Vision, Modeling, and Visualization (2018)
F. Beck, C. Dachsbacher, and F. Sadlo (Eds.)

The Parallel Eigenvectors Operator

T. Oster, C. Rössl and H. Theisel

University of Magdeburg, Germany

Abstract

The parallel vectors operator is a prominent tool in visualization that has been used for line feature extraction in a variety of applications such as ridge and valley lines, separation and attachment lines, and vortex core lines. It yields all points in a 3D domain where two vector fields are parallel. We extend this concept to the space of tensor fields, by introducing the parallel eigenvectors (PEV) operator. It yields all points in 3D space where two tensor fields have real parallel eigenvectors. Similar to the parallel vectors operator, these points form structurally stable line structures. We present an algorithm for extracting these lines from piecewise linear tensor fields by finding and connecting all intersections with the cell faces of a data set. The core of the approach is a simultaneous recursive search both in space and on all possible eigenvector directions. We demonstrate the PEV operator on different analytic tensor fields and apply it to several data sets from structural mechanics simulations.

CCS Concepts

• ***Human-centered computing*** $\rightarrow$ ***Scientific visualization;***

1. Introduction

The *parallel vectors* (PV) operator [PR99] is a generic and widely spread concept in visualization. Given two 3D vector fields $\mathbf{v}_1, \mathbf{v}_2$, the PV operator delivers all points in the 3D domain where $\mathbf{v}_1$ and $\mathbf{v}_2$ are linearly dependent. It is known that the PV operator delivers structurally stable lines which we call PV lines. Applying it to different concrete vector fields, the PV operator has been proven to be a generic tool to extract vortex core lines or bifurcation lines. Several numerical methods to extract PV lines have been developed.

Recently, Oster et al. [ORT18] introduced the concept of *tensor core lines*, which represent cores of swirling hyperstreamlines in tensor fields. These lines are defined as the locations where a (real) eigenvector of the tensor field is parallel to a (real) eigenvector of the tensor field's directional derivative in the direction of this eigenvector. In this paper we extend this concept to a more generic formulation: the *parallel eigenvectors* (PEV) operator of two (not necessarily symmetric) 3D second order tensor fields. Let $\mathbf{S}(\mathbf{x})$ and $\mathbf{T}(\mathbf{x})$ be two such tensor fields. Then the PEV operator collects all points where $\mathbf{S}$ and $\mathbf{T}$ have parallel real eigenvectors. This can be concisely expressed as

$$\mathrm{PEV}(\mathbf{S},\mathbf{T}) = \{\mathbf{x} \mid \exists\, \mathbf{e}, \mathbf{e} \parallel \mathbf{S}(\mathbf{x})\,\mathbf{e} \parallel \mathbf{T}(\mathbf{x})\,\mathbf{e}\}.$$

In this work, we establish this operator by...

- ... studying its properties. In particular, we show that the PEV operator produces structurally stable line structures.
- ... presenting a numerical algorithm to extract PEV lines in piecewise linear tensor fields. The main idea is to do a recursive search not only in 3D space but simultaneously in 3D space and the space of all possible eigenvectors.
- ... applying it to compare pairs of stress tensor fields defined on the same domain.

Relation to the PV operator

At first glance, the PEV operator seems to be a straightforward extension of the classical PV operator: given two tensor fields $\mathbf{S}, \mathbf{T}$, consider the eigenvector fields as vector fields and apply the PV operator to them. However, as Oster et al. [ORT18] pointed out, using such a naive approach would suffer from several problems:

- Interpreting the eigenvectors of a tensor field as a vector field would require a heuristic choice of the orientation and magnitude of the vectors. This choice can not generally be made in a globally consistent manner.
- In regions with three real eigenvectors, it is not clear which one to choose for a representative vector field.
- Small changes in a tensor may result in a dramatic change in eigenvector direction, or even the sudden appearance or disappearance of real eigenvectors.

All of these problems show that eigenvector fields are fundamentally different from vector fields, for which the PV operator is designed. Extracting PEV lines requires new algorithms that are explicitly designed for tensor fields.

There are a number of approaches in vortex extraction where a vector field $\mathbf{v}$ is compared to the eigenvectors of a tensor field $\mathbf{S}$. This "mixed case" (eigenvector parallel to a vector) can be easily

DOI: 10.2312/vmv.20181251

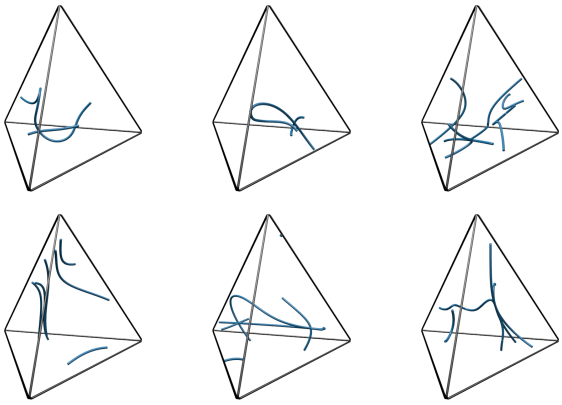

Figure 1: *PEV lines in pairs of random linear tensor fields. Top: General tensors. Bottom: Symmetric tensors. Symmetric tensors generally produce more lines, as they always have three real eigenvectors. General tensor PEV lines generally do not intersect, while it is common for symmetric tensor PEV lines to do so. If they do, three PEV lines always intersect at once, as their eigenvectors are orthogonal.*

reduced to the classical PV operator by comparing $\mathbf{v}$ and $\mathbf{S}\,\mathbf{v}$. The challenging case considered in this paper is the comparison of the eigenvectors of two tensor fields.

In the following, we first give an overview of related work and introduce the notation used throughout the paper. We then give some theoretical properties of the PEV operator in section 3, before detailing our algorithm for finding PEV lines in piecewise linear tensor fields in section 4. In section 5, we present our results for mechanical stress tensor data. We close with a discussion and future work in section 6 and section 7.

2. Related Work

The PEV operator is a generalization of tensor core lines. It is related to the PV operator, by which it was motivated, as well as tensor field visualization in general. In this section, we give a more detailed explanation of tensor core lines, as well as an overview of relevant previous work.

Tensor Core Lines

Tensor core lines were introduced by Oster et al. [ORT18] as an equivalent to vortex core lines for tensor fields. Similar to the Sujudi/Haimes criterion for vortex core lines [SH95], they define tensor core lines as the locations where eigenvector trajectories have locally vanishing curvature. For a tensor field $\mathbf{T}$ and a vector $\mathbf{r}$, a tensor core line passes through location $\mathbf{x}$ if

$$\mathbf{r} \parallel \mathbf{T}(\mathbf{x})\,\mathbf{r} \parallel \nabla_{\mathbf{r}}\mathbf{T}(\mathbf{x})\,\mathbf{r}\,.$$

To understand this criterion, it is important to note that $\mathbf{r} \parallel \mathbf{T}\,\mathbf{r}$ implies that $\mathbf{r}$ is an eigenvector of $\mathbf{T}$. This is equivalent to the more commonly used formulation $\mathbf{T}\mathbf{r} = \lambda\mathbf{r}$. The tensor core line criterion consequentially states that $\mathbf{r}$ is an eigenvector of $\mathbf{T}(\mathbf{x})$, and that $\mathbf{r}$ is also an eigenvector of $\nabla_{\mathbf{r}}\mathbf{T}(\mathbf{x})$, the directional derivative of $\mathbf{T}$ along $\mathbf{r}$. In other words, $\mathbf{T}(\mathbf{x})$ and $\nabla_{\mathbf{r}}\mathbf{T}(\mathbf{x})$ have a parallel eigenvector $\mathbf{r}$. In this work, we relax this criterion and find locations where any two tensor fields have parallel eigenvectors.

Parallel Vectors

The parallel vectors operator was introduced by Peikert and Roth in 1999 [PR99] as a generalization of a concept that had been used with slight variations in a lot of different contexts. Among these are ridge detection in scalar fields [Har83], extraction of attachment/separation lines in flows [KHL99], and the identification of vortex core lines [SH95, BS95].

In his PhD thesis, Martin Roth [Rot00] gives an overview of several numerical algorithms for the parallel vectors operator. Most of them are based on first finding intersections of PV lines with the cell faces of a dataset. The resulting intersection points are then connected to lines using different kinds of heuristics.

An alternative approach is to trace parallel vector lines starting from a seed point. Algorithms using this general approach have been proposed by Banks and Singer [BS95], Miura and Kida [MK97], Sukharev et al. [SZP06] and Theisel et al. [TS03]. Methods for avoiding the accumulation of errors when tracing PV lines were introduced by van Gelder and Pang [vGP09], as well as Weinkauf et al. [WTvGP10].

While most PV algorithms operate on piecewise linear data that is not time-dependent, there are some publications that deal with higher-order data or use higher-order methods. This includes approaches for finding curved vortex core lines [RP98], scale-space techniques [BP02], and computing the PV operator on time-dependent [TSW*05, FPH*07] or piecewise analytic vector fields [POS*11].

Tensor Field Visualization

Tensor fields are generally visualized by Glyph-based, Integration-based or Topology-based methods.

Tensor glyphs show the properties of a single tensor as a geometric object. These glyphs vary in complexity depending on the properties of the tensor that is visualized. Glyphs have been designed for symmetric positive definite tensors [Kin04], indefinite symmetric tensors [SK10], and general tensors in 2D and 3D [GRT17]. Specialized glyphs for structural mechanics applications [HYW*03] as well as comparative visualization of medical diffusion tensor fields [ZSL*16] have also been proposed. To visualize a tensor field using glyphs, they are usually placed on the grid nodes of the dataset or on a regular grid superimposed on the data. Kindlmann and Westin [KW06] proposed a more sophisticated glyph placement strategy that avoids occlusion.

Integration-based methods create geometric structures by following the field of eigenvector directions starting from a seed structure. The simplest example of this class is the hyperstreamline [DH93]. It is commonly implemented as a tube following an eigenvector corresponding to some ordered eigenvalue. The cross-section of the tube is scaled and rotated according to the other

eigenvalues. Weinstein et al. [WKL99] introduced a similar concept that is more stable in the vicinity of isotropic points. An extension of hyperstreamlines are hyperstreamsurfaces [JSF*02], which use a line instead of a point as the seed structure.

Topology-based methods focus on extracting a topological skeleton capturing critical structures in tensor fields. Degenerate structures where two or more eigenvectors are equal, were first described for symmetric tensor fields by Delmarcelle [DH94] and Hesselink [HLL97]. Zheng and Pang [ZP04,ZPP05] introduced numerical algorithms for extracting such structures. A more stable approach for noisy data was proposed by Tricoche et al. [TKW08] The topology of asymmetric tensor fields has also been studied [ZP05, ZYLL09]. Recently, surfaces of neutral and traceless tensors were added to the topological features of symmetric tensor fields [PYW*16].

3. Theoretical Considerations

In this section we study which structurally stable structures the PEV operator yields. Structurally stable here means stable in the presence of noise. From the knowledge of the PV operator for vector fields, one would expect curves as PEV solutions. The case, however, is slightly more complicated because eigenvectors can transition from real to imaginary, and they are not uniquely defined in isotropic regions. Even under consideration of these cases, we can formulate the main theorem

Theorem 1 The PEV operator delivers structurally stable curves that are either closed or end at the boundaries of the domain.

We provide the proof for this theorem in the additional material accompanying this paper.

Bifurcation points We define bifurcation points as locations where two or more PEV lines intersect, i.e., locations with more than one pair of parallel eigenvectors. If $\mathbf{S},\mathbf{T}$ are general second order tensors, bifurcation points are structurally unstable, i.e., they disappear under small perturbations of $\mathbf{S},\mathbf{T}$. To show this, we consider a PEV line l and observe the other eigenvectors (the ones that do not define l) along its path. Since they are not constrained by each other, more than one condition must be fulfilled along l for the other eigenvectors to become parallel. This can be interpreted as having at least two independent scalar values that must vanish at the same point along l. If this happens, adding noise will split up the points on l of common zero crossings.

This situation is different if $\mathbf{S},\mathbf{T}$ are symmetric. In this case, there are structurally stable bifurcations points where all three PEV lines intersect, i.e., where $\mathbf{S},\mathbf{T}$ have three pairs of parallel eigenvectors. This can be shown as follows: If $\mathbf{S},\mathbf{T}$ have two pairs of parallel eigenvectors, the third pair must be parallel as well, due to the orthogonality of the eigenvectors. Further, we consider again a PEV line l that is defined by the vector $\mathbf{e}$ along l that is eigenvector of both $\mathbf{S}$ and $\mathbf{T}$. Then every other eigenvector of $\mathbf{S}$ and $\mathbf{T}$ is perpendicular to $\mathbf{e}$ and can therefore be expressed by one number: the rotation angle around $\mathbf{e}$. This way, the conditions of further pairs of common eigenvectors can be described as the roots of one scalar function. This is structurally stable: adding noise will slightly change the location of l and slightly change the location of zero crossings on l, but does not make them disappear. Some PEV lines with bifurcation points can be seen in Figure 1.

4. Extracting PEV Lines from Piecewise Linear Data

We will now detail our algorithm for finding PEV lines in piecewise linear tensor fields. We assume that both tensor fields are defined on the vertices of the same tetrahedral mesh. The general approach is to first find all intersections of PEV lines with the faces of the mesh, and then to connect those points to lines.

We showed that PEV structures are lines in the structurally stable case. It follows that their intersections with the triangular faces of a tetrahedral mesh are isolated points. Finding an analytic solution to the parallel eigenvector problem is impossible, as it involves the intersection of cubic polynomials. Instead, we opt for a numerical approach that is based on recursive subdivision both on the triangle and in the space of possible eigenvector directions.

Our algorithm can be summarized as follows: We first find a direction $\mathbf{r}$ which becomes an eigenvector of both $\mathbf{S}$ and $\mathbf{T}$ at some (possibly different) points inside the triangle. If such a direction is found, we subdivide the triangle and check the parts for possible eigenvector directions again. We do this until we converge on a single point where both $\mathbf{S}$ and $\mathbf{T}$ have parallel eigenvectors.

In order to find a valid direction $\mathbf{r}$, we perform another recursive search in the space of possible eigenvector directions. We represent this space as some triangulation of a hemisphere centered at the origin. For each triangle of directions, we have to decide whether it contains a valid eigenvector direction, i.e., a direction that can become an eigenvector of both $\mathbf{S}$ and $\mathbf{T}$ within the current sub-triangle in space. If we are sure that there are no valid directions in the triangle, we can discard it. If we are sure that all directions within a triangle are valid directions, we can terminate the recursion. If we can not be sure whether the triangle contains valid directions, we subdivide it and check the parts again.

Our algorithm is similar to the one described by Oster et al. [ORT18]. Both are numerical algorithms that find singularities of polynomials of higher degree. However, the problem we solve in this work is different, as it does not involve the derivatives of a tensor field, leading to a different algorithm. In the following, we describe the details of this algorithm.

4.1. Mathematical Basis

A linear tensor field on a triangle is completely defined by the tensors at its three corners. We denote the set of corner points as $\mathbf{x}_\triangle = \{\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3\}$, and the set of corner tensors as $\mathbf{S}_\triangle = \{\mathbf{S}_1,\mathbf{S}_2,\mathbf{S}_3\}$ and $\mathbf{T}_\triangle = \{\mathbf{T}_1,\mathbf{T}_2,\mathbf{T}_3\}$. We express the tensor fields in barycentric coordinates $\mathbf{w} = (w_1,w_2,w_3)^\mathrm{T}$:

$$\mathbf{S}(\mathbf{w}) = \sum_i w_i\mathbf{S}_i\,,\quad \mathbf{T}(\mathbf{w}) = \sum_i w_i\mathbf{T}_i\,,\quad \text{with}\quad \sum_i w_i = 1\,.$$

The position (in barycentric coordinates) at which an arbitrary direction $\mathbf{r}$ becomes an eigenvector in $\mathbf{S}$ is given by the solution to

$$\mathbf{S}(\mathbf{w})\mathbf{r} = \sum_i w_i\mathbf{S}_i\mathbf{r} = \lambda\mathbf{r}\,.$$

Rather than needing the exact position, we want to know if the position is inside the triangle or not, i.e., if $\mathbf{r}$ is a valid eigenvector direction for $\mathbf{S}$. In barycentric coordinates, a point is inside the triangle if all $w_i > 0$. Since the scaling factor λ is arbitrary, we eliminate it:

$$\sum_i \tilde{w}_i \mathbf{S}_i \mathbf{r} = \mathbf{A}(\mathbf{r})\tilde{\mathbf{w}} = \mathbf{r},$$
$$\text{with} \quad \mathbf{A}(\mathbf{r}) = \begin{pmatrix} \mathbf{S}_1\mathbf{r} & \mathbf{S}_2\mathbf{r} & \mathbf{S}_3\mathbf{r} \end{pmatrix}, \quad \tilde{\mathbf{w}} = \mathbf{w}/\lambda,$$

and only require that all $\tilde{w}_i$ have the same sign. Using Cramer's rule, we can give an analytic solution for the components of $\tilde{\mathbf{w}}$:

$$\tilde{w}_i = \frac{\det \mathbf{A}_i(\mathbf{r})}{\det \mathbf{A}(\mathbf{r})}$$

Here, $\mathbf{A}_i$ denotes the matrix $\mathbf{A}$ with its i-th column replaced by $\mathbf{r}$. Note that all $\tilde{w}_i$ are divided by the same factor $\det \mathbf{A}(\mathbf{r})$. Since this influences all signs of $\tilde{w}_i$ equally it can be ignored, leading to

$$\hat{w}_i(\mathbf{r}) = \det \mathbf{A}_i(\mathbf{r}). \tag{1}$$

The equations for $\mathbf{T}$ are analogous. In the following, we show all equations for $\mathbf{S}$ only. The equivalent equations for $\mathbf{T}$ can be obtained trivially by substituting $\mathbf{T}$ for $\mathbf{S}$. We denote the solutions for $\mathbf{S}$ and $\mathbf{T}$ by $\hat{\mathbf{w}}_S$ and $\hat{\mathbf{w}}_T$ respectively, whenever it is necessary to discriminate them.

4.2. Subdivision in Direction Space

The core of the algorithm is to find a direction $\mathbf{r}$ for which all components of $\hat{\mathbf{w}}_S(\mathbf{r})$ have the same sign, and all components of $\hat{\mathbf{w}}_T(\mathbf{r})$ also have the same (but possibly different) sign. This means that the direction $\mathbf{r}$ becomes an eigenvector somewhere inside the triangle for both $\mathbf{S}$ and $\mathbf{T}$. Note that $\hat{w}_i(\mathbf{r})$ is cubic in $\mathbf{r}$. Finding an analytic solution for $\mathbf{r}$ means analytically finding the intersections of the roots of $\hat{w}_i(\mathbf{r})$, which is impossible. Instead, we solve the problem numerically by applying another recursive search in the space of all possible eigenvector directions. The magnitude and orientation of $\mathbf{r}$ is not significant. We can therefore represent this space by some triangulation of a hemisphere centered at the origin (Figure 2, right). We again express a direction in a triangle $\mathbf{r}_\triangle = \{\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3\}$ on this hemisphere in barycentric coordinates u_j of its corner vectors:

$$\mathbf{r}(\mathbf{u}) = \sum_j u_j \mathbf{r}_j.$$

Substituting this in (1), the barycentric coordinate functions now become

$$\hat{w}_i(\mathbf{u}) = \det\left(\sum_j u_j \mathbf{A}_i(\mathbf{r}_j)\right).$$

We can now express $\hat{w}_i$ in Bernstein-Bézier basis:

$$\hat{w}_i(\mathbf{u}) = \sum_{\substack{j,k,l \geqslant 0, \\ j+k+l=3}} \frac{3!}{j!\,k!\,l!}\, u_1^j u_2^k u_3^l \cdot \alpha_{jkl}.$$

Here, α_{jkl} are the 10 coefficients needed to express a trivariate polynomial of degree 3. We use the property that a polynomial in Bernstein-Bézier form is bounded in its domain by the convex hull of its coefficients [Far97]. This means that $\hat{w}_i$ is positive over the whole triangle if all $\alpha_{jkl} > 0$, and negative over the whole triangle if all $\alpha_{jkl} < 0$. If the α_{jkl} have different signs, $\hat{w}_i$ might become 0 somewhere inside the triangle.

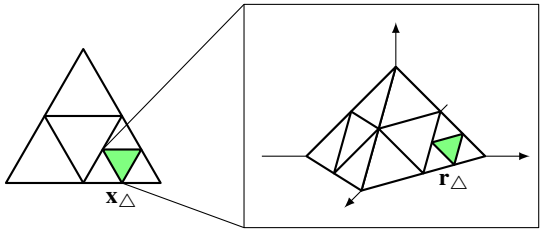

Figure 2: *Two-level recursion scheme for finding intersections of PEV lines with the faces of piecewise linear tensor fields. For each sub-triangle $\mathbf{x}_\triangle$, a recursive search in the space of possible eigenvector directions is performed to find a direction $\mathbf{r}$ that becomes an eigenvector of both $\mathbf{S}$ and $\mathbf{T}$ within $\mathbf{x}_\triangle$. (Image similar to [ORT18].)*

We use this when recursively subdividing the triangle $\mathbf{r}_\triangle$. If any $\hat{w}_i$ might have roots within the triangle according to the Bernstein-Bézier coefficients, then we can not make a decision. We need to subdivide the triangle and check the different parts again. If no $\hat{w}_i$ can have roots within the triangle as indicated by the coefficients, then there are two possibilities:

1. All $\hat{w}_i$ have the same sign everywhere on the triangle
2. The $\hat{w}_i$ have different signs everywhere on the triangle

In case 1., all directions within the triangle become eigenvector directions somewhere in $\mathbf{x}_\triangle$ for both $\mathbf{S}$ and $\mathbf{T}$. If this happens, we can accept any direction within the current triangle as a possible solution. In case 2., no direction within the triangle can become an eigenvector of both $\mathbf{S}$ and $\mathbf{T}$, and the triangle is discarded. When the triangle becomes smaller than some subdivision threshold ε_r, and we still can not say for sure that there are no possible eigenvector directions inside, we accept the central direction as a candidate.

4.3. Final Numerical Algorithm

The complete algorithm for finding intersections of PEV lines with a triangle of the dataset now works as follows: Start with the complete triangle as $\mathbf{x}_\triangle$. Then, search for a direction that becomes an eigenvector of both $\mathbf{S}$ and $\mathbf{T}$ somewhere inside the triangle by using the algorithm described in subsection 4.2. If such a direction is found, subdivide the triangle and process the parts recursively. If no direction is found, discard the triangle. When a spatial sub-triangle becomes smaller than a subdivision threshold ε_s, we accept the center of the triangle as a solution candidate.

The result of the algorithm is a list of points on $\mathbf{x}_\triangle$ with corresponding eigenvector directions. This list of points has to be post-processed for two reasons:

1. For each intersection of the PEV line with the triangle, multiple adjacent candidate points may be found. This happens if eigenvectors of $\mathbf{S}$ and $\mathbf{T}$ are closer than ε_r in a region larger than ε_s, e.g., because the gradient of the tensor fields is very small, or because the PEV line intersects the face at a very steep angle. Choosing ε_r very small helps with this, but it can not be avoided in the presence of limited numerical precision on a computer.

2. A candidate point might not be a PEV point at all. These false positives occur if there are directions $\mathbf{r}$ that become eigenvectors of one of the tensor fields inside $\mathbf{x}_\triangle$, while $\mathbf{A}(\mathbf{r})$ has rank 1 for the other tensor field. For this case, $\hat{\mathbf{w}} = \mathbf{0}$, which means that a consistent sign of all components can never be determined, and subdivision can not be terminated early, even if the tensor field does not have any valid eigenvector directions inside $\mathbf{x}_\triangle$.

We deal with item 1. by clustering candidate points that are closer than a certain distance threshold ε_c. We employ a simple single-linkage hierachical clustering algorithm [ELLS11]. Each candidate point starts as a separate cluster. Clusters are merged if the smallest distance between them is smaller than ε_c. This is repeated until the number of clusters converges. The clustering algorithm is the same as the one used by Oster et al. [ORT18] for a similar purpose. We then select the point in each cluster where the corresponding eigenvectors are most parallel as the representative and discard the others. Since we already have eigenvector directions for each point, we do not need to explicitly compute them again. Instead, we use the parallelity error

$$e_p = \left\| \frac{\mathbf{S}(\mathbf{w})\mathbf{r}}{\|\mathbf{S}(\mathbf{w})\mathbf{r}\|} \times \frac{\mathbf{r}}{\|\mathbf{r}\|} \right\| + \left\| \frac{\mathbf{T}(\mathbf{w})\mathbf{r}}{\|\mathbf{T}(\mathbf{w})\mathbf{r}\|} \times \frac{\mathbf{r}}{\|\mathbf{r}\|} \right\|,$$

which measures the deviation of $\mathbf{r}$ from the true eigenvectors of both $\mathbf{S}(\mathbf{w})$ and $\mathbf{T}(\mathbf{w})$.

In order to address item 2., we discard all candidate points for which e_p is greater than some parallelity threshold ε_p. This threshold can be chosen quite coarse (e.g. 0.01), as e_p is typically quite large for false positive candidate points.

In certain cases, the PEV line might not intersect the triangle at a single point. This happens in the structurally unstable cases where eigenvectors are parallel on a structure with a dimension larger than 1, or where the PEV line is completely in the plane of the triangle. In these cases, a recursive subdivision will not converge on isolated points and slow down the algorithm considerably. To mitigate this, we terminate the recursion if the number of triangle subdivision operations exceeds a reasonable threshold.

Once we have clustered the candidate solutions and removed false positives, we have a number of final PEV points for each face of the mesh. These PEV points are now connected to lines on a cell-by-cell basis. This problem is also faced when computing the PV operator, where it has been solved in a variety of ways using different heuristics, which can be employed here as well. In our implementation, we simply connect two points if they are the only two intersections of a PEV line with a grid cell. In case of more than one intersection, we greedily connect pairs of points that have the most similar parallel eigenvector directions, assuming that PEV lines are generally smooth relative to the grid resolution.

5. Results

We applied our method to different stress tensor fields from structural mechanics simulations. Stress tensors are symmetric tensors that describe the local stress at a point in a material under acting force. Its eigenvectors are real and orthogonal and are aligned with the principal stress directions acting on the point. When comparing two different stress tensor fields, PEV lines occur where two principal stress directions align. We show PEV lines for three different stress tensor datasets: Two different point loads applied to a uniform material, two different traction forces applied to the end of a clamped beam, and two different load scenarios applied to a flange.

We used the same parameters for all datasets: $\varepsilon_s = 1.0 \times 10^{-3}$, $\varepsilon_c = 5 \times \varepsilon_s$, $\varepsilon_d = 1.0 \times 10^{-9}$, $\varepsilon_p = 1.0 \times 10^{-3}$. Our results were computed on a 4-core Intel Core i7 CPU at 3.4 GHz.

Point Loads

In this example, we compare two different point loads applied to a uniform material with infinite extent. The first load (red arrow in Figure 3) is a compressive force, the second load (blue arrow) is a tensile force of equal magnitude. We compute the PEV operator for the two resulting stress tensor fields. The Point Load case has a closed analytic solution [Saa13], which we sampled on a regular grid with $100 \times 50 \times 50$ points using the `vtkPointLoad` source from the Visualization Toolkit [SML06]. We then tetrahedralized the data, resulting in 1.1 million cells and 4.8 million faces. The computing time for this dataset was 4.2 h, which means that PEV intersections on each face were found in 3.2 ms on average.

Since the point loads were applied in the same plane, this synthetic dataset shows the rare case where eigenvectors are parallel on a plane instead of a line. This degenerate case also accounts for the long computing time, as for each face intersected by the PEV plane, the recursive subdivision can not be terminated early. Even though this structurally unstable case produces visual artifacts when using our method, interesting PEV line structures are still visible. There is a bifurcation point exactly at both load points, extending into a curved ring slightly below the surface. At the second intersection of this ring with the central plane, another closed PEV structure embedded into the plane becomes visible. Within a PEV plane, structures where all three eigenvectors are parallel become lines, instead of points. A similar structure can be observed starting at the load points and leading outwards. In the center of the dataset, there is another PEV line, orthogonal to the plane and slightly curved downwards, separating the two load points. For this dataset, we colored the PEV lines by the absolute eigenvalue ratio of the parallel eigenvectors. Since both tensor fields result from forces of equal magnitude, this ratio makes visible the directions in which forces propagate outwards from the load points.

Clamped Beam

Next, we extracted PEV lines for a beam that is fixed on one side. We applied two different traction forces on the free end of the beam, whose directions are shown in Figure 4. The Clamped Beam dataset consists of 150k cells and 600k faces. The computing time was 26 min, which means 2.6 ms per face on average.

There are two regions of particular interest in the Clamped Beam. The first is near the middle of the beam, where a curved structure has high eigenvalues in both tensor fields (visible in red and blue in Figure 4). This is where the beam experiences a lot of stress, and therefore the tensor fields have a high magnitude. The second is near the fixed end. Here, the stress is high particularly for

the more diagonal force indicated by the red arrow in Figure 4. We find a high number of bifurcation points in this region. Additionally, near the middle of the beam all three eigenvector directions are parallel along a line structure near the center with considerable length. This area seems to be the most similar between the two scenarios in terms of stress directions.

Flange

Our final stress tensor dataset is a flange geometry from an OpenFoam [Ope] tutorial. We subjected the flange to two different loads, applied on the back wall and the outlet tube (see red and blue arrows in Figure 5). The original mesh uses polygonal cells, which is why we resampled the data, resulting in 1.2 million cells and 5 million faces. The computing time was 36 min, i.e., 0.5 ms per face.

The dataset exhibits a lot of PEV lines, which can be seen in Figure 5 on the left. Most of the PEV lines correspond to eigenvectors with small eigenvalues in both tensor fields. We therefore filtered out all PEV lines where both eigenvalues are very small in the center and right images in Figure 5. Especially prominent are two bifurcation points with high eigenvalues between the two outer screw holes and the central tube. There are also PEV lines leading outwards both above and below the screw holes. In general, the most similar directions of significant stress are near the screw holes and in the area where the large outlet tube meets the central block.

6. Discussion

The PEV operator was introduced as a generic operator, its interpretation is dependent on the application scenario.

For stress tensors in mechanical engineering, the PEV operator gives insight into the alignment of the tensors under different acting forces. Areas with PEV lines can be wanted or unwanted. In areas with present PEV lines, the stress tensor is similarly oriented for different external forces. This could be used e.g., for deciding the placement of structural reinforcements or to guide the selection of materials. In regions without PEV lines, there is no stress in a preferred direction when applying different outer forces and material with a more isotropic behavior could be used.

Besides this particular interpretation, there are general interpretations that are common to all applications of the PEV operator. The PEV operator is agnostic to isotropic scaling of the tensors. It gives information about the orientation of the tensors only. In this way, the PEV operator can be seen as an addition to many standard measures for comparing tensors like norm, trace, or eigenvalues.

The presented algorithm for piecewise linear tensor fields does not use any derivatives of the data. It depends on a number of thresholds to guide subdivision levels and filtering. The spatial subdivision threshold ε_d influences the accuracy of the resulting PEV lines. A small threshold means more subdivisions and is one of the main factors influencing performance. Since subdivision converges to single points, the computing time increases logarithmically when decreasing ε_d.

The directional subdivision threshold ε_r guides the accuracy of the obtained eigenvector direction. Typically, the smaller the current spatial triangle $\mathbf{x}_\triangle$ becomes, the smaller the region of valid eigenvector directions. For increasing subdivision level in space, the valid eigenvector directions will converge on a point. This means that for small ε_d, the recursion in the space of directions will generally proceed to the highest subdivision level. The influence of ε_r on computation time is about the same as for ε_d. However, an accurate determination of eigenvector direction is essential to decrease the number of candidate PEV solutions that have to be clustered. This means that ε_r should be chosen very small. We find $\varepsilon_r = 1 \times 10^{-9}$ to be a choice that provides consistently good results.

Because our algorithm can produce multiple candidate points for an intersection of a PEV line with a tetrahedral face, we need to cluster the results. Theoretically, all candidate points should be in adjacent triangles, as (unoriented) eigenvector directions in linear tensor fields do not oscillate on small scales. However, due to numerical noise and rounding errors on a computer, some candidate triangles might not be exactly adjacent to each other. To bridge this gap, the clustering threshold ε_c defines the radius in which two candidate solutions are considered to belong to the same cluster. Because the numerical noise influencing the size of gaps between candidate solutions is random, we do not expect candidates to be more than two or three lengths of ε_d from each other. We recommend to set ε_c to some fixed multiple of ε_d. In our experiments, $\varepsilon_c = 5\varepsilon_d$ proved sufficient for all datasets.

The parallelity threshold ε_p is used to weed out false positive candidates that are a byproduct of our algorithm. It must be chosen carefully to separate false positive solutions from numerical errors. Because of this threshold, the spatial subdivision threshold ε_d can not be chosen arbitrarily large. The larger ε_d, the larger the possible difference in eigenvector direction between the real PEV point and the tensor at the center of the triangle, which is chosen as a representative. In general, the choice of ε_p is dependent on ε_d. More spatial subdivision levels enable a smaller choice of the parallelity threshold. In our experiments, a choice of $\varepsilon_p = 1 \times 10^{-2}$ for $\varepsilon_d = 1 \times 10^{-3}$, and $\varepsilon_p = 1 \times 10^{-3}$ for $\varepsilon_d = 1 \times 10^{-6}$ worked very well.

7. Limitations and Future Research

Limitations can be discussed from two points of view: the operator itself and the presented numerical extraction algorithm. A limitation of the PEV operator is that it can only be applied to problems where the norm of the tensors does not matter. This limits the applicability but on the other hand focuses on features of the tensor fields that are less covered by other methods.

The presented extractor works for piecewise linear tensor fields only. An extension to hexahedral grids as well as higher order interpolations is subject of future research. The performance of the algorithm can be improved by parallelization. In principle, the algorithm is parallelizable (each cell can be treated independently). However, even if this is carefully carried out, interactive frame rates (for instance for comparing time-dependent tensor fields) are hardly achievable because we still have to do a search in a 5D space. Due to the possibility of many candidate solutions for each intersection of a PEV line with a face, we can not give an upper limit on the error of the PEV line position. This might limit its applicability in cases where a highly accurate PEV line is required.

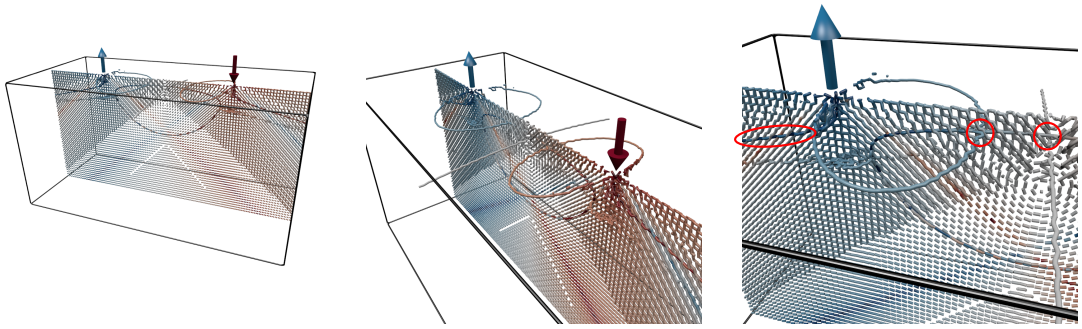

Figure 3: *PEV lines for the Point Load dataset. Lines are colored by absolute eigenvalue ratio. A compressive force (red arrow) is applied to obtain one stress tensor field, while an equivalent tensile force (blue arrow) is applied for the second stress tensor. Due to the parallel application of forces orthogonal to the surface, a plane of parallel eigenvectors forms between the load points.*

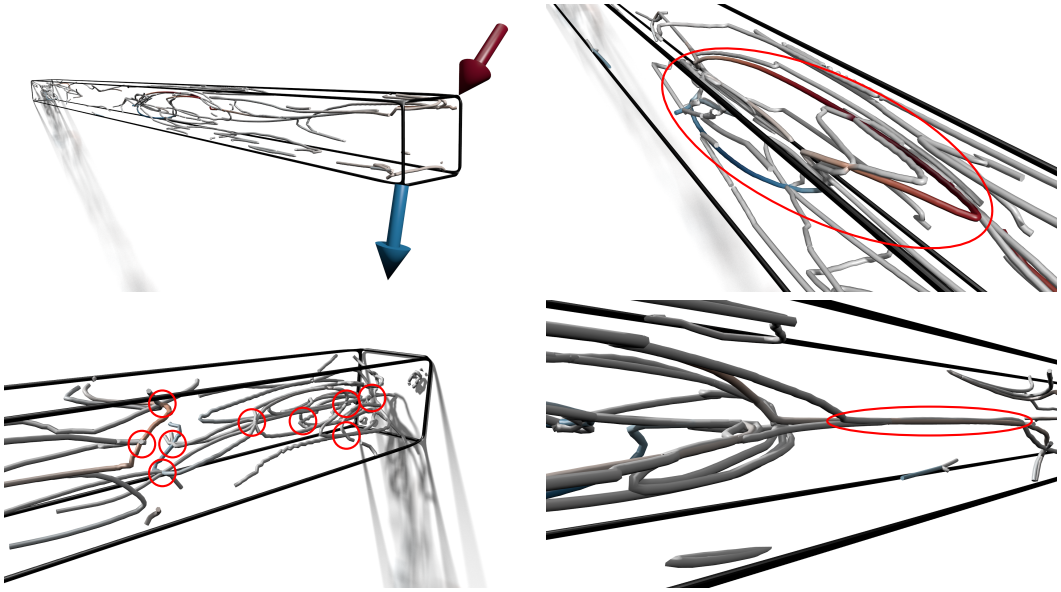

Figure 4: *PEV lines for the Clamped Beam dataset. Two different traction forces are applied to the free end of the beam (red and blue arrows), while the other end is fixed on the wall. Lines are colored by the eigenvalue of the stress tensor corresponding to the red arrow (red is positive, blue is negative). Interesting structures mentioned in the text are highlighted.*

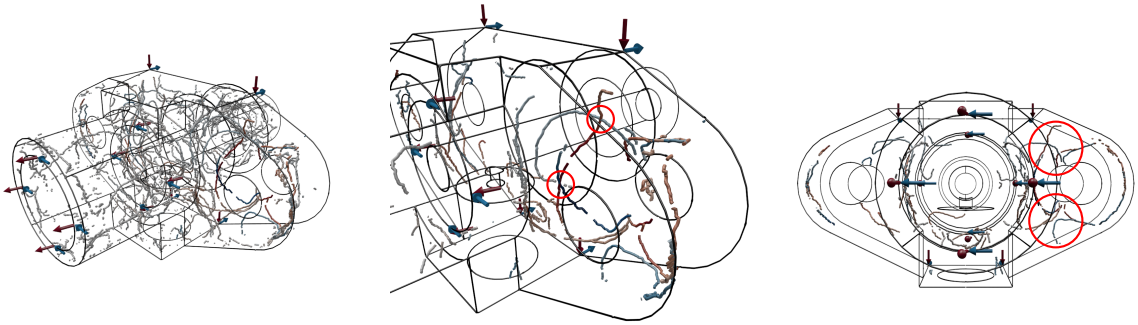

Figure 5: *PEV lines for the Flange dataset. Two different load scenarios (indicated by blue and red arrows) are simulated. Lines are colored by the eigenvalue corresponding to the red arrows. To avoid visual clutter, we filtered the lines in the middle and right image by eigenvalue, removing lines where both eigenvalues are very small. Interesting structures mentioned in the text are highlighted.*

We stated in Theorem 1 that PEV lines are generally closed. However, the results obtained from our algorithm sometimes exhibit gaps. Because the extractor is a numerical algorithm, and not a combinatorial one, we will sometimes not find solutions on faces where the presence of an intersection point is numerically unstable. This happens for example if the PEV line is parallel and very close to a face, or when the tensor field is almost zero.

In this paper, we only show examples of the PEV operator for (symmetric) stress tensor fields. Further possible scenarios that are left to future research are the comparative visualization of DT-MRI data or a comparative visualization of Jacobian fields for flow visualization, which are not necessarily symmetric.

Acknowledgements

This work was partially supported by DFG grant no. TH 692/8.

References

[BP02] BAUER D., PEIKERT R.: Vortex tracking in scale-space. In *Proc. VisSym* (2002), pp. 233–240. 2

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

[DH93] DELMARCELLE T., HESSELINK L.: Visualizing second-order tensor fields with hyperstreamlines. *IEEE CGA 13*, 4 (1993), 25–33. 2

[DH94] DELMARCELLE T., HESSELINK L.: The topology of symmetric, second-order tensor fields. In *Proc. IEEE VIS* (1994), pp. 140–147. 3

[ELLS11] EVERITT B. S., LANDAU S., LEESE M., STAHL D.: *Hierarchical Clustering*. Wiley-Blackwell, 2011, ch. 4, pp. 71–110. 5

[Far97] FARIN G.: *Curves and Surfaces for Computer Aided Geometric Design*, 4th ed. Academic Press, Boston, 1997. 4

[FPH*07] FUCHS R., PEIKERT R., HAUSER H., SADLO F., MUIGG. P.: Parallel vectors criteria for unsteady flow vortices. *IEEE TVCG 14* (12 2007), 615–626. 2

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

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

[HLL97] HESSELINK L., LEVY Y., LAVIN Y.: The topology of symmetric, second-order 3D tensor fields. *IEEE TVCG 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. 2

[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 TVCG 5*, 2 (1999), 135–144. 2

[Kin04] KINDLMANN G.: Superquadric tensor glyphs. In *Proc. VisSym* (2004), Eurographics Association, pp. 147–154. 2

[KW06] KINDLMANN G., WESTIN C.-F.: Diffusion tensor visualization with glyph packing. *IEEE TVCG 12*, 5 (2006). 2

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

[Ope] Openfoam: The open source CFD toolbox. http://www.openfoam.org. URL: www.openfoam.org. 6

[ORT18] OSTER T., RÖSSL C., THEISEL H.: Core lines in 3d second-order tensor fields. *CGF (Proc. EuroVis)* (2018), in press. 1, 2, 3, 4, 5

[POS*11] PAGOT C., OSMARI D., SADLO F., WEISKOPF D., ERTL T., COMBA J.: Efficient parallel vectors feature extraction from higher-order data. In *CGF (Proc. EuroVis)* (2011), vol. 30, pp. 751–760. 2

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

[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 TVCG 22*, 3 (2016), 1248–1260. 3

[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 VIS* (1998), pp. 143–150. 2

[Saa13] SAADA A. S.: *Elasticity: theory and applications*, vol. 16. Elsevier, 2013. 5

[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. 2

[SK10] SCHULTZ T., KINDLMANN G. L.: Superquadric glyphs for symmetric second-order tensors. *IEEE TVCG 16*, 6 (2010), 1595–1604. 2

[SML06] SCHROEDER W., MARTIN K., LORENSEN B.: *The Visualization Toolkit*, 4 ed. Kitware, 2006. 5

[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. *IEEE TVCG 14*, 6 (2008), 1627–1634. 3

[TS03] THEISEL H., SEIDEL H.-P.: Feature flow fields. In *Proc. VisSym* (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 VIS* (2005), pp. 631–638. 2

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

[WKL99] WEINSTEIN D., KINDLMANN G., LUNDBERG E.: Tensorlines: Advection-diffusion based propagation through diffusion tensor fields. In *Proc. IEEE VIS* (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 TVCG* (2010). 2

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

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

[ZPP05] ZHENG X., PARLETT B., PANG A.: Topological lines in 3D tensor fields and discriminant hessian factorization. *IEEE TVCG 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 TVCG 22*, 1 (2016), 797–806. 2

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