ. We use this criterion for an iterative subdivision of a triangle Δ into smaller and smaller triangles that either may contain or cannot contain a root.

5.3. Parameterization of the Search Space

The equations in the system (4) are polynomials in $\mathbf{x}$ and $\mathbf{r}$. This means the search space consists of two independent domains: position and direction.

As pointed out before, positions $\mathbf{x}$ are restricted to points on triangles bounding tetrahedral cells. For each triangle of a tetrahedron we represent positions $\mathbf{x}$ in barycentric coordinates w.r.t. this triangle. Barycentric coordinates are defined in *local coordinates* of the triangle and therefore have two degrees of freedom. After switching to barycentric coordinates the further steps, polynomial evaluation and subdivision, are independent of the supporting triangle.

We represent a direction vector $\mathbf{r}$ as a point on a hemisphere. As not only its orientation (and thus sign) is irrelevant for the system (4) but also its scale (as long as $\mathbf{r} \neq 0$), we approximate the hemisphere by a triangulation. This way, we use the same parameterization and the same representation for $\mathbf{x}$ and $\mathbf{r}$. We remark that there is no need for an "accurate" approximation of the hemisphere. We simply use a four-sided open pyramid (see Figure 3).

For a given triangle, we have to consider a tensor field $\mathbf{T}$ that is linear in $\mathbf{x}$ and a direction vector $\mathbf{r}$ that is linearly interpolated on a triangle of the "hemisphere". Then the left-hand-sides of the system (4) give three polynomials that are linear in $\mathbf{x}$ and quadratic in $\mathbf{r}$ and three polynomials that are cubic in $\mathbf{r}$ as they don't depend on $\mathbf{x}$ because $\nabla_{\mathbf{r}}\mathbf{T}$ is constant.

Each of these six polynomials can be written in the form

$$p(\mathbf{x},\mathbf{r}) \;=\; \sum_{\substack{i+j+k=1 \\ \alpha+\beta+\gamma=3}} B^1_{ijk}(\mathbf{x})\, B^3_{\alpha\beta\gamma}(\mathbf{r})\, b_{ijk\alpha\beta\gamma}\;. \tag{5}$$

This is the tensor product of the interpolation in $\mathbf{x}$ and $\mathbf{r}$. It gives

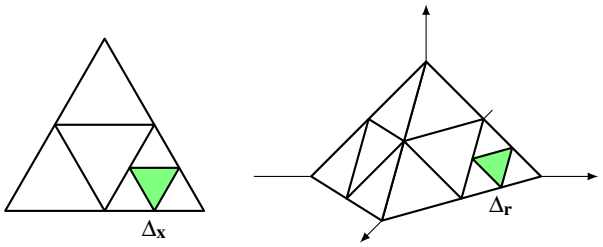

Figure 3: *The search space is parameterized on triangles $\Delta_{\mathbf{x}}$ in position space and $\Delta_{\mathbf{r}}$ on the hemisphere of possible eigenvector directions. The figure shows a pair of triangles after several subdivision steps.*

$3 \times 10 = 30$ coefficients $b_{ijk\alpha\beta\gamma}$. (All indices are nonnegative. Latin indices indicate position space, Greek indices denote direction space.) This form has degree 4 and can represent all polynomials in system (4). We use this unified representation for didactic purposes. Using the real number of degrees in $\mathbf{x}$ and $\mathbf{r}$ for each polynomial gives a smaller number of coefficients (18 or 10). Note that position and direction are expressed in *local coordinates* (or barycentric coordinates) of the triangles, such that p depends on only four coordinates in total. For the sake of a concise notation we write $p(\mathbf{x},\mathbf{r})$, $B^1_{ijk}(\mathbf{x})$ and $B^3_{\alpha\beta\gamma}(\mathbf{r})$.

5.4. Root Finding by Subdivision

Algorithms for finding roots of Bézier curves and surfaces are typically based on the convex hull property and use a recursive bisection [RHD89, HL93]. We adopt this technique. The main differences in our setting are the fact that all six equations in (4) must be satisfied simultaneously and that this is checked in two different two-dimensional domains: position and space.

Roots of One Single Polynomial

The system (4) defines six polynomials. For the initialization of the algorithm, we need to determine the Bernstein-Bézier form, i.e., the coefficients $b_{ijk\alpha\beta\gamma}$ in equation (5), for each of these polynomials. This can be done easily by sampling and interpolation: There are 3×10 coefficients and two triangular parameter domains. As equation (5) lives in a tensor-product space, interpolation of position and direction can be separated. Chose 3 or 10 distinct sampling positions in $\Delta_{\mathbf{x}}$ or $\Delta_{\mathbf{r}}$, respectively, and evaluate the values for the given polynomial. Then interpolate these values using the Bernstein-Bézier basis. The interpolation conditions define a linear system that has a unique solution. We remark that the choice of sampling positions can be arbitrary as long as they are distinct. For polynomial degree n, we use the domain points $\frac{1}{3}(i, j, k)$ with $i + j + k = n$ in barycentric coordinates. This ensures a well-conditioned system matrix. As the sampling positions are fixed, the system matrix is constant, i.e., interpolation requires only inversion or factoring. So after sampling, the conversion to Bernstein-Bézier form reduces to a linear transform than can be expressed as a matrix-multiplication.

The outline of the subdivision algorithm is as follows. We are given a pair $(\Delta_{\mathbf{x}}, \Delta_{\mathbf{r}})$ of position-direction parameter triangles and

a polynomial in Bernstein-Bézier form. The coefficients $b_{ijk\alpha\beta\gamma}$ indicate the absence of zero crossing if they are either all positive or all negative. In this case, no root is found and processing of $(\Delta_\mathbf{x}, \Delta_\mathbf{r})$ stops. If there is any sign change or zeros in the coefficients, there *may* exist roots within the parameter space $(\Delta_\mathbf{x}, \Delta_\mathbf{r})$. In this case we *subdivide* one of the parameter triangles. We alternate between subdividing $\Delta_\mathbf{x}$ in position-space and $\Delta_\mathbf{r}$ in direction-space. For both, we use a regular 1:4-split (see Figure 3). Each of the four new subtriangles is processed recursively in the same way.

System of Polynomials

We explained the basic algorithm for a single polynomial. Solving system (4) means finding solutions $\mathbf{x}, \mathbf{r}$ where all six polynomials become zero simultaneously. We test each polynomial *individually*. Only if there is a sign change in the coefficients for *all* polynomials, a simultaneous root can exist.

Every level of subdivision restricts the parameter domain and thus puts tighter bounds on the region that (potentially) contains a root. The subdivision stops either if there cannot be any root in $(\Delta_\mathbf{x}, \Delta_\mathbf{r})$ or when the magnitude of all polynomials drops below a threshold (see below). In the latter case, we have found a root, and the barycenters of the triangles define its location in parameter space.

Similar to the initial interpolation, the Bernstein-Bézier form of the subdivided polynomial can be evaluated by a linear transformation: Evaluate the polynomial at domain points in the new, smaller triangle and apply interpolation. Evaluation and interpolation can be combined to one transformation for each of the four new triangles.

Solution and Stopping Criterion

All computations involving Bernstein-Bézier polynomials are done in barycentric coordinates, which yields a concise implementation of the algorithm. However, the barycentric coordinates are relative to the current triangle, and we still need to keep track of the absolute positions of its vertices in parameter space for bounding the regions of roots. This can be done with a small amount of bookkeeping by tracking the subdivision steps such that any "child" triangle can be reconstructed from its "parent" and ultimately from the initial parameter triangle.

We stop the subdivision when we are close enough to a root. We require the magnitude of all polynomials to drop below a threshold simultaneously. For each polynomial its magnitude is bounded in $(\Delta_\mathbf{x}, \Delta_\mathbf{r})$, and we test

$$|p| \; < \; \varepsilon_\mathrm{t} \cdot ||\mathbf{T}||_\infty \; ,$$

where $|p| := \max\{\, |b_{ijk\alpha\beta\gamma}| \,\}$ is an upper bound for the magnitude of p in $(\Delta_\mathbf{x}, \Delta_\mathbf{r})$. The magnitude of the tensor field in $\Delta_\mathbf{x}$ is given as

$$||\mathbf{T}||_\infty = \sup_{\mathbf{x}\in\Delta_\mathbf{x}} \{\, ||\mathbf{T}(\mathbf{x})|| \,\} = \max_i \{\, ||\mathbf{T}_i|| \,\} \; ,$$

where $\mathbf{T}_i$ denote the constant tensors at the triangle vertices $i = 1,2,3$, and $||\mathbf{T}_i||$ denotes their spectral norm.

The parameter ε_t defines a *relative* threshold, which is independent of the local magnitude $||\mathbf{T}||_\infty$ of the tensor field within $\Delta_\mathbf{x}$.

Breadth-First Search Modification

As we hinted at in section 4, the symmetries and regular shapes inherent to data such as stress simulations of mechanical parts often lead to (3) being fulfilled or almost fulfilled on surface- or volume-type structures. Also there may be tensor core lines which do not intersect but which are part of the domain triangle. In these cases the roots are no longer isolated points but algebraic structures. As a consequence, the presented algorithm would not be efficient, as it would do an exhaustive search for all "points" on the structure. In cases where the higher-order structures are disturbed by noise and break down to lines, the algorithm still has to do a large number of subdivisions before reaching a termination criterion.

A simple modification of the algorithm enables detecting such cases: we apply a breadth-first search when looking for roots. In the implementation, we use a queue of pairs $(\Delta_\mathbf{x}, \Delta_\mathbf{r})$ of parameter regions with potential roots. If the number of elements in the queue exceeds a threshold M, we assume that the solution to (4) forms an algebraic structure and terminate the local search. If the search is terminated for one of the initial triangles $\Delta_\mathbf{r}$ that tessellate the hemisphere, we still need to consider the other triangles, because they may still contain isolated solutions.

5.5. Clustering and Line Connection

The root finding returns a list of small parameter regions $(\Delta_\mathbf{x}, \Delta_\mathbf{r})$, which are assumed to contain a solution to (4). The size of these triangles is steered by the threshold ε_t. Typically, the algorithm returns multiple regions that all refer to the same solution due to numerical noise. For this reason, we apply a post-process that clusters such regions and selects a unique representative $(\mathbf{x}, \mathbf{r})$ for each solution.

Given two representatives, we define the distance as the maximum of their distances in position- and in direction space. We employ a simple single-linkage hierachical clustering algorithm: We start with each parameter region as a single cluster. Two clusters are merged if the distance between any two elements from both clusters is smaller than a clustering threshold ε_c. We repeat this process until the number of clusters no longer changes. For each cluster, we select the triangle pair as a representative where $\max\{|p_i|\}$ is smallest for all polynomials $i = 1, \ldots, 6$. We select the center points of both triangles in this pair as the solution represented by this cluster.

This algorithm has a complexity of $O(n^3)$ in the number of solution candidates. Typically n is small: less than 200 candidates are found in the vast majority of cases. At this scale, the performance impact of the clustering algorithm is negligible.

For each tetrahedral cell of the dataset, we now connect the root points found on its faces by a line segment. Since in piecewise linear fields, tensor lines are always straight within a cell (see section 4), we greedily connect the two points with the smallest difference in eigenvector direction until no more pairs are left. Similar to the vortex core lines by Sujudi and Haimes [SH95], this gives a set of discontinuous line segments.

5.6. Filtering

If the dataset contains not only lines, but also surface- or volume-type structures where eigenvector trajectories are locally straight,

our algorithm might still find isolated solutions on these structures. These occur if the unstable structures are disturbed by noise. To filter these spurious points, we measure the numeric stability of a solution given its representative $(\mathbf{x}, \mathbf{r})$ with

$$s(\mathbf{x},\mathbf{r}) = \left|\det\left(\frac{\nabla_{\mathbf{r}_2}\mathbf{T}(\mathbf{x})\,\mathbf{r}}{\|\mathbf{T}(\mathbf{x})\|_\infty} \quad \frac{\nabla_{\mathbf{r}_1}\mathbf{T}(\mathbf{x})\,\mathbf{r}}{\|\mathbf{T}(\mathbf{x})\|_\infty} \quad \mathbf{r}\right)\right|,$$

where $\mathbf{r}, \mathbf{r}_1, \mathbf{r}_2$ are orthonormal.

The stability $s(\mathbf{x},\mathbf{r})$ measures the directional change that the eigenvector $\mathbf{r}$ of the tensor $\mathbf{T}(\mathbf{x})$ experiences when moving orthogonal to the tensor core line. If this change is small, the magnitude of the determinant in s will be small. This is an indicator that the line is numerically unstable. The normalization by $\|\mathbf{T}(\mathbf{x})\|_\infty$ ensures the independence from the scale of the tensor field.

Filtering out lines with small numeric stability s is a post-process that must be done by a user. In practice, the distribution of s over all found solutions shows an exponential behavior. In order to facilitate choosing a threshold, we visualize s on a logarithmic scale.

6. Results

We applied our algorithm to stress tensor fields obtained from structural mechanics simulations of varying complexity. The Cauchy stress tensor (often referred to as σ in mechanics literature) is a symmetric tensor that desribes the local stress state of an object experiencing small elastic deformations. Its eigenvectors point in the directions of the principal stresses. The sign of the eigenvalues indicate if the stress is compressive or tensile. Swirling structures in stress tensor fields can result from torque induced in part of a structure. As we will show, it is not always intuitive where this will happen in a complex structure subject to a load or deformation. Computing tensor core lines allows an easy identification of these phenomena. In this section, we present the results of our algorithm on several datasets, we analyze its performance and parameter sensitivity, and we compare our results to the toplogical skeleton formed by degenerate lines.

Cylinder In Figure 1 we show the eigenvector trajectories resulting from applying a torque around the long axis of a cylinder. The yellow line visible in the center is the result of our algorithm applied to this case after filtering out numerically unstable solutions. These solutions occur because the third eigenvector, which is orthogonal to the other two, points outwards from the center line everywhere in the domain. This means that the trajectories of this eigenvector are straight lines everywhere inside the cylinder. Situations like this are common in stress tensor fields, and are handled in our algorithm by the threshold M. Nevertheless, single line segments with low numeric stability s may occur due to noise (see Figure 9). After filtering them out, the clear central line visible in Figure 1 remains.

Handle This case shows a handle-like structure with a right angle being deformed in two different ways. One end is fixed, while the other end experiences different displacements. The first is a rotation around the shaft, which applies a torque to it. The second includes an additional downward shift. Figure 4 shows a tensor core line in the center of the shaft for both cases. Interestingly, a line is also visible in the "handle"-part of the structure, even though no direct torque was applied here. The line in the handle shifts away from the plane of symmetry in the second deformation case. A look at the tensor field around the core lines confirms that they are indeed the center of a swirling behavior of the tensor field.

Truck Bumper This case shows a load applied to the extreme end of the bumper of a cargo truck. Applying our algorithm to the dataset results in a large number of lines being found all over the domain. This may in part be explained by the low resolution of the simulation. After applying a filter on the numeric stability s, two lines with high stability stand out. Somewhat counterintuitively, these are found on the side opposite to the end experiencing the load. In Figure 5, we can clearly see the radial behavior of the tensor field around both lines. Finding these locations by manually inspecting the tensor field in detail would be a tedious task. Using the tensor core line extractor, they can be identified at a glance.

Crane In this dataset, the arm of a crane is exposed to a downward pull applied to the lower side of a cube at the end (see Figure 6). Similar to the truck bumper, it is not intuitively clear in which parts of the structure a swirling behavior of the tensors will occur, just from looking at the setup of the case. Almost all stable solutions we find are located in the diagonal rods on the lower side. Again, looking closely at the tensors around the core lines, we can see the radial behavior.

Spring A simulation of a coil spring being compressed and slightly bent between two plates is shown in Figure 7. Apart from numerical noise in the poorly resolved plates, we find significant tensor core lines at the center of the coil's cross-section. A look at the tensor field visualized by glyphs reveals that in this case, we do not have a simple swirling behavior of the tensors. Instead, the tensor field shows something similar to a hyperbolic behavior in vector fields. In the rightmost picture in Figure 7, we can see that eigenvector trajectories start at the wall on both sides and curve into the same direction. This direction is reversed on the top and bottom side of the spring. In the middle, there is a surface where these curves become straight lines along the diameter of the cross-section. This is exactly where we find a tensor core line.

6.1. Performance and Parameter Study

The performance of our algorithm is dependent on the dataset. If we find a large number of tensor core lines in the dataset, computation will be slower as fewer cells can be discarded early. We tested our algorithm using a consumer PC with a 4-core Intel Core i7 CPU at 3.4 GHz. Our implementation is parallelized over the faces of the dataset using OpenMP [Ope13]. Performance numbers for the different datasets shown in this paper are presented in Table 1. To examine the dependence of the performance and results of our algorithm on the parameters M, ε_t and ε_c, we conducted a parameter study. We selected baseline parameters $M = 10^3$, $\varepsilon_t = 1 \times 10^{-6}$ and $\varepsilon_c = 1 \times 10^{-3}$. We then varied each parameter separately and applied our algorithm to the cylinder dataset. The results can be seen in Figure 8. We can see that the performance is controlled by the threshold M, which controls at which point we assume we are not converging onto an isolated solution. Increasing M also increases the number of solutions we find. However, if we look at the number

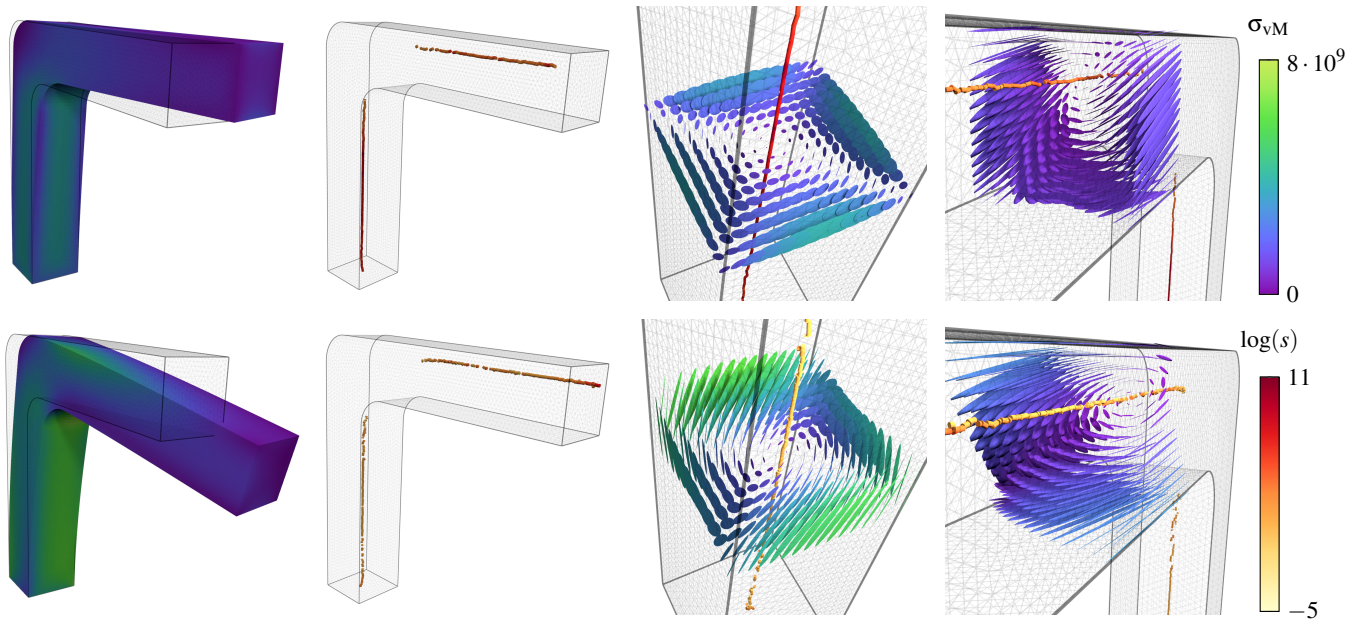

Figure 4: *Tensor core lines for two different deformations induced in a handle-like object by applying a displacement to an end surface. On the left side, we show the resulting deformations. The von Mises stress σ_{vM} is color-coded on the surface. We represent the tensor core line as tubes in the undeformed coordinate system. Their color indicates the numerical stability s. The tensor field is shown for context using elliptical glyphs.*

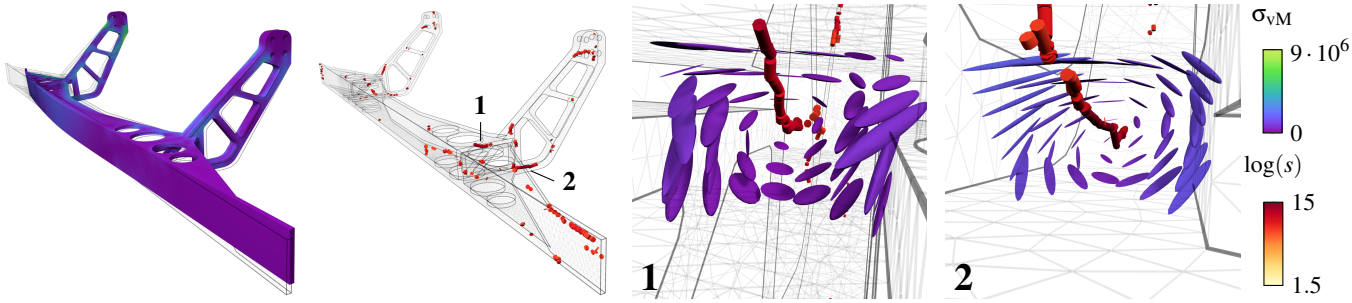

Figure 5: *Tensor core lines in a truck bumper with a load applied to one end. The deformation shown on the left is scaled 500 times for illustrative purposes. The right shows detail views of two interesting lines.*

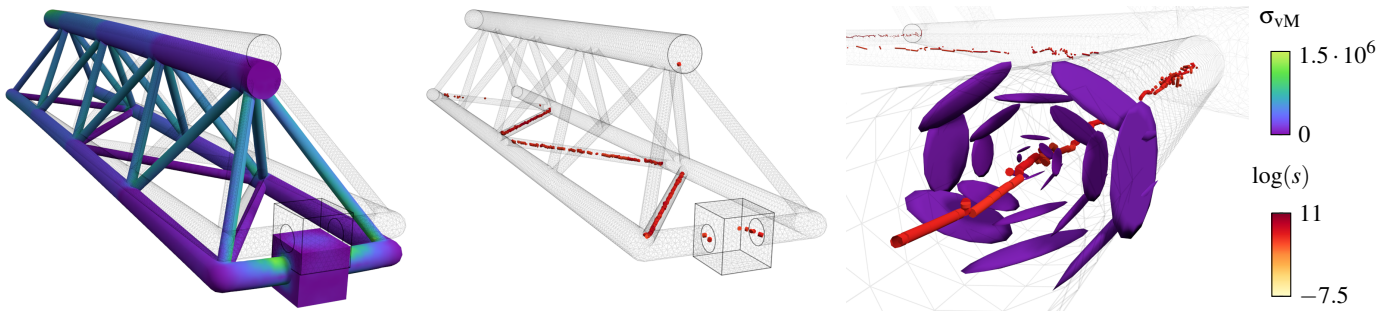

Figure 6: *Tensor core lines in a crane arm with a load applied to the end. The resulting deformation on the left is scaled 1500 times. Significant tensor core lines are only found in the lower diagonal rods. The detail image on the right shows the tensors are indeed aligned around a common core.*

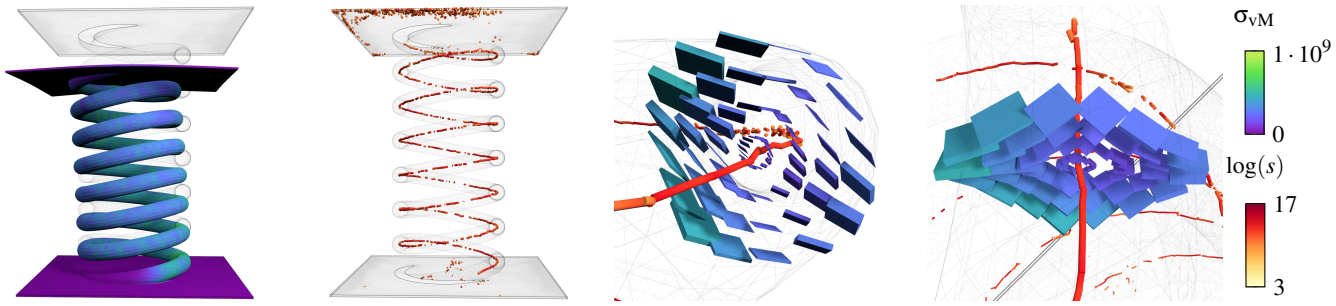

Figure 7: *A coil spring being compressed and slightly bent between two plates. We visualize the tensors near the core line with box glyphs in this case. They make it easier to see the hyperbolic behavior of the eigenvectors that occurs in the coil's cross-section.*

Dataset	# of cells	time [s]	avg. time/face [ms]
Cylinder	65 k	8.4	0.034
Handle	235 k	36	0.038
Bumper	97 k	32	0.081
Crane	108 k	63	0.146
Spring	181 k	82	0.114

Table 1: *Performance of the algorithm for the datasets presented in this paper.*

of solutions that remain after filtering based on numeric stability, it becomes clear that these solutions are only caused by noise. Increasing M did not result in any additional numerically stable lines. The parameters ε_t and ε_c have almost no noticeable impact on runtime or solutions, unless we choose unreasonable numbers. In case of ε_t, choosing a value that is larger than 1×10^{-3} causes an explosion of the number of found solutions, as the tolerance is not tight enough. Choosing ε_c smaller than ε_t means that candidate solutions belonging to the same cluster often can not be clustered, because the search radius is smaller than the distance between the triangle centers. Otherwise, ε_c is very stable. This is because for solutions which are isolated points and belong to different eigenvectors, the separation between clusters in direction space is rather large. This means the choice of ε_c is not critical as long as it is not chosen extremely small, or so large that solutions which belong to different eigenvectors are clustered together.

Stability tests on our other datasets all produced very similar results. We recommend choosing $M = 10^2$, $\varepsilon_t = 1 \times 10^{-3}$ and $\varepsilon_c = 1 \times 10^{-2}$ if performance is important. If accuracy is important, we found that choosing stricter tolerances than $M = 10^3$, $\varepsilon_t = 1 \times 10^{-6}$ and $\varepsilon_c = 1 \times 10^{-3}$ does not produce noticeably better results.

6.2. Comparison with Degenerate Lines

Tensor core lines are mathematically distinct from degenerate lines where two or more eigenvalues are equal. The criterion for finding tensor core lines is completely independent of the eigenvalues of the tensor field. However, when looking at our results in stress tensor fields, one might wonder if tensor core lines coincide with degenerate lines in practice. To investigate this, we extracted degenerate

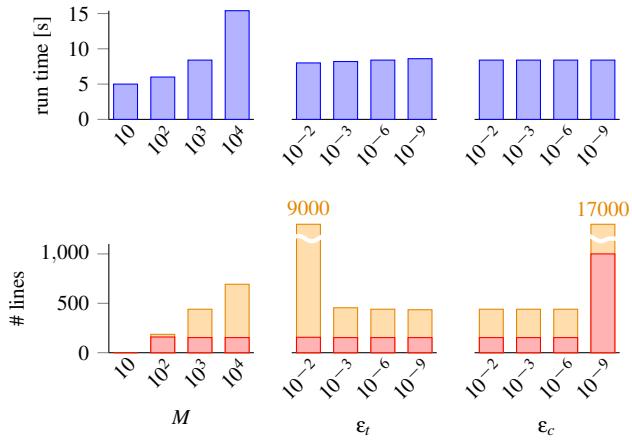

Figure 8: *Run times and number of found lines in the cylinder dataset for various different parameters. We show the total number of lines found ( ) and the number of lines remaining after filtering out numerically unstable lines ( ).*

lines from our datasets using the method presented by Zheng and Pang [ZP04]. In stress tensor fields, degenerate lines mark locations where no unique principal directions of stress can be established. We found that tensor core lines and degenerate lines sometimes coincide, but neither is a subset of the other. In the Crane dataset, degenerate lines are found near the center of the lower diagonal rods, where we also find tensor core lines. However, a lot of degenerate lines are also found in regions where no tensor core lines are located. In the Truck Bumper dataset, a degenerate line coincides with one of the two most significant tensor core lines we find, but not the other one. Closeups of both datasets are shown in Figure 10. In several datasets, such as the Cylinder and Handle, Zheng and Pang's method fails to locate any degenerate lines at all.

7. Discussion and Limitations

We introduced tensor core lines as a new feature of second-order tensor fields. It enables the quick detection of swirling behavior in eigenvector trajectories. Such behavior might not have a distinct physical meaning in all applications. However, finding core lines

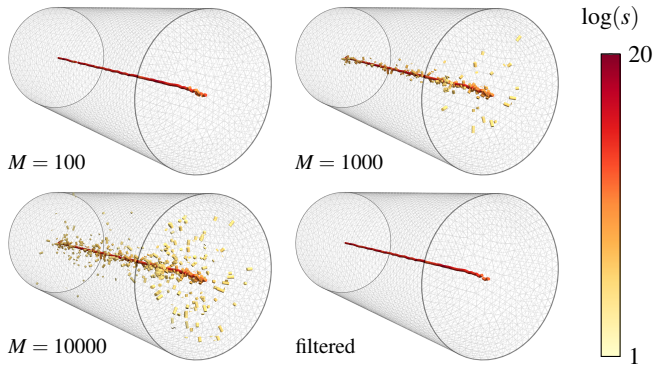

Figure 9: *Results of our algorithm on the Cylinder dataset for different choices of M. Increasing M results in more numerically unstable lines being found. If we filter them out, the result is virtually identical.*

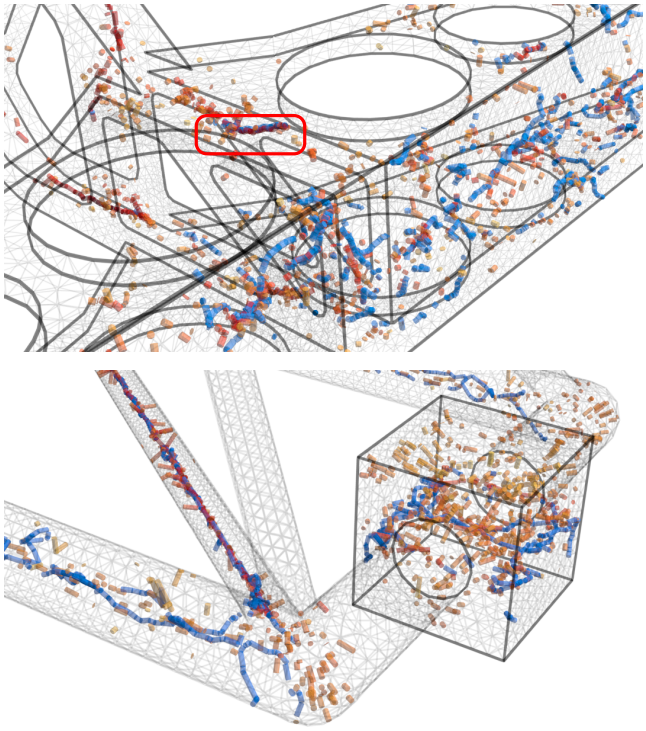

Figure 10: *Comparison of unfiltered tensor core lines (red/yellow) and degenerate tensor lines (blue) for the Truck Bumper (top) and Crane dataset (bottom). The red box marks the coincidence of a numerically stable tensor core line with a degenerate tensor line.*

helps to understand the structure of the tensor field by breaking down a complex feature into a simple line structure that can be easily visualized. In this regard, our method fits in well with other topological visualization methods.

Our method is a direct extension of the Sujudi/Haimes method for the extraction of vortex core lines in vector fields. As such, it shares many of its advantages and drawbacks. The criterion is completely local and does not require integration. As such, it is well parallelizeable and not vulnerable to accumulating numerical errors. Still, we are hardly able to reach interactive run times, as we need to perform an exhaustive search in a 5-D space. Like Sujudi/Haimes, we perform a search on piecewise linear data, which results in straight lines within cells and discontinuities of the tensor core lines at cell boundaries. Using higher-order interpolation of the tensor field would help finding continuous lines.

We have chosen to focus on piecewise linear tensor fields where each tensor component is interpolated independently. While alternative interpolation schemes have been proposed [KTW07], component-wise interpolation is still widely used as a standard approach for both tensor- and vector fields.

Unlike Sujudi/Haimes, we have no way of explicitly ensuring our solutions show only swirling behavior by restricting them to regions where the derivative has complex eigenvalues. The derivative of the tensor field $\nabla\mathbf{T}$ is a third-order tensor, for which the definition of eigenvalues and eigenvectors is non-trivial [ZPM07]. This means that we also find structures similar to hyperbolic trajectories in vector fields [MSE13, MBES16]. Further research is necessary in order to be able to distinguish these different features.

We introduced a measure for the numeric stability of tensor core lines. Unfortunately, filtering out numerically unstable solutions must be done as an interactive post-processing step, as the threshold is different for each dataset. It is worth investigating if this process can be automated. Nevertheless, the measure enables us to distinguish significant and insignificant solutions, which is a very useful tool for asessing the result of our algorithm.

Our algorithm is numerically very stable. We have three free parameters, two of which can be chosen in a wide range without significant influence on the results. The parameter M, which influences run time the most, can be chosen the same for most datasets and as such does not require fine-tuning either.

Our algorithm is only designed for extracting structurally stable line features, but surfaces or regions where the zero curvature criterion is almost fulfilled seem to be common in real-world stress tensor data. This might be due to the common occurrence of symmetries and regular shapes in human-made objects, which are most frequently the focus of structural analysis. It would therefore be interesting to investigate if these structures can explicitly be extracted.

Finally, it is worth noting that neither the formal definition of tensor core lines nor the extraction algorithm poses any restrictions on the tensor field, except that it be differentiable. As such, it might also be used on indefinite tensor data, such as the Jacobian of a vector field. Finding applications outside of stress tensor analysis is a subject for further research.

Acknowledgements

We thank Stefan Hörner and Samuel Voß for their help in setting up structural mechanics simulations to generate test data. The crane, truck bumper and spring datasets were obtained from SimScale (https://www.simscale.com). This work was partially supported by DFG grant no. TH 693/12.

References

[BP02] BAUER D., PEIKERT R.: Vortex tracking in scale-space. In *Proceedings of the Symposium on Data Visualisation 2002* (2002), Eurographics Association, pp. 233–ff. 3

[BS95] BANKS D., SINGER B.: A predictor-corrector technique for visualizing unsteady flow. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (1995), 151–163. 2

[DH93] DELMARCELLE T., HESSELINK L.: Visualizing second-order tensor fields with hyperstreamlines. *IEEE Computer Graphics and Applications 13*, 4 (1993), 25–33. 3

[DH94] DELMARCELLE T., HESSELINK L.: The topology of symmetric, second-order tensor fields. In *Proceedings of the conference on Visualization'94* (1994), IEEE Computer Society Press, pp. 140–147. 3

[GRT17] GERRITS T., RÖSSL C., THEISEL H.: Glyphs for general second-order 2d and 3d tensors. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Scientific Visualization) 23*, 1 (2017), 980–989. 3

[Har83] HARALICK R.: Ridges and valleys on digital images. *Computer Vision, Graphics, and Image Processing 22* (1983), 28–38. 2

[HL93] HOSCHEK J., LASSER D.: *Fundamentals of Computer Aided Geometric Design*. A. K. Peters, Ltd., 1993. 5

[HLL97] HESSELINK L., LEVY Y., LAVIN Y.: The topology of symmetric, second-order 3D tensor fields. *IEEE Transactions on Visualization and Computer Graphics 3*, 1 (1997), 1–11. 3

[HYW*03] HASHASH Y., YAO J. I., WOTRING D. C., ET AL.: Glyph and hyperstreamline representation of stress and strain tensors and material constitutive response. *International journal for numerical and analytical methods in geomechanics 27*, 7 (2003), 603–626. 3

[JSF*02] JEREMIĆ B., SCHEUERMANN G., FREY J., YANG Z., HAMANN B., JOY K. I., HAGEN H.: Tensor visualizations in computational geomechanics. *International Journal for Numerical and Analytical Methods in Geomechanics 26*, 10 (2002), 925–944. 3

[KHL99] KENWRIGHT D., HENZE C., LEVIT C.: Feature extraction of separation and attachment lines. *IEEE Transactions on Visualization and Computer Graphics 5*, 2 (1999), 135–144. 2

[Kin04] KINDLMANN G.: Superquadric tensor glyphs. In *Proceedings of the Sixth Joint Eurographics-IEEE TCVG conference on Visualization* (2004), Eurographics Association, pp. 147–154. 3

[KTW07] KINDLMANN G., TRICOCHE X., WESTIN C.-F.: Delineating white matter structure in diffusion tensor MRI with anisotropy creases. *Medical Image Analysis 11*, 5 (Oct. 2007), 492–502. 10

[KW06] KINDLMANN G., WESTIN C.-F.: Diffusion tensor visualization with glyph packing. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006). 3

[MBES16] MACHADO G. M., BOBLEST S., ERTL T., SADLO F.: Space-time bifurcation lines for extraction of 2D Lagrangian coherent structures. *Computer Graphics Forum 35*, 3 (2016), 91–100. 1, 10

[MK97] MIURA H., KIDA S.: Identification of tubular vortices in turbulence. *J. Physical Society of Japan 66*, 5 (1997), 1331–1334. 2

[MSE13] MACHADO G. M., SADLO F., ERTL T.: Local extraction of bifurcation lines. In *Proceedings of International Workshop on Vision, Modeling and Visualization (VMV)* (2013), pp. 17–24. 1, 3, 10

[Ope13] OPENMP ARCHITECTURE REVIEW BOARD: OpenMP application program interface version 4.0, 2013. 7

[POS*11] PAGOT C., OSMARI D., SADLO F., WEISKOPF D., ERTL T., COMBA J.: Efficient parallel and vectors feature and extraction from and higher-order data. In *Computer Graphics Forum* (2011), vol. 30, Blackwell Publishing Ltd, pp. 751–760. 3

[PR99] PEIKERT R., ROTH M.: The parallel vectors operator - a vector field visualization primitive. In *Proc. IEEE Visualization 99* (1999), pp. 263–270. 1, 2, 4

[PYW*16] PALACIOS J., YEH H., WANG W., ZHANG Y., LARAMEE R. S., SHARMA R., SCHULTZ T., ZHANG E.: Feature surfaces in symmetric tensor fields based on eigenvalue manifold. *IEEE Transactions on Visualization and Computer Graphics 22*, 3 (Mar 2016), 1248–1260. 3

[RHD89] ROCKWOOD A., HEATON K., DAVIS T.: Real-time rendering of trimmed surfaces. *SIGGRAPH 23*, 3 (July 1989), 107–116. 5

[Rot00] ROTH M.: *Automatic extraction of vortex core lines and other line type features for scientific visualization*. PhD thesis, ETH Zürich, 2000. 2

[RP98] ROTH M., PEIKERT R.: A higher-order method for finding vortex core lines. In *Proc. IEEE Visualization '98* (Los Alamitos, 1998), Ebert D., Hagen H., Rushmeier H., (Eds.), IEEE Computer Society Press, pp. 143–150. 1, 2, 3

[SH95] SUJUDI D., HAIMES R.: *Identification of Swirling Flow in 3D Vector Fields*. Tech. rep., Department of Aeronautics and Astronautics, MIT, 1995. AIAA Paper 95-1715. 1, 2, 4, 6

[SK10] SCHULTZ T., KINDLMANN G. L.: Superquadric glyphs for symmetric second-order tensors. *IEEE transactions on visualization and computer graphics 16*, 6 (2010), 1595–1604. 3

[STS07] SCHULTZ T., THEISEL H., SEIDEL H.-P.: Topological visualization of brain diffusion mri data. *IEEE Transactions on Visualization and Computer Graphics 13* (2007). 3

[SZP06] SUKHAREV J., ZHENG X., PANG A.: Tracing parallel vectors. In *Proc. SPIE* (2006), vol. 6060, International Society for Optics and Photonics, pp. 606011–606011–10. 2

[TKW08] TRICOCHE X., KINDLMANN G., WESTIN C.-F.: Invariant crease lines for topological and structural analysis of tensor fields. *Visualization and Computer Graphics, IEEE Transactions on 14*, 6 (2008), 1627–1634. 3

[TS03] THEISEL H., SEIDEL H.-P.: Feature flow fields. In *Data Visualization 2003. Proc. VisSym 03* (2003), pp. 141–148. 2

[TSW*05] THEISEL H., SAHNER J., WEINKAUF T., HEGE H.-C., SEIDEL H.-P.: Extraction of parallel vector surfaces in 3D time-dependent fields and application to vortex core line tracking. In *Proc. IEEE Visualization 2005* (2005), pp. 631–638. 3

[vGP09] VAN GELDER A., PANG A.: Using PVsolve to analyze and locate positions of parallel vectors. *IEEE Transactions on Visualization and Computer Graphics 15*, 4 (Jul-Aug 2009), 682–695. 2, 3

[WKL99] WEINSTEIN D., KINDLMANN G., LUNDBERG E.: Tensorlines: Advection-diffusion based propagation through diffusion tensor fields. In *Proc. IEEE Visualization '99* (Los Alamitos, 1999), Ebert D., Gross M., Hamann B., (Eds.), pp. 249–253. 3

[WTvGP10] WEINKAUF T., THEISEL H., VAN GELDER A., PANG A.: Stable feature flow fields. *IEEE Transactions on Visualization and Computer Graphics* (2010). 3

[ZP04] ZHENG X., PANG A.: Topological lines in 3D tensor fields. In *Proc. IEEE Visualization 2004* (2004), pp. 313–32. 3, 4, 9

[ZP05] ZHENG X., PANG A.: 2d asymmetric tensor analysis. In *VIS 05. IEEE Visualization, 2005.* (Oct 2005), pp. 3–10. 3

[ZPM07] ZHENG X., PALFFY-MUHORAY P.: Eigenvalue decomposition for tensors of arbitrary rank. *electronic-Liquid Crystal Communications* (2007). 10

[ZPP05] ZHENG X., PARLETT B., PANG A.: Topological lines in 3D tensor fields and discriminant hessian factorization. *IEEE Transactions on Visualization and Computer Graphics 11*, 4 (2005), 395–407. 3

[ZSL*16] ZHANG C., SCHULTZ T., LAWONN K., EISEMANN E., VILANOVA A.: Glyph-based comparative visualization for diffusion tensor fields. *IEEE Trans. on Visualization and Computer Graphics 22*, 1 (2016), 797–806. 3

[ZYLL09] ZHANG E., YEH H., LIN Z., LARAMEE R. S.: Asymmetric tensor analysis for flow visualization. *IEEE Transactions on Visualization and Computer Graphics 15*, 1 (Jan 2009), 106–122. 3