Tensor Field Visualization using Fiber Surfaces of Invariant Space

Felix Raith, Christian Blecha, Thomas Nagel, Francesco Parisio, Olaf Kolditz,
Fabian Günther, Markus Stommel, and Gerik Scheuermann

Abstract— Scientific visualization developed successful methods for scalar and vector fields. For tensor fields, however, effective, interactive visualizations are still missing despite progress over the last decades. We present a general approach for the generation of separating surfaces in symmetric, second-order, three-dimensional tensor fields. These surfaces are defined as fiber surfaces of the invariant space, i.e. as pre-images of surfaces in the range of a complete set of invariants. This approach leads to a generalization of the fiber surface algorithm by Klacansky et al. [16] to three dimensions in the range. This is due to the fact that the invariant space is three-dimensional for symmetric second-order tensors over a spatial domain. We present an algorithm for surface construction for simplicial grids in the domain and simplicial surfaces in the invariant space. We demonstrate our approach by applying it to stress fields from component design in mechanical engineering.

Index Terms—visualization, tensor field, invariants, fiber surface, interaction

1 Introduction

Symmetric second-order tensor fields are used in different disciplines like structural mechanics, fluid mechanics, geometry, or neuroscience describing important variables like stress, strain, deformation, metric, or diffusion. But their visualization is substantially less developed than the visual representation of scalar and vector fields. In practice, tensors are often reduced to scalar or vector fields prior to visualization. For showing the complete tensor, glyphs visualizing the value at isolated positions, and tensor lines following some eigenvector field are the most often used methods [17]. There is also a variety of methods derived from tensor lines like Hyperstreamlines [4], HyperLIC [34], tensor fabrics [7, 13], and tensor topology [5, 11] that are sometimes used. However, the intuitive, interactive analysis of three-dimensional symmetric tensor fields like a typical stress field in structural mechanics remains a challenge. Typically, the engineer is looking for an overview of the whole field first that highlights extremal tensor values. The failure of a constructed component or some natural structure is of highest interest in typical cases. Since all failure criteria are invariant functions, the engineer will concentrate on invariants before looking at directional aspects. This stands in strong contrast to all visualization methods derived from tensor lines which are essentially independent of the invariants (except for equal eigenvalues showing up as degenerate points). Consequently, this paper proposes an efficient method to provide an overview and interactive analysis of invariants of symmetric second-order tensor fields over three-dimensional domains, like the stress or strain fields of components in mechanical engineering.

The underlying idea is the three-dimensional nature of the invariant space. For a symmetric second-order tensor field over a three-dimensional domain, the space of the invariants is three-dimensional. This can be derived from the three real eigenvalues or the three invariants of the characteristic polynomial. Therefore, the mapping from the tensor field domain to the invariant space is a mapping from 3-space to 3-space, allowing for a direct visualization of the invariant space. Furthermore, it is possible to select a volume in the invariant space and show the pre-image in the domain, i.e. the parts of the object under study where the tensor field has invariants in the selected volume. It is also possible to compute the bounding surface of this pre-image which leads to the notion of a fiber surface of the invariant space. The method in this paper is therefore a generalization of the fiber surface approach from Klacansky et al. [16] to the case of a three-dimensional range. We show that this approach allows an exact, marching tetrahedron style algorithm under modest assumptions:

- A simplicial mesh in the three-dimensional domain
- Symmetric second-order tensors given at vertices
- Piecewise linear interpolation of tensor invariants
- Selection of volume in invariant space by simplicial mesh.

Of course, generalizations are possible at the cost of exactness and computational complexity. Also, we do not claim that the presented algorithm is the final word on efficiency, but rather provide a proof of concept that allows for nearly interactive visual analysis of stress fields from component design in mechanical engineering.

2 Related Work

While there is a lot of work on visualizing diffusion tensor images, we do not refer to this material here, as it is only relevant if it can be applied to indefinite tensors and does not have a strong relation to neuroscience. It should be noted, however, that our work might be applied in this context in the future. The survey of Kratz et al. [17] splits the current state of the art of tensor visualization beyond the positive definite case into three groups: the local, the continuous and the multi-view tensor visualization. Local methods mostly use glyphs, but the problem of design, placement and rendering comes along with them. Although glyphs are the only known method to represent all properties of a tensor, they should not be too complex. A popular method avoiding ambiguities depending on viewing directions are superquadrics by Kindlmann et al. [14]. An important limitation of local methods is the missing reflection of the continuity of the tensor field. Continuous methods always pick a part of the tensor information for visualization. Most continuous methods are built upon the tensor line idea from Dickinson [6], i.e. a streamline following one of the eigenvectors. Popular techniques are Hyperstreamlines of Delmarcelle and Hesselink [4], tensor topology [5, 11] and texture based approaches like HyperLIC of Zheng et al. [34], or fabrics [7, 13]. However, these approaches nearly ignore the invariant part as mentioned before. Finally, multi-view visualizations link the well-known visualizations with new ones to ease the interpretation and development of new visualization techniques as

- *Felix Raith and Christian Blecha share first authorship with equal contribution.*
- *Felix Raith, Christian Blecha, and Gerik Scheuermann are with Institute of Computer Science, Leipzig University, PF 100920, D-04009 Leipzig, Germany.*
- *Thomas Nagel, Francesco Parisio, and Olaf Kolditz are with Department of Environmental Informatics, Helmholtz Center for Environmental Research, Permoserstraße 15, 04318 Leipzig, Germany*
- *Fabian Günther and Markus Stommel are with Faculty for Mechanical Engineering, TU Dortmund University, Leonhard-Euler-Str. 5, 44227 Dortmund*

Manuscript received 31 Mar. 2018; accepted 1 Aug. 2018.
Date of publication 16 Aug. 2018; date of current version 21 Oct. 2018.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2018.2864846

well as the representation of at least one more property of a tensor. Multi-view visualization is not a recent idea in general as it is already used in texture mapping [28, 29], fluid simulation on surfaces [31], and additional topics but it is new in regard to the visualization of tensor fields. Our approach is a multi-view, in fact two-view, approach, but it computes explicit geometry in the domain providing non-local information, so it has properties of continuous methods as well. As it focuses exclusively on the invariants, it is complementary to most continuous techniques.

Only very few articles studied the visualization of symmetric tensor fields by looking at the invariant space. The two approaches closest to our work have been published recently by Palacios et al. [24] and Zobel and Scheuermann [35]. Both approaches define specific surfaces in the domain by properties of the invariants or the invariant mapping in the sense of a feature surface. Palacios et al. [24] define the surface of traceless tensors and the surface of tensors with mode zero (tensor mode will be defined in the next section). They also report that isosurfaces of tensor invariants like tensor mode, tensor isotropy, and tensor magnitude are helpful in engineering contexts. This was a source of inspiration for us. Zobel and Scheuermann [35] define so-called extreme points of the tensor field as points where the gradient of the invariant map has lower rank. This leads to extremal lines in two-dimensional domains and extremal surfaces in three-dimensional domains. In these earlier works, surfaces in the physical domain are defined based on mathematical properties in the invariant space. In contrast, we propose a general interactive mechanism to define surfaces in invariant space and visualize the pre-images of these surfaces in the physical domain. Thus, we allow for an interactive exploration of tensor fields based on invariants.

As we construct separating surfaces with explicit geometry from grid-based data by iterating over all cells, our approach is a Marching Cubes [21] style approach, compare also the marching tetrahedra variant [1]. Basically, if our defining surface in the invariant space is normal to one coordinate axis, we compute an isosurface of an invariant as suggested by Palacios et al. [24]. Indeed, we are convinced that our approach can also profit from the many improvements and acceleration techniques of marching cubes like Livnat's work [20].

As mentioned in abstract and introduction, our algorithm is a generalization of the work by Klacansky et al. [16] which builds upon the idea of fiber surfaces from Carr et al. [2]. In their case, the isovalue is an element of a two-dimensional co-domain which leads to the usage of polylines in the range for the definition of fiber surfaces in the domain. Basicly, we have a three-dimensional co-domain, the invariant space of the tensor field, which leads to using triangulated surfaces in the co-domain defining fiber surfaces in the domain of the tensor field. They use octree-based subdivision of the domain and pre-calculated lookup tables of all possible triangle characterizations in the spirit of marching cubes. We do not use this in our current implementation, but these ideas should lead to faster implementations in our case, too. But we follow the idea of a fast exact algorithm based on elementary geometric operations as suggested by Klacansky et al. [16] for their case.

Mechanical engineers are the main target users of our technique, even if neuroscientists or geometric designers may also profit. In mechanical engineering, the use of isosurfaces of tensor invariants is a daily business, as a look at the post-processing capabilities of widespread finite element software like Abaqus [32] shows. Zobel et al. [36] also comment about the necessity to visualize tensor gradients under certain conditions, going beyond the second-order case. A very nice application showing the power of advanced tensor visualization in mechanical engineering was given by Kratz et al. [18]. They improved the strength of a mechanical component, especially a brake lever, by visualizing tensor lines to support the design of rib structures. This led to a stiffer design with the same amount of material which could also be used to reduce material consumption under given loading. We hope for similar results by using our approach in the future.

3 Tensor Fields and Invariant Space

3.1 Symmetric second-order tensor fields

We consider a symmetric, second-order tensor field over a three-dimensional domain. Typical examples would be stress or strain fields in structural mechanics, the rate of deformation tensor in fluid dynamics, the metric tensor in geometry, or the diffusion tensor in diffusion tensor imaging (DTI) in neuroscience.

The set of all second-order tensors $\mathbb{R}^3 \otimes \mathbb{R}^3$ is nine-dimensional. For a fixed Cartesian coordinate system e^1, e^2, e^3, it has the orthonormal basis

$$A_{ij} := e^i \otimes e^j, \quad i,j = 1,2,3. \tag{1}$$

Similarly, the set of all symmetric tensors $\mathrm{Sym}(\mathbb{R}^3 \otimes \mathbb{R}^3)$ is six-dimensional and has the orthonormal basis

$$\begin{aligned} B^{11} &= e^1 \otimes e^1 && (2)\\ B^{12} &= (e^1 \otimes e^2 + e^2 \otimes e^1)/\sqrt{2} && (3)\\ B^{13} &= (e^1 \otimes e^3 + e^3 \otimes e^1)/\sqrt{2} && (4)\\ B^{22} &= e^2 \otimes e^2 && (5)\\ B^{23} &= (e^2 \otimes e^3 + e^3 \otimes e^2)/\sqrt{2} && (6)\\ B^{33} &= e^3 \otimes e^3 && (7) \end{aligned}$$

see e.g. Kindlmann et al. [15]. A symmetric tensor $T \in \mathrm{Sym}(\mathbb{R}^3 \otimes \mathbb{R}^3)$ can be written in this basis as

$$T = \sum_{j \le i} T_{ij} B^{ij}. \tag{8}$$

From now on, all tensors will be symmetric, second-order and three-dimensional. We will need some classical operations on tensors. The inner product of two tensors is written as

$$A : B := \sum_{ij} A_{ij} B_{ij}. \tag{9}$$

The norm of a tensor is defined by

$$|T| := \sqrt{T : T}. \tag{10}$$

Trace and determinant are given by

$$\begin{aligned} \mathrm{tr}(T) &= T_{11} + T_{22} + T_{33} && (11)\\ \det(T) &= T_{11}T_{22}T_{33} + T_{12}T_{23}T_{13} + T_{13}T_{12}T_{23} && (12)\\ &\quad -T_{13}^2 T_{22} - T_{23}^2 T_{11} - T_{12}^2 T_{33} \end{aligned}$$

We will also look at the deviator of T defined as

$$\mathrm{dev}(T) := T - \frac{1}{3}\mathrm{tr}(T) I \tag{13}$$

where $I = B^{11} + B^{22} + B^{33}$ is the identity tensor.

A symmetric, second-order tensor field over a three-dimensional domain $\mathbb{D} \subset \mathbb{R}^3$ is a map

$$\begin{aligned} \mathbf{T} : \mathbb{D} &\to \mathrm{Sym}(\mathbb{R}^3 \otimes \mathbb{R}^3) && (14)\\ x &\mapsto \mathbf{T}(x) \end{aligned}$$

3.2 Invariant Space and its Coordinate Systems

We are interested in the invariants of a symmetric second-order tensor T. An invariant of a symmetric tensor T is a scalar function

$$\beta : Sym(\mathbb{R}^3 \otimes \mathbb{R}^3) \to \mathbb{R} \tag{15}$$

that is invariant to the operations of rotations of $\mathbb{R}^3$, i.e. to operations of the special orthogonal group $SO(3)$ which means

$$\beta(T) = \beta(R T R^T) \quad \forall R \in SO(3). \tag{16}$$

The space $\mathbb{I}$ of all invariants of a three-dimensional, symmetric, second-order tensor is three-dimensional [30]. Therefore, for any symmetric second-order tensor field $\mathbf{T}$ over a domain $\mathbb{D} \subset \mathbb{R}^3$, we have a map

$$\begin{aligned} \mathbf{T}_{\mathbb{I}} : \mathbb{D} &\rightarrow \mathbb{I} \\ x &\mapsto \mathbf{T}_{\mathbb{I}}(x) \end{aligned} \tag{17}$$

We may select a basis $\beta^1, \beta^2, \beta^3$ of $\mathbb{I}$ and study the mapping of coefficients

$$\begin{aligned} \mathbf{T}_{\mathbb{I}} : \mathbb{R}^3 \supset \mathbb{D} &\rightarrow \mathbb{R}^3 \\ (x_1, x_2, x_3) &\mapsto \sum_{i=1}^{3} \beta_i(x) \beta^i \end{aligned} \tag{18}$$

with $x = \sum_{i=1}^{3} x_i e^i$. This is a map between three-dimensional spaces and will be used for visualization purposes.

Obviously, the choice of the basis is a key question for visualization and interpretation. Literature mentions several basis sets of practical relevance. Three basis sets can be derived from the characteristic equation

$$0 = \det(T - \lambda I) = \lambda^3 - I_1(T)\lambda^2 + I_2(T)\lambda + I_3(T) \tag{19}$$

Its coefficients are one popular basis set and are given by

$$I_1(T) := \operatorname{tr}(T) \tag{20}$$

$$\begin{aligned} I_2(T) &:= \operatorname{tr}(T)^2 - \operatorname{tr}(T^2) \\ &= T_{11}T_{22} + T_{11}T_{33} + T_{22}T_{33} - T_{12}^2 - T_{13}^2 - T_{23}^2 \end{aligned} \tag{21}$$

$$I_3(T) := \det(T). \tag{22}$$

Engineers like to study a somewhat different basis which makes explicit use of the deviator:

$$I_1(T) = \operatorname{tr}(T) \tag{23}$$

$$J_2(T) := \frac{1}{2}\operatorname{dev}(T) : \operatorname{dev}(T) \tag{24}$$

$$J_3(T) := I_3(\operatorname{dev}(T)) = \det(\operatorname{dev}(T)) \tag{25}$$

A third important set is given by the roots of the characteristic equation, i.e. the eigenvalues of T which are always real numbers in our case of symmetric tensors. We denote them by $\lambda_1(T), \lambda_2(T), \lambda_3(T)$.

For two more basis sets, we may use the R-invariants by Ennis and Kindlman [8]

$$R_1(T) = |T| \tag{26}$$

$$R_2(T) = \sqrt{\frac{3}{2}} \frac{|\operatorname{dev}(T)|}{|T|} \tag{27}$$

$$R_3(T) = \operatorname{mode}(\operatorname{dev}(T)) = 3\sqrt{6} det\left(\frac{\operatorname{dev}(T)}{|\operatorname{dev}(T)|}\right) \tag{28}$$

and the K-invariants from Criscioni et al. [3]

$$K_1(T) := \operatorname{tr}(T) = I_1(T) \tag{29}$$

$$K_2(T) := |\operatorname{dev}(T)| = 4\sqrt{J_2(T)} \tag{30}$$

$$K_3(T) := \operatorname{mode}(\operatorname{dev}(T)) = 3\sqrt{6}\frac{J_3(T)}{J_2(T)^{\frac{3}{2}}} \tag{31}$$

Ennis and Kindlmann [8] show that the I_i, λ_i, R_i and K_i are orthonormal bases (which was proven for the K_i by Crisconi et al. [3] before). Furthermore, the I_i and λ_i can be seen as Cartesian coordinates for the invariant space, while the K_i can be interpreted as cylindrical coordinate system, and the R_i as spherical coordinate system.

4 Stress Tensor Invariant Space

4.1 Stress Tensor Invariants

While symmetric second-order tensors over three-dimensional domains appear in many different application domains, e.g. diffusion tensor imaging in neuroscience, rate of deformation tensor in fluid mechanics, or metric tensors in differential geometry and geometric design, we focus here on structural mechanics as application domain. The presented technique can be applied in these other domains without changes, but to be a bit more explicit in the results section, we picked this domain as example. Also, there are several tensor fields in structural mechanics that are symmetric, second-order and three-dimensional, especially most stress and strain tensors. But, we concentrate here on the popular Cauchy stress tensor to demonstrate and evaluate the concept and leave the application to other fields for future work.

The Cauchy stress tensor $\sigma(x)$ describes the local stress $t(x,n) \in \mathbb{R}^3$ at position $x \in \mathbb{D}$ as force per unit area on any plane with normal $n \in \mathbb{R}^3$ by

$$t(x,n) = \sigma(x)n \tag{32}$$

where $t(x,n), x$ and n are given in Eulerian coordinates. This version of the stress tensor is usually computed and studied in finite element simulations. There are other stress tensors around. For a description of them and the relations among them, consult a typical engineering reference like Holzapfel's textbook [12].

Let us start with some well-known interpretations of stress tensor invariants in mechanical terms. The trace of the stress tensor

$$I_1(\sigma) = K_1(\sigma) = \operatorname{tr}(\sigma) \tag{33}$$

is closely related to the mean stress

$$p(\sigma) := \frac{1}{3}\operatorname{tr}(\sigma) = \frac{1}{3}I_1(\sigma) = \frac{1}{3}K_1(\sigma) = \frac{\lambda_1(\sigma) + \lambda_2(\sigma) + \lambda_3(\sigma)}{3} \tag{34}$$

which describes **hydrostatic stress**. If the deviator is zero, e.g. $J_2(\sigma) = J_3(\sigma) = 0$ resp. $K_2(\sigma) = K_3(\sigma) = 0$, it holds

$$\sigma = p(\sigma)I \tag{35}$$

with identity tensor I which means that we have equal stress (usually tension for positive sign and compression for negative sign) on all planes at this point. For an (elastic) fluid that cannot sustain any shear stress, p is called **pressure**, an important variable in fluid mechanics.

If the hydrostatic stress is ignored or not relevant for the question at hand, the deviator of the stress $\mathbf{dev}(\sigma)$ is studied. A fixed multiple of its norm

$$\sigma_v := \sqrt{\frac{3}{2}}|\operatorname{dev}(\sigma)| = \sqrt{\frac{3}{2}}K_2(\sigma) = \sqrt{24J_2(\sigma)} \tag{36}$$

is called **von Mises stress** [22] and is a popular yield criterion for important materials like steel (and many other metals) or some uniform plastics. It measures the strength of shear stress by looking at the squared sum of the differences of all eigenvalues. The underlying assumption is that the considered material starts plastic deformation under shear stress above a certain limit. This scalar limit can be measured for a material in an uniaxial test while it is applied in complicated triaxial stress states. If yielding depends on the isostatic pressure $I_1(\sigma) = K_1(\sigma)$ as well, or at the relation between all eigenvalues, i.e. it depends also on $K_3(\sigma), I_3(\sigma), J_3(\sigma)$, more complicated **yield functions** are used which are described shortly in the next section.

In general, the eigenvalues describe the normal stresses in the eigensystem of the stress tensor, i.e. local Cartesian coordinates where all shear stresses vanish. The smallest (minor) eigenvalue

$$\lambda_{\min}(\sigma) = \min(\lambda_1(\sigma), \lambda_2(\sigma), \lambda_3(\sigma)) \tag{37}$$

defines the maximal normal compression, and the largest (major) eigenvalue

$$\lambda_{\max}(\sigma) = \max(\lambda_1(\sigma), \lambda_2(\sigma), \lambda_3(\sigma)) \tag{38}$$

describes the maximal possible tension.

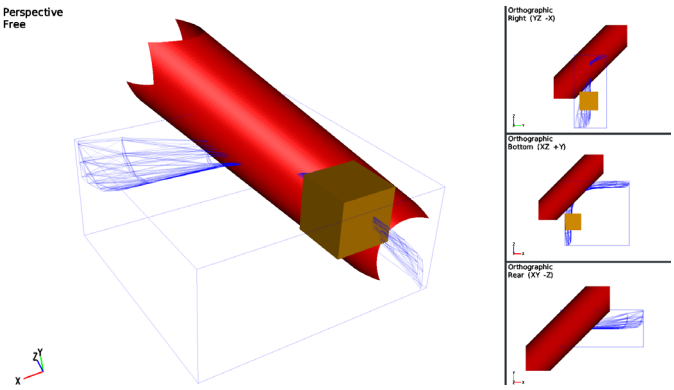

Fig. 1. Exemplary visualization of the eigenvalue-system of a bending beam with an additional von Mises surface. The cube is used to analyze the invariants which induce malfunction.

4.2 Yield Surfaces

In one typical structural mechanics application, there is a geometric design of a component together with loading conditions on some part of the boundary. The engineer has to check if the current design withstands the defined loading conditions. Before doing any physical tests, there will be a finite element simulation to check whether the component fails, i.e. where and under how much load it breaks.

For most materials, there is a prediction model for the stress states that will create plastic deformation instead of elastic deformation. Quite often, this stress is considered the limit to failure. This limit is given in these models as **yield surface**. Yield surfaces are usually described as implicit function of invariants of the stress tensor, i.e. there is a function

$$\begin{array}{rcl} f : \mathbb{I} & \rightarrow & \mathbb{R} \\ \sigma_{\mathbb{I}} & \mapsto & f(\sigma_{\mathbb{I}}) \end{array} \tag{39}$$

such that the yield surface is given by

$$f(\sigma_{\mathbb{I}}) = c \tag{40}$$

for some material-dependent constant $c \in \mathbb{R}$. In the simple case of a purely elastic simulation, the engineer is interested in all points with stresses outside the yield surfaces as the component will fail there (see Figure 1). He may also be interested in positions where the stress gets close to the surface as the model may still predict stability but the physical test may already show failure. We will discuss more practical questions in the results section.

The explicit formulae for the yield function are given in some basis of the invariant space (and can be expressed in the other bases by transformation, of course). We note a few popular cases. The **von Mises yield surface** is given by

$$\sqrt{\frac{3}{2}} K_2(\sigma) = \sigma_v \tag{41}$$

where σ_v is called **von Mises stress**. If visualized using the principal stresses as coordinates, it forms an infinite cylinder along the mean stress axis, i.e. $\lambda_1(\sigma) + \lambda_2(\sigma) + \lambda_3(\sigma)$ (see Figure 1). For a short explanation, see the previous section, some engineering book, or the original paper by von Mises [22].

The **Tresca yield surface** is given by

$$\begin{array}{rcl} \sigma_T & = & \lambda_{\max}(\sigma) - \lambda_{\min}(\sigma) \\ & = & \max(|\lambda_1(\sigma) - \lambda_2(\sigma)|, |\lambda_2(\sigma) - \lambda_3(\sigma)|, |\lambda_3(\sigma) - \lambda_1(\sigma)|) \end{array} \tag{42}$$

which means that there is a limit for the maximal shear stress. If visualized using the principal stresses as coordinates, it forms an infinite six-sided prism along the mean stress axis.

The **Drucker-Prager yield surface** combines hydrostatic stress and shear stress limits into

$$\beta K_1(\sigma) + K_2(\sigma) = \sigma_t \tag{43}$$

with the weight β between the two stress invariants and the scalar limit σ_t. It can be seen as a generalization of the von Mises criterion allowing for hydrostatic stress dependency. This model is often used for materials like concrete or rock with the assumption that there is a shear stress limit as in the von Mises models above, but it depends on the normal stress (typically tension). In the principle stress coordinate system, it looks like a cone with axis along the mean stress.

The **Mohr-Coulomb yield surface** adds a hydrostatic pressure dependency to the Tresca model leading to the formula

$$\begin{array}{rcl} \max(|\lambda_1(\sigma) - \lambda_2(\sigma)| + K(\lambda_1(\sigma) + \lambda_2(\sigma)), & & \\ |\lambda_2(\sigma) - \lambda_3(\sigma)| + K(\lambda_2(\sigma) + \lambda_3(\sigma)), & & \\ |\lambda_3(\sigma) - \lambda_1(\sigma)| + K(\lambda_3(\sigma) + \lambda_1(\sigma))) & = & \sigma_{MC} \end{array} \tag{44}$$

where $K = \frac{m-1}{m+1}$ with $m = \frac{\sigma_c}{\sigma_t}$, the maximal tensile stress σ_t, and the maximal compressive stress σ_c in the uniaxial test. It is also used for concrete and rock, like Drucker-Prager. In principle stress coordinates, the Mohr-Coulomb yield surface looks like a conical prism. There are many more yield surface models, including all types of quadratic surfaces (Burzyski-Yagn model) generalizing the examples above.

Of course, all these laws are approximate, so engineers will keep margins to be on safe side. Sometimes the chosen law may not be perfect, so a look at the whole stress state instead of the invariant helps. For purely elastic simulation, the model will compute stress states outside the yield surface, thus indicating failure. For elasto-plastic simulations, the stress will never be outside the yield surface, but stay there while plastic deformation takes place.

There are also indications for materials like metal that local failure does not only depend on stress state but also on the gradient. Basically, it is assumed that a high stress does not yield failure if there is also a high gradient indicating substantially lower stress in close surrounding. Again, engineers are interested in observing the gradient together with stress in such cases [36].

5 Overview

The idea of the paper is to develop an algorithm that extends the approach of Klacansky et al. [16] so that the three-dimensional invariant space of symmetric second-order tensors can be used as range instead of a two-dimensional one. The reason is the difficult interpretation of invariant spaces which can be eased by interactive visualization and direct linkage to the physical domain. Moreover, our method offers new approaches for the analysis of invariants and their relationships in real world applications. In order to generalize the work of Klacansky et al. [16], it is not sufficient to adapt the algorithm itself. The interaction in the range has to be redesigned as well, to deal with the characteristics of the three-dimensional space. This will be discussed in section 7. The invariant space hampers the generalization because it is in the nature of typical data that the geometric primitives (e.g. tetrahedra) interleave and infuse each other heavily which increases the challenge for effective representation of the data. The interaction will be handled by selection of regions of interest with geometrical primitives which are assumed to be closed and convex. The algorithm itself assumes a triangulated surface of the geometrical primitives as input, referred to as interactor.

As mentioned before, we assume that our tensor field is given over a tetrahedral grid in three-dimensional space with Cartesian coordinates and tensor values given at the vertices. Each vertex is mapped onto a vertex in the invariant space by the invariant map $\mathbf{T}_{\mathbb{I}}$, see equation 18. Due to the non-linear formulae of the invariants, a correct interpolation over the tetrahedra would lead to high computational load and an exact algorithm would be very difficult. Therefore, we assume a linear interpolation of the invariants over each tetrahedron, even if this is mathematically incorrect. Consequently, the invariant map pushes the structure of the grid in the domain to the co-domain. In other words,

the image in invariant space is a tetrahedral mesh too, and all neighbor relations of vertices are kept. However, there is heavy self-intersection of the mesh because of the same invariants which appear several times in the tensor field. Unlike the domain, the co-domain is displayed in Cartesian or in cylindrical coordinates depending on the selected basis, see section 3.2.

6 Fiber Surface Computation

6.1 Basic Approach

As mentioned in the previous section, our assumption of linear interpolation of the invariants maps each tetrahedron of the grid in the physical domain to a tetrahedron in invariant space. In invariant space, we assume that the selection of the volume of interest is done by an interactor which is given as triangulated surface. We define our fiber surface in invariant space as the intersection of the interactor with all tetrahedra. Therefore, we basically intersect each tetrahedron in invariant space with each triangle of the interactor. This leads to a large number of triangle-tetrahedron intersection computations.

To intersect a triangle with a tetrahedron, we compute the plane of the triangle and check if the vertices of the tetrahedron lie on different sides of the plane. If not, there cannot be an intersection. Otherwise, we compute the four faces of the tetrahedron and carry out simple triangle-triangle intersections using methods from literature [9, 23, 33]. The intersecting lines of the triangle-face pairs are calculated and its vertices are added to a list of all intersecting points. In addition, we check for the vertices of the triangle from the interactor if they are inside the tetrahedron. If they are inside, they are also added to the list. We order all intersection points into a polygon as they must lie in the plane of the interactor triangle and create triangles. In an additional step, we identify intersection points from different tetrahedra on the same interactor face. In this way, we get a triangulated surface in the invariant space.

After these intersection computations, we need to map the resulting triangulated surface back to physical space. Since we assume a linear mapping of invariants, the inverse mapping is also linear and we can use barycentric coordinates inside each tetrahedron to map the intersection back to physical space. The resulting triangulated surface is the fiber surface.

In addition, a post-process ensures that the set of triangles is split into (connected) contours which allows a simple visual analysis of connected components in the object space. A half-edge data structure is used to realign the normals of the triangles of the separate structures to identify the interior and exterior of a surface clearly. Furthermore, we index the triangles of the fiber surface by the interactor triangles. This allows for a visualization of the part of the fiber surface associated with a specific interactor triangle, e.g. by simple color coding. Algorithm 1 describes the general fiber surface extraction algorithm.

6.2 Special Cases

As usual with computational geometry algorithms, special cases need special attention. A first problem arises from numerical errors in computing the intersection points. This may lead to slightly different coordinates for the same point when computing it multiple times. This happens especially if the tetrahedra are very small so that even small round-off errors have a significant effect on the results (see Figure 2). Therefore, a numerically correct calculation of the intersection points must be used to avoid problems with connectivity or even visible holes in the resulting surface. A second challenge is provided by exactly planar tetrahedra. In these cases, there is no bijection between invariant space and physical domain. These tetrahedra are not considered by our algorithm at all because the intersections of the adjacent tetrahedra of the interleaving faces create the necessary geometry. We just connect the triangles correctly to get the right topology for our fiber surface.

6.3 Acceleration

Considering Figure 2, it is noticeable that the mapped tetrahedra vary substantially in size and shape in the co-domain which hampers a partition for fast extraction of fiber surfaces. Nevertheless, we need to reduce the number of triangle-tetrahedron intersection tests to enable an interactive exploration of datasets. This is done in two steps:

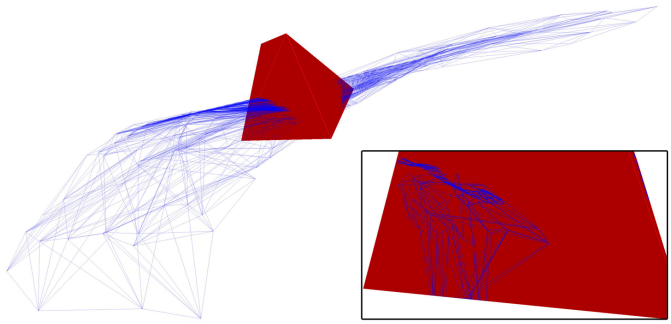

Fig. 2. An example of a tensor field visualized as wireframe in invariant space with an intersecting interactor (red). The computed intersections of one triangle of the interactor with the tetrahedra are emphasized.

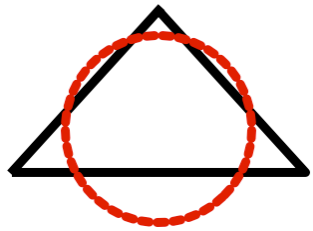

Fig. 3. Two-dimensional sketch of an interactor (red circle) and a tetrahedron (black triangle) which overlap and where none of the vertices of the tetrahedron lies inside the interactor.

1. Bounding spheres of the interactor and each tetrahedron are created and checked for intersection. If no intersection is found, the algorithm skips the current tetrahedron for any further calculations. The method for the computation of the bounding sphere of a tetrahedron is illustrated in Algorithm 2.

2. A signed distance field similar to marching cubes is applied to the remaining tetrahedra. Only if the vertices of a tetrahedron lie in different half spaces created by the plane through the current triangle, the current tetrahedron must be observed in the fiber surface extraction process.

The partition of all interactors into their triangles facilitates a later parallelization and introduces the ability to look at the different fiber surfaces which were extracted. The second reduction cannot be applied to the entire interactor because it is a closed body and the tetrahedron can infuse it without a vertex inside the tetrahedron (see Figure 3). For the sphere intersection test, the center M of the interactor is calculated using the arithmetic mean of all points of the triangulation of the interactor. After this, the radius R of the interactor is calculated using the center M and the farthest point. Next the bounding spheres of all tetrahedra are created by computing their centers m_i and radii r_i. The calculation of the sphere around a tetrahedron depends on the type of the tetrahedron. If the sphere through all four vertices has a center inside the tetrahedron, we use this sphere. If the center is outside, we look at the largest triangular face. We take the center of the circumference of this triangle and use its radius as radius of the sphere. The fourth vertex is inside this sphere in this case, so we get a smaller sphere than using the sphere through all vertices.

In addition, all tetrahedra whose radius equals zero are removed because they consist of only one point. For the remaining tetrahedra, we check the spheres for intersection. An intersection of a tetrahedron and a triangle is only possible if the distance between the centers m_i and M of their bounding spheres is smaller than the added radii $r_i + R$.

Algorithm 1 Calculation of a Fiber Surface

Require: Grid *OSG* in Object Space, Possible Intersecting Tetrahedra *PITet*, list of triangles *Tri* from interactor
for all triangles $tri \in Tri$ **do**
  for all tetrahedra $tet \in PITet$ **do**
    set case $in = false$, $out = false$
    for all vertices $v_i \in tet$ **do**
      compute position of vertex v_i relative to plane E across *tri*
      if $v_i < 0$ **then set** $in = true$
      else set $out = true$
      end if
    end for
    if $in = true$ **and** $out = true$ **then**
      add *tet* to *CoTet* ▷ List of Constrained Tetrahedron
    end if
  end for
  for all tetrahedra $tet \in CoTet$ **do**
    for all faces $f \in tet$ **do**
      compute intersecting points *IP* from f with *tri*
      add *IP* to *IsectPointsTet*
    end for
    for all points $P \in tri$ **do**
      if P inside *tet* **then** ▷ Test with linear combination
        add P to *IsectPointsTet*
      end if
    end for
    for all points $P \in IsectPointsTet$ **do**
      projection $P \rightarrow P'$ in *OSG*
      add P' to $IsectPointsTet'$
    end for
    triangulation of the *intersectionPlane* ▷ Created of $IsectPointsTet'$
    set the normals of the *triangulationPlane*
  end for
end for

Algorithm 2 Sphere Intersection Test

Require: grid *ISG* in invariant space, list of triangles *Tri* of the interactor
compute midpoint M from interactor
compute radius R from interactor
for all tetrahedra $tet_i \in ISG$ **do**
  compute midpoint m_i from tetrahedron tet_i
  compute radius r_i from tetrahedron tet_i
  compute midpoint distance $d_i = norm(m_1 - M)$
  remove tet_i with $r_i = 0$ ▷ Compute possible intersection from tet_i and the interactor
  if $d_i < (r_i + R)$ **then**
    add tet_i to *PITet* ▷ List of possible intersecting tetrahedra
  end if
end for

7 Interactive Visual Analysis

We aim at interactive visual analysis of a symmetric, second-order, three-dimensional tensor field by selecting surfaces in invariant space which define fiber surfaces in the domain. For this purpose, we present the user two linked views, one for the physical three-dimensional domain, and one for the invariant space. While there are several obvious choices for showing context in the physical domain, like boundary grid and global coordinate system, the presentation of the invariant space is less straightforward.

7.1 Coordinate System

For projecting to invariant space, we need to define coordinates in invariant space. In section 3.2, we presented five different coordinate systems. Even if we do not consider the R_i-System (a spherical coordinate system) for our mechanical applications, we have at least three Cartesian and one cylindrical system with clear semantics as explained in section 4. In principle, we leave the decision up to the user who may decide to look at their data in different coordinate systems. However, there is one issue about the I- and I,J-systems that requires care: scaling. From the definitions, it can be seen easily that trace is a linear, second invariant a quadratic, and determinant a cubic quantity in terms of tensor entries. Therefore, there will be most likely substantial differences in value for these coordinates. One solution is to take roots or powers to bring them into a similar range. another would be the scaling by the user. Overall, we discourage the use of these two coordinate systems because of this scaling issue. A short look at the K-basis shows that K_2 is basically the square root of J_2 which indicates that units fit well to K_1. Furthermore, K_3 is a ratio between well scaled J_3 and J_2, known to be in the interval $[-1,1]$. If one interprets K_3 as an angle by $\arcsin(K_3)$ in a cylindrical system, there is no scaling issue with the K-invariants. Also, there is also no scaling issue with the eigenvalues, i.e. λ-invariants.

As additional context, we offer the user to present one of the four yield surfaces described in section 4.2 with adjustable parameters to allow for additional context besides the coordinate system (see Figure 1). Of course, if a different yield surface fits the used material model better, we can easily add this surface as well to the system.

7.2 Invariant Space Image

After setting up the coordinate system, we need to show the image of the invariant map in our invariant space view. As mentioned before, we assume that the tensor values are given at the vertices of a tetrahedral mesh in the domain. We compute the invariant coordinates according to the chosen basis at the vertices. If two neighboring vertices have the same invariants or the Euclidean distance between them is smaller than a predefined ε, the two vertices are moved to the same position in invariant space. The edge between them is still stored to keep the grid topology. This procedure helps the triangle-tetrahedron intersection algorithm of the previous section to compute a valid inverse mapping. Nevertheless, we can still have planar or almost planar tetrahedra in invariant space which create numerical errors in the inverse mapping.

To map the tetrahedra, we assume the linear interpolation of invariants inside each tetrahedron despite mathematical errors, so that each tetrahedron in the domain maps to a tetrahedron in invariant space. This leaves us with the question how to show all those tetrahedra or more generally, the image of the invariant map. We have used four different possibilities in our tests:

Point Representation Our simplest visual representation in the invariant space is to show all vertex images as points resp. small spheres, see Figure 4A.

Wireframe Representation While the mesh topology is kept in invariant space, there is usually heavy overlapping of the tetrahedra in the invariant space. This is due to the fact that the same invariant set is reached at several positions in the data. Nevertheless, we frequently use a wireframe representation of the mesh in the invariant space. This puts visual emphasis on extreme values and outliers which are typically of interest for further inspection. It also gives some clue where many tetrahedra overlap, but without any indication of number or density, see Figure 4B.

Surface Representation We have also experimented with rendering all faces of the tetrahedra as transparent triangles. This allows for some indication of density (see Figure 4C), even if this is mathematically incorrect because the density inside the tetrahedra and the distance between front and back face are ignored.

Direct Volume Rendering of Density In principle, it is possible to compute an exact density distribution in the invariant space based on the mapped tetrahedra. This density can be visualized by direct volume rendering. However, due to the massively overlapping tetrahedra, it is a real challenge to implement an algorithm for this that allows for interactive rendering. Therefore, we leave this task for future work and

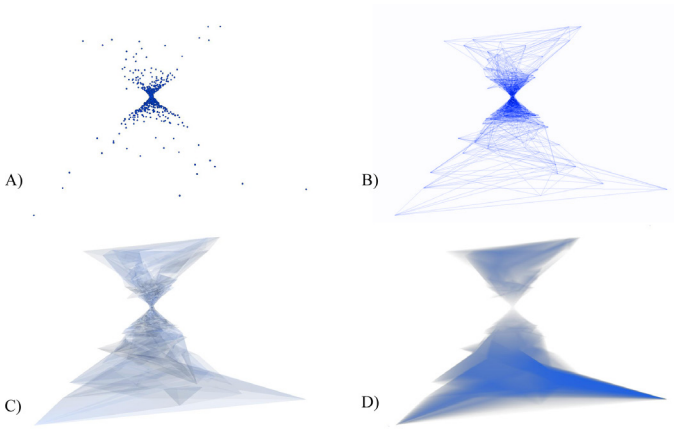

Fig. 4. Different representations of the mesh of tetrahedra in the three-dimensional invariant space: point representation of the vertices (A), wireframe representation (B), surface representation (C), and representation with direct volume rendering of the density (D).

use a rather coarse approximation. As there are fast volume rendering algorithms for voxel grids, we simply define such a grid with a size close to the maximum texture memory of the graphics card. In a preprocessing step, we compute the density for each voxel. For this, we iterate over all tetrahedra, compute the voxels intersected by the mapped tetrahedron in invariant space, and add up the corresponding density. This leads to a sampling of the correct density over the voxel grid. We use a standard raycasting algorithm implemented using shaders on the graphics card to render this density interactively. As transfer function, we use simply a constant blue value and linearly increasing opacity. For a white background, this results in a good indication of the density distribution of the image in invariant space. However, extreme values are invisible in the volume rendering due to low density. Here, visual attention is directed towards the areas of high density in the invariant space. More precisely, the image corresponds to the density projected along the current viewing direction, but after some interactive rotations, the viewer gets an idea of the density distribution, see Figure 4D.

The engineers liked the wireframe representation best, as it indicates extremal values and gives some indication of the neighboring values around those. Sometimes, they also asked for the simple point representation. The other two options were less interesting to them, but this may be due to the fact that they are more difficult to understand or interpret. Further research will be necessary to find the best solution.

7.3 Interaction

The user needs to position a triangular surface in the invariant space to define the fiber surface. As this should be possible in an interactive fashion, we decided to use a small set of predefined "interactors". These interactors are closed graphical primitives with triangulated surfaces that can be scaled, rotated, and moved in invariant space. By right mouse click, the current position is communicated to the fiber surface algorithm which calculates the fiber surface shown in the domain. Currently, we support the following interactor geometries: tetrahedra, cubes (see Figure 5, 6, 7), spheres (see Figure 9), regular cylinders, pyramids, and prisms. It is possible to use several interactors at the same time, each of them creating a fiber surface. They are selected via left mouse click. However, we kept the number of interactors and number of triangles per interactor small to get close to interactivity for medium sized models, so we have not tested many interactors or very complicated designs. Of course, there are many more possibilities to define the interaction. We expect more research about this in the future, especially together with faster implementations of the fiber surface algorithm.

As shown in Figure 1 a yield surface could improve the exploration. Assuming that the yield surface defines the maximal stress for the specific application, all tetrahedra in the invariant space which lie beyond the surface are an indicator for a collapse. An interactor could be used to select these tetrahedra to show the regions in the physical domain where the collapse could occur.

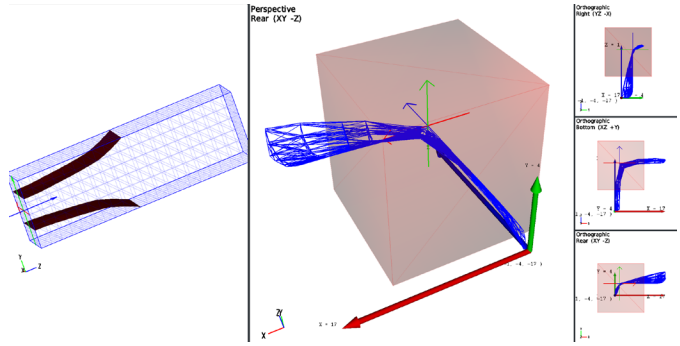

Fig. 5. Usage of a great cuboid interactor in the invariant space of a bending beam (right) and visualization of the extracted fiber surfaces inside the corresponding grid in the object space (left).

Inverse Fiber Surface Algorithm A very nice property of our algorithmic setup is that we can also calculate the forward mapping from the domain to the invariant space without a new algorithm or changes to the fiber surface algorithm. Our user interface allows using the interactors also in the physical domain defining a triangulated surface there. Now, the fiber surface algorithm computes the intersection of these triangles with the tetrahedra of the grid in the domain. Then, it maps the intersection, consisting of a few triangles per tetrahedron, to invariant space using the (forward) invariant map. The resulting triangular surface informs the user about the sub-set of invariant space that is reached by the selected part of the domain. The triangular surface in invariant space may be self-intersecting, but it serves its purpose well as can be seen in the results section.

8 Results

We demonstrate our method and its potential by applying it to three typical examples from mechanical engineering. The first case is a classical bending beam (cantilever). The cantilever is a well-known mechanical problem and therefore easily understood by engineers, so we use it to explain basic functionality. The second example looks at two loading simulations of a composite-metal hybrid component. In these simulations, the influence of composite defects on the strength of the component is investigated. Finally, the third example consists of a three-dimensional single-edge notched beam in a three-point bending configuration. The demonstrations were run on a standard workstation with an Intel Xeon E5-2630 v3 with 2.40 GHz, 32 GB Ram and a Nvidia GeForce GTX 980. Implemented is the algorithm inside our framework with C++.

8.1 Bending Beam

At first, we use a familiar engineering application, the cantilever, to show our approach in a familiar engineering application. The test dataset is a cantilever and has been generated as well simulated with Abaqus/Standard CAE [32]. The beam's dimensions are $30 \times 30 \times 100$ mm and it is completely fixed on one side (encastré) while on the loose side a force is introduced over the entire cross-section surface. The load is 1000 N in negative y-direction. The beam material is metal with a Youngs modulus of 210 GPa and the Poissons ratio of 0.3. The cantilever is meshed with three-dimensional quadratic tetrahedral continuum elements (C3D20) with a size of 5 mm.

In Figure 5 the beam is shown as the blue wireframe in the domain view on the left side. On the right side, the invariant space view is given where the three principal stresses are used as Cartesian coordinates. Here, the major principal stress is mapped at the x-axis, the medium principal stress at the y-axis and minor principal stress at the z-axis.

First, the symmetric load distribution in the bending beam can be seen in the obviously symmetric structure of the invariant space. The

structure is not easily interpreted but using the interactor, specific areas of interest either in invariant space or object space can be examined in more detail. For instance, this allows the definition of regions where significant high values or extreme differences of stress occur. In Figure 5, a cubic interactor (transparent red) is chosen with dimensions of $20 \times 20 \times 20$ MPa. The cube cuts the invariant space with two sides which are visualized as fiber surfaces. The corresponding fiber surfaces are rendered in the beam in the domain view on the left side. The upper surface shows a specific tensile stress while the second surface is a specific compression load in the lower part of the beam. This image is expected because of the applied bending force at the opposite end of the beam. Equally, the neutral phase of the bending beam could be pictured lying parallel in the middle of the beam. We could also show the maximum tensile stress in the upper region or the high compression in the lower region next to the fixed cross-section surface.

The extraction of the fiber surfaces of this small dataset (4320 tetrahedra) with an interactor of 12 triangles takes up to 1.0 second beginning with the translation of the interactor and ending with the rendering of the fiber surface. In this case the interactor is too great so that the algorithm 2 would not remove any tetrahedron.

8.2 Fiber reinforced metal component

A more interesting example represents a fiber reinforced plastic part. Due to the more complex material of multiple components (metal, fibers and matrix) the material behavior is more difficult to describe and to understand. Here, the well-known von Mises stress (σ_v, see section 4) is not a feasible yield criterion due to the anisotropic material parameters.

Our simulation uses a multi-continuum material model by Abaqus [32] and Helius [10] where the polymer matrix and the fibers are separately considered which is shown in Figure 6. The model represents a metal-composite connection part with metal inset (three arm structure). Because of the symmetry towards the y-z-plane, only half of the part is simulated. Therefore, symmetry boundary conditions are applied to the left side while the right and top side are fixed and a tensile load of 3000 N is applied at the bottom of the inset in negative y-direction. Due to the boundary conditions, the part geometry, and the material behavior, the invariant space is much more complex as it can be seen on the right of Figure 6. Again, we decided to use the principle stresses as coordinates after testing other options. The linked view interface provides two options to analyze the simulation results. On the one hand, significant peaks in the invariant space can be selected with the interactor and the corresponding regions in the object space are highlighted. Hence, the engineer can evaluate how critical the superposition of different invariants resp. stresses effect special regions in the part. On the other hand, the interactor can be placed in a region in the part and the corresponding invariant space will be highlighted. This can be useful to compare simulations with similar part geometries but alternating fiber orientations or defects.

The interactor encloses a significant area with high invariant density showing a combination of high values in the first and second principal stress, as can be seen in Figure 6. The corresponding region in the object space is a half round region above the metal insert where damages due to peeling normal forces in the part can be typically observed. This clearly indicates that the invariant space has to be investigated more to define critical areas and gain a better understanding for engineering applications like this one. The algorithm introduced in this article provides the base to do so in the future.

We have also used the inverse fiber surface algorithm (forward mapping) of section 7.3 in this example, as shown in Figure 7. Here, two simulations with the same model geometry are compared. The simulations differ because one contains an initial defect within the selected range of the interactor. It can be seen that the enclosed invariant space regions are slightly different. Due to the fact that the invariant space in this region does not differ significantly, it can be deduced that the effect of the inserted defect is not critical to the investigated composite part under static tensile load.

This approach allows investigating regions of special interest in an unusual and particular way. In general, the stress tensor invariant space

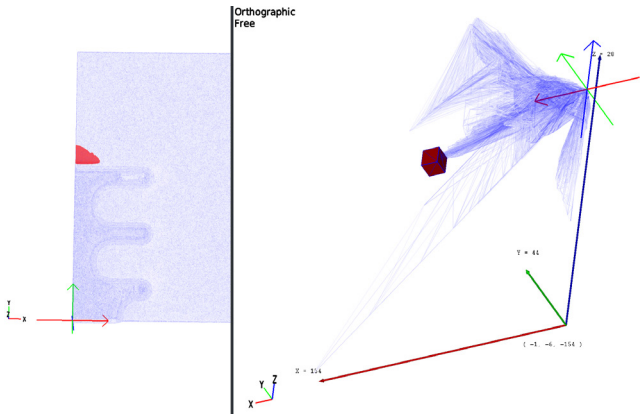

Fig. 6. Interaction with the invariant space of a fiber reinforced metal component (right) and visualization of the extracted fiber surfaces in the object space (left).

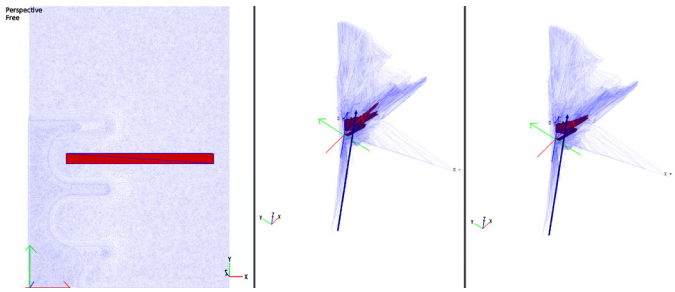

Fig. 7. Interaction with the object space of a fiber reinforced metal component (left) and visualization of fiber surfaces of the selected area with brought in defect (mid) and without brought in defect in the invariant space (right).

provides great potential to visualize special phenomena in mechanical engineering. E.g. it is conceivable that not just invariant sets like the principal stresses or the other base sets from section 3.2 may be used, but further values derived from the stress tensor and invariants can be displayed. Therefore, adapting the invariant space to specific questions in simulations depending on materials and part geometries offers great development and application potential for the proposed algorithm.

The extraction of the fiber surfaces of this large dataset (about 2.5 million tetrahedra) with an interactor of 12 triangles took up to 3.0 seconds. The sphere intersection test acceleration removes almost 99 % of all the tetrahedra in the dataset. Depending on the position of the interactor the set of possible tetrahedra is reduced to around 50000 in Figure 6 and to around 150000 in Figure 7.

8.3 Three-point bending of a single-edge notched beam

As last example, we present the application of our method to a solid mechanics case consisting of a three-dimensional single-edge notched beam in a three-point bending configuration. The finite element software solves the discretized weak form of the partial differential equation

$$\operatorname{div} \sigma = \mathbf{0} \tag{45}$$

subjected to suitable displacement and traction boundary conditions. The constitutive model linking stresses to strains is a coupled plastic-damage model with integral regularization, see Parisio et al. [25], for which the strength envelope consists of a Drucker-Prager yield surface, compare section 4.2 of the form

$$f^{\mathrm{p}}(\tilde{\sigma}) = \frac{1}{4} K_2(\tilde{\sigma}) - \beta K_1(\tilde{\sigma}) + k = 0. \tag{46}$$

In this case, $\tilde{\sigma} = \sigma / (1-d)$ is called the damage effective stress tensor, i.e., the stress acting in the undamaged part of the material (for further

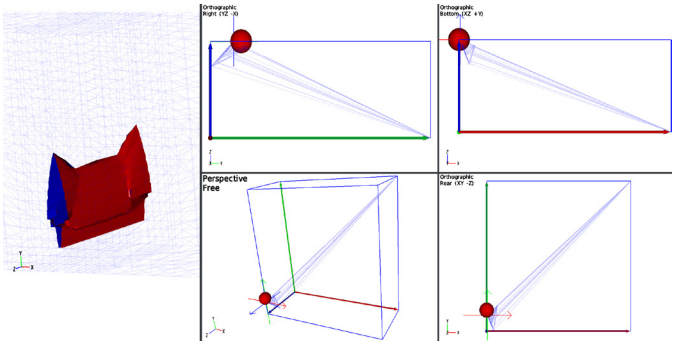

Fig. 8. Representation of a crack tip inside a notched beam (left). The exterior of the fiber surface is red and the interior blue. Representation of the eigenvector space (x shows the major, y the intermediate and z the minor eigenvalue) on the right side.

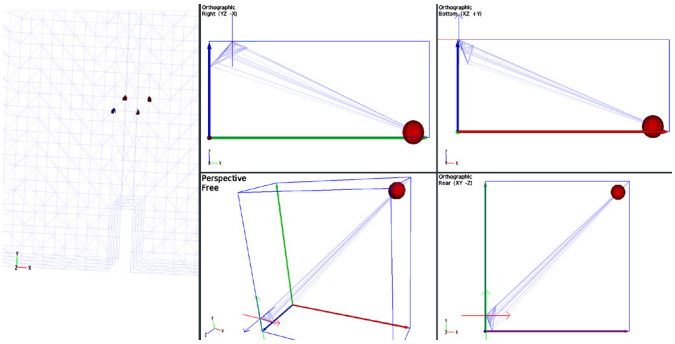

Fig. 9. Selection of the extreme point inside the eigenvalue space of Figure 8 (right) which corresponds to some small fiber surfaces (left) which is an indication of a shortcoming inside the used dataset.

details, see [19, 26, 27]). As noted in section 3.2, the strength envelope looks like a circular cone with its axis directed along the mean stress in invariant space with principal effective stresses as coordinates. The evolution of damage d is indicative of the progressive failure of the material.

Figure 8 shows the state of stress in terms of principal components. It can be seen that the vicinity of the crack is predominantly in a state of tensile stress (mode I failure). The red surface indicates indeed the crack envelope in the beam mesh, while it appears that the surface presents some extreme points of very high compressive damage effective stress. On a closer analysis, it was evidenced that the points are located just at the crack tip and at the outer boundaries of the model (Figure 9). Here, equilibrium imposes total stress to be null, while high values of damage might bring the effective stress to a very high value. Furthermore, the stresses are evaluated at Gauss points and later on extrapolated to obtain nodal values for post-processing: this procedure inevitably produces errors, and inaccuracies, which could also be a concurrent cause of this outlier. Figure 9 evidences the fact that it is a single point (showing up four times due to symmetries) with a very high stress value. In this case, the visualization in invariant space highlighted a shortcoming which could have hardly been noticed with conventional visualization methods. It furthermore points to the fact that possible improvements could be achieved by a higher mesh resolution.

The extraction of fiber surfaces of this dataset (about 16000 tetrahedra) with an interactor of 180 triangles took up to 5.0 seconds. The interactor is in Figure 8 in the center from all tetrahedra so the algorithm 2 removes nothing. Around 100 tetrahedra remain in Figure 9 after the algorithm.

9 Conclusion and Future Work

In this article, we introduce the notion of a fiber surface of invariant space for symmetric, second-order, three-dimensional tensor fields. This allows for a (nearly) interactive study of such tensor fields. For this purpose, we suggest an interface with two linked views, one showing the physical domain of the field, and the other showing the image of the invariant map in the invariant space. We discussed possible coordinate choices for the invariant space in a mechanical engineering context, as well as additional contexts like yield surfaces. We suggest the use of so-called interactors, simple geometric primitives represented as triangulated surfaces to select the range which is then pulled back to the domain defining the fiber surfaces.

For the computation of the fiber surfaces, we assume a tetrahedral grid in the domain with tensor values on the vertices. Furthermore, we assume piecewise linear interpolation of invariants in invariant space. This leads to an algorithm that basically intersects each interactor triangle with each grid tetrahedron in invariant space, and maps the result back to the domain. We demonstrate that even our proof of concept implementation allows for nearly interactive extraction of the fiber surfaces in realistic examples. Interestingly, the same algorithm allows for forward mapping of a triangulated surface in the domain to invariant space, so we obtain a linkage in both directions without additional costs.

We presented three real applications from mechanical engineering looking at the Cauchy stress tensor field. We showed that it helps to find and understand extreme values, and allows for sanity tests of stress values including interpolation errors. Especially, it allows separating extreme values in one invariant by values in other invariants.

As indicated at various points throughout the paper, there is a lot of potential for future work: There are several obvious possibilities to speed up the algorithm, e.g. by using a lookup table like marching tetrahedra, by additional tests to avoid unnecessary intersections, or by parallelization. Furthermore, we would like to see research on the best visual representation of the image in invariant space. This may go hand in hand with a search for an interactive direct volume rendering of density in invariant space. There is also a need for an evaluation of interaction mechanisms to define the surfaces in invariant space, e.g. to go beyond our interactors. It may also be worthwhile to look at algorithms for other grid types, and at the errors by our linear interpolation of invariants. Finally, there is definitely a need for more applications, including work outside of mechanics.

Acknowledgments

This work was funded by the German Federal Ministry of Education and Research within the project *Competence Center for Scalable Data Services and Solutions* (ScaDS) Dresden/Leipzig (BMBF 01IS14014B).

References

[1] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341 – 355, 1988. doi: 10.1016/0167-8396(88)90013-1

[2] H. Carr, Z. Geng, J. Tierny, A. Chattopadhyay, and A. Knoll. Fiber surfaces: Generalizing isosurfaces to bivariate data. *Computer Graphics Forum*, 34(3):241–250, 2015. doi: 10.1111/cgf.12636

[3] J. C. Criscione, J. D. Humphrey, A. S. Douglas, and W. C. Hunter. An invariant basis for natural strain which yields orthogonal stress response terms in isotropic hyperelasticity. *Journal of the Mechanics and Physics of Solids*, 48(12):2445–2465, 2000.

[4] T. Delmarcelle and L. Hesselink. Visualizing second-order tensor fields with hyperstreamlines. *IEEE Computer Graphics and Applications*, 13(4):25–33, July 1993. doi: 10.1109/38.219447

[5] T. Delmarcelle and L. Hesselink. The topology of symmetric, second-order tensor fields. In *Proceedings of the conference on Visualization'94*, pp. 140–147. IEEE Computer Society Press, 1994.

[6] R. R. Dickinson. A unified approach to the design of visualization software for the analysis of field problems. In *Three-dimensional visualization and display technologies*, vol. 1083, pp. 173–181. International Society for Optics and Photonics, 1989.

[7] S. Eichelbaum, M. Hlawitschka, B. Hamann, and G. Scheuermann. Fabric-like visualization of tensor field data on arbitrary surfaces in image space. In *New Developments in the Visualization and Processing of Tensor Fields*, pp. 71–92. Springer, 2012.

[8] D. B. Ennis and G. Kindlmann. Orthogonal tensor invariants and the analysis of diffusion tensor magnetic resonance images. *Magnetic resonance in medicine*, 55(1):136–146, 2006.
[9] C. Ericson. *Real-time collision detection*. CRC Press, 2004.
[10] A. GmbH. Helius composite. Available: https://www.autodesk.de/products/helius-composite/overview Accessed on: Feb. 08 2018.
[11] C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. De Floriani, G. Scheuermann, H. Hagen, and C. Garth. A survey of topology-based methods in visualization. In *Computer Graphics Forum*, vol. 35, pp. 643–667. Wiley Online Library, 2016.
[12] A. G. Holzapfel. Nonlinear solid mechanics ii. 2000.
[13] I. Hotz, L. Feng, B. Hamann, and K. Joy. Tensor-fields visualization using a fabric-like texture applied to arbitrary two-dimensional surfaces. In *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, pp. 139–155. Springer, 2009.
[14] G. Kindlmann. Superquadric tensor glyphs. In *Proceedings of the Sixth Joint Eurographics - IEEE TCVG Conference on Visualization*, VISSYM'04, pp. 147–154. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2004. doi: 10.2312/VisSym/VisSym04/147-154
[15] G. Kindlmann, D. B. Ennis, R. T. Whitaker, and C.-F. Westin. Diffusion tensor analysis with invariant gradients and rotation tangents. *IEEE Transactions on Medical Imaging*, 26(11):1483–1499, 2007.
[16] P. Klacansky, J. Tierny, H. Carr, and Z. Geng. Fast and exact fiber surfaces for tetrahedral meshes. *IEEE Transactions on Visualization and Computer Graphics*, 23(7):1782–1795, July 2017. doi: 10.1109/TVCG.2016.2570215
[17] A. Kratz, C. Auer, M. Stommel, and I. Hotz. Visualization and analysis of secondorder tensors: Moving beyond the symmetric positivedefinite case. *Computer Graphics Forum*, 32(1):49–74, 2013. doi: 10.1111/j.1467-8659.2012.03231.x
[18] A. Kratz, M. Schoeneich, V. Zobel, B. Burgeth, G. Scheuermann, I. Hotz, and M. Stommel. Tensor visualization driven mechanical component design. In *Visualization Symposium (PacificVis), 2014 IEEE Pacific*, pp. 145–152. IEEE, 2014.
[19] J. Lemaitre, J.-L. Chaboche, A. Benallal, and R. Desmorat. *Mécanique des matériaux solides-3eme édition*. Dunod, 2009.
[20] Y. Livnat, H.-W. Shen, and C. R. Johnson. A near optimal isosurface extraction algorithm using the span space. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):73–84, Mar 1996. doi: 10.1109/2945.489388
[21] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH Computer Graphics*, 21(4):163–169, Aug. 1987. doi: 10.1145/37402.37422
[22] R. v. Mises. Mechanics of solid bodies in the plastically-deformable state. *Göttin Nachr Math Phys*, 1:582–592, 1913.
[23] T. Möller. A fast triangle-triangle intersection test. *Journal of graphics tools*, 2(2):25–30, 1997.
[24] J. Palacios, H. Yeh, W. Wang, Y. Zhang, R. S. Laramee, R. Sharma, T. Schultz, and E. Zhang. Feature surfaces in symmetric tensor fields based on eigenvalue manifold. *IEEE transactions on visualization and computer graphics*, 22(3):1248–1260, 2016.
[25] Parisio, Tarokh, Makhnenko, Naumov, Miao, Kolditz, and Nagel. Experimental characterization and numerical modelling of fracture processes in granite. international journal of solids and structures (under review). 2018.
[26] F. Parisio and L. Laloui. Plastic-damage modeling of saturated quasi-brittle shales. *International Journal of Rock Mechanics and Mining Sciences*, 93:295–306, 2017.
[27] F. Parisio, S. Samat, and L. Laloui. Constitutive analysis of shale: a coupled damage plasticity approach. *International Journal of Solids and Structures*, 75:88–98, 2015.
[28] E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pp. 465–470. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000. doi: 10.1145/344779.344987
[29] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pp. 409–416. ACM, New York, NY, USA, 2001. doi: 10.1145/383259.383307
[30] A. J. M. Spencer. *Continuum mechanics*. Courier Corporation, 2004.
[31] J. Stam. Flows on surfaces of arbitrary topology. *ACM Trans. Graph.*, 22(3):724–731, July 2003. doi: 10.1145/882262.882338
[32] D. Systemes. Abaqus/cae. Available: https://www.3ds.com/de/produkte-und-services/simulia/produkte/abaqus/ Accessed on: Feb. 08 2018.
[33] G. Van Den Bergen. *Collision detection in interactive 3D environments*. CRC Press, 2003.
[34] X. Zheng and A. Pang. Hyperlic. In *IEEE Visualization, 2003. VIS 2003.*, pp. 249–256, Oct 2003. doi: 10.1109/VISUAL.2003.1250379
[35] V. Zobel and G. Scheuermann. Extremal curves and surfaces in symmetric tensor fields. *The Visual Computer*, pp. 1–16, 2017.
[36] V. Zobel, M. Stommel, and G. Scheuermann. Visualizing gradients of stress tensor fields. In *Modeling, Analysis, and Visualization of Anisotropy*, pp. 65–81. Springer, 2017.