DOI: 10.1111/cgf.13983
Eurographics Conference on Visualization (EuroVis) 2020
M. Gleicher, T. Landesberger von Antburg, and I. Viola
(Guest Editors)

Volume 39 (2020), Number 3

Fiber Surfaces for many Variables

C. Blecha[1], F. Raith[1], A. J. Präger[1], T. Nagel[2], O. Kolditz[3], J. Maßmann[4], N. Röber[5], M. Böttinger[5], and G. Scheuermann[1]

[1] Institute of Computer Science, Leipzig University, Leipzig, Germany
[2] Chair of Soil Mechanics and Foundation Engineering, Geotechnical Institute, Technische Universität Bergakademie Freiberg, Freiberg, Germany
[3] Department of Environmental Informatics, Helmholtz Center for Environmental Research, Leipzig, Germany
[4] Federal Institute for Geosciences and Natural Resources (BGR), Hanover, Germany
[5] Deutsches Klimarechenzentrum, Hamburg, Germany

Abstract

*Scientific visualization deals with increasingly complex data consisting of multiple fields. Typical disciplines generating multivariate data are fluid dynamics, structural mechanics, geology, bioengineering, and climate research. Quite often, scientists are interested in the relation between some of these variables. A popular visualization technique for a single scalar field is the extraction and rendering of isosurfaces. With this technique, the domain can be split into two parts, i.e. a volume with higher values and one with lower values than the selected isovalue. Fiber surfaces generalize this concept to two or three scalar variables up to now. This article extends the notion further to potentially any finite number of scalar fields. We generalize the fiber surface extraction algorithm of Raith et al. [RBN*19] from 3 to d dimensions and demonstrate the technique using two examples from geology and climate research. The first application concerns a generic model of a nuclear waste repository and the second one an atmospheric simulation over central Europe. Both require complex simulations which involve multiple physical processes. In both cases, the new extended fiber surfaces helps us finding regions of interest like the nuclear waste repository or the power supply of a storm due to their characteristic properties.*

1. Introduction

Isosurfaces are some of the most frequently used tools in scientific visualization. Together with direct volume rendering, they are probably the most popular basic visualization technique for scalar field data over three-dimensional domains. An important property is their ability to split the domain into two parts, one with values higher than the isovalue, and one with values lower than the isovalue. Another important property is that isosurfaces are described as surfaces, so they are often used to obtain surfaces which are processed further. Also, they are easily understood by scientists and engineers.

Unfortunately, isosurfaces work only for a single scalar variable. The first generalization was given by Carr et al. [CGT*15], when they defined fiber surfaces. Two scalar fields define a two-dimensional range or attribute space. If one selects a polygon in this attribute space, the preimage is a surface with properties similar to an isosurface, like the separation of parts in the domain which lie in the inside or outside of the polygon. This concept has been generalized even further by Raith et al. [RBN*19], where they defined fiber surfaces for the three scalar invariants of symmetric, second-order tensor fields. As can be seen in the paper by Blecha et al. [BRS*19], this algorithm can also be applied to three arbitrary scalar fields over a three-dimensional domain. This raises the natural question if the number of dimensions of the domain limits the number of scalar fields that can be used to define fiber surfaces in a meaningful way. Many highly relevant application domains of scientific visualization, like climate research, geology, material science, or bioengineering, deal regularly with more than three scalar fields. Scientists also want to see the relation between these fields and ask questions, where certain constraints on these variables are fulfilled.

This article shows that fiber surfaces can be defined in a natural way. We also present an algorithm to compute a fiber surface for more than three scalar variables. In principle, the algorithm can deal with any number of scalar fields, but we show only results for up to six variables. The new algorithm assumes that we are in a piecewise linear setting, despite the fact that all definitions can be given in a continuous way. Therefore, we assume that d scalar fields are given over some tetrahedral grid in our three-dimensional physical domain. Furthermore, we assume that there is a piecewise linear hypersurface in the d-dimensional attribute space. The preimage of this object defines the fiber surface on the given tetrahedral grid.

The algorithm considers the tetrahedral grid as a set of tetrahedra in the d-dimensional attribute space by using the values of the scalar fields at each vertex as coordinates. A convex object, hereafter called interactor, is defined as the subspace which is constrained through the intersection of several hyperplanes defined using a d-dimensional normal and a distance to some point. The algorithm clips each tetrahedron against each hyperplane in the attribute space. The result is split into tetrahedra which will be clipped with the remaining hyperplanes. The key idea is that the mapped tetrahedron defines a three-dimensional affine subspace of the attribute space. Therefore, we can project the hyperplane into this subspace. This means that we are actually computing the intersection between a plane and a tetrahedron in three dimensions. The algorithm computes this intersection and refines the tetrahedron into a set of new tetrahedra such that the actual intersection is a set of faces of the new tetrahedra. In the end, we simply project all intersection faces back to the physical space and render the resulting triangles. As one can see, the number of scalar fields is only important for the intersection of a hypersurface and the subspace and for storing coordinates, so it does not substantially affect the overall complexity.

However, a working interactive system needs a mechanism to define the hypersurface in an efficient way without confusing users. Since an intuitive visualization of more than three dimensions in a single image is difficult, we use several three-dimensional views for this purpose. In our current implementation, users can select triplets of scalar variables and look at the corresponding three-dimensional projection of the attribute space. Here, they can define the hypersurface with respect to these three variables. We consider these as constraints and allow to add additional constraints in other such views of three variables. We allow simple geometric objects, like planes, cubes, and triangulated balls as selection objects. The actual constraint is the intersection of all these objects in attribute space. In the end, we use data from a simulation of a generic nuclear waste repository, which was also described in the work of Blecha et al. [BRS*19], and an atmospheric simulation to demonstrate our algorithm using up to six scalar fields.

2. Related Work

The method presented in this paper can be classified as multivariate visualization technique. Fuchs and Hauser [FH09] wrote a survey about existing visualization techniques for multivariate scientific data in 2009. They showed the advantages of existing visualization techniques if they were applied to multivariate data and identified the most important approaches for dealing with multivariate scientific data. A further overview of research challenges, and the state of the art in visualization was presented at a Dagstuhl Seminar in 2011 [DHCRJ*14]. The visualization of multivariate data was mentioned as one of the challenges at this seminar. They presented, for example, techniques like glyphs [CLKH14], which are a default technique for the visualization of multivariate data. Furthermore, the importance of feature-based techniques, like the work of Obermaier et al. [OP14], and Carr [Car14], was clarified.

Sauber et al. [STS06] introduced multifield graphs to visualize the relationships between multiple scalar fields defined on the same domain. The standard approach during correlation analysis is to calculate correlation fields for multiple combinations of scalar fields. This leads to a high number of computed fields, whereby not every combination exhibits some interesting or important correlations. They used their graph to give an overview of this large number of correlation fields. Nagaraj et al. [NNN11] improved this approach by introducing a comparative measurement to reduce the overrepresentation of the minimum correlation between two scalar fields from Sauber et al. [STS06]. Liu and Shen [LS16] followed a different approach using association rules to generate a parallel coordinate diagram for the analysis of multivariate data sets. This diagram can be used to analyze isovalues and the associated values in other fields but in the end, they are only visualizing the relationships between multiple scalar fields.
The work of Jänicke et al. [JWSK07] outlines another method for analyzing multivariate data. In that work, interesting regions are automatically identified using local statistical complexity described by partial differential equations. Nagaraj et al. [NN11] introduced the idea that the interactions between the fields have to be considered in order to find interesting isovalues of a field. In general, they investigate the variation of the remaining fields over the isosurfaces of a selected field.

Additional analysis techniques for multivariate data were introduced by Edelsbrunner et al. in 2004 and 2008. They calculated the Jacobi sets of collections of piecewise linear continuous functions on a common triangulated manifold [EH04], and they generalized Reeb spaces to multivariate, piecewiese linear mappings on combinatorial manifolds [EHP08].

Despite much work in the domain of field data visualization, most techniques refer to univariate data. The investigation of multi- or at least bivariate data sets is still a challenging research topic. Carr et al. [CGT*15] made a significant contribution by introducing an approach to the extraction of fiber surfaces. These are the two-dimensional equivalent to isosurfaces of a bivariate field. In general, fiber surfaces are the inverse image of a polygon (or polyline) in the bivariate field. Based on this work, Klacansky et al. [KTCG17] proposed an optimized implementation of their algorithm to allow interactive exploration of bivariate fields. Wu et al. [WKI*17] achieved an interactive data analysis using fiber surfaces by extending the normal ray casting method to visualize fiber surfaces of bivariate data. Futhermore, such fiber surfaces are implemented in the Topology ToolKit (TTK) [TFL*18] which increases their accessibility and usability. The TTK was introduced to enable easy, general, and time-efficient topology data analysis through the deployment of a framework for the coding of such analysis algorithms. Fiber surfaces were extended to flexible fiber surfaces by Sakurai et al. [SOC*19]. Analogously to flexible isosurfaces, they used the contour tree to track and show only some components of a fiber surface to reduce visual occlusion. Tierny and Carr [TC17] used fiber surfaces in their algorithm for an efficient computation of the Reeb space of bivariate functions on a tetrahedral mesh. In particular they introduced jacobi fiber surfaces which are analog to critical contours of scalar fields.

Another approach to extend the concept of isolines and isosurfaces was published by Jankowai and Hotz [JH20]. They introduced feature-level sets and traits as general approach for the visualization of multivariate data. Isosurfaces and the fiber surfaces of Carr et al. [CGT*15] are special cases of their approach. In their work, users select some interesting regions (*traits*) to analyze. The corresponding feature-level-sets are extracted and visualized in the physical domain using techniques like direct volume rendering or Marching Cubes. Traits are defined as geometric objects in the attribute space and features as the preimage of a trait in the spatial domain. In contrast to the interactive definition of feature-level-sets of Jankowai and Hotz [JH20] through star plots or parallel coordinates, we define our fiber surfaces using multiple three-dimensional convex solids in different three-dimensional subspaces of the whole multifield data set. Our approach is also different in the way that we do not only extract the fiber surface, but we also reduce and refine the tetrahedral input grid.

Sakurai et al. [SSC*16] developed a tool to help mathematicians and novices to learn and investigate the topology of fibers of functions from $R^3 \to R^2$. Their tool visualizes multivariate functions, extracts fiber singularities, and let the users interactively perform function perturbations to better understand the fiber topology.

Some work was done in the field of multiphysics visualization, where the work by Perdikaris et al. [PIG*16] is an example. They analyzed a cerebral aneurysm using streamlines to visualize blood flow and highlighted local stress on the arterial wall to show, if the flow inside the aneurysm implies high stress on the wall. To put it in another way, they visualized multiphysics interactions of the blood flow and the surrounding tissue boundary.

In the context of visualization of multivariate, geological data, Rocha et. al [RASS17] introduced a new technique called Decal-Maps. It maps two-dimensional texture quads onto an arbitrary curved surface, which evades the clipping and detachment problems of the default quad-based placement of glyphs on such surfaces. This technique is used in combination with layering and illustrative techniques to visualize multiple attributes of an oil reservoir in the geological subsurface at the same time. But this technique also has its constraints in the number of visualized fields, which is not a problem for our algorithm.

Another visualization tool for multivariate, geological data was introduced by Dasgupta et. al [DKG15]. They want to analyze the interactions of chemical species and microorganisms over time. To overcome this problem, they used parallel coordinates as well as scatter plot matrices to show the vast amount of bivariate attribute interactions, and density views to show the temporal distribution of each attribute. In contrast to our approach, they only use the default techniques to visualize multiple fields at the same time and they are only able to show the interactions between two attributes per view, whereby we could show the interactions of three attributes and could also use the advantages of the three-dimensional space, like rotation, to focus on regions of interest.

For an explorative visualization of d-dimensional data, it is necessary to filter them. Parallel coordinates and multiple views belong to the most common methods for filtering. The articles by Roberts [Rob07], and Heinrich and Weiskopf [HW13] present an overview of existing techniques. The work of Johannsson and Forsell [JF16] gives an overview of the extensions of parallel coordinates including an evaluation of the usability of them. Another paper by Lu and Shen [LS17] describes an interactive workflow for processing multivariate data using connected subspaces. This allows the investigation of volumetric data sets and the identification of features in the subspaces.

In this paper, the work of Blecha et al. [BRS*19] is extended to d-dimensions. They used fiber surfaces for the analysis of a multiphysics simulation of a generic, i.e. non-site specific, model of a repository for nuclear waste, which they extracted from a three-dimensional attribute space. This space was built by one scalar field per physical process, i.e., thermodynamics, hydrodynamics, and solid mechanics. This application uses the fiber surface extraction algorithm of Raith et al. [RBN*19], which generalizes mentioned work by Carr et al. [CGT*15] to three dimensions in the range.

3. Fiber Surfaces for many Variables: The Smooth Case

We consider a number d of smooth scalar fields over a three-dimensional domain $D \subset \mathbb{R}^3$,

$$s : \mathbb{R}^3 \supset D \to \mathbb{R}^d. \tag{1}$$

We are interested especially in the case $d > 3$. Similarly to [KTCG17, RBN*19], we define the fiber f_y of s for the value $y = (y_1, \dots, y_d) \in \mathbb{R}^d$ as the set

$$f_y := s^{-1}(y), \tag{2}$$

i.e., all points $x \in D$ with $s(x) = y$. The majority of fibers f_y of s will be empty, as there are more conditions $s_j(x) = y_j$ from the scalar variables than the three dimensions x_i. Nevertheless, some fibers are obviously not empty, as there are some values y of the function s. The basic insight is that we need to combine enough values y to create a surface in D consisting of fibers. For this purpose, we use hypersurfaces of values in $\mathbb{R}^d$. Let $M \subset \mathbb{R}^d$ be a smooth $(d-1)$-manifold which is locally defined as being orthogonal to a d-dimensional normal $n : M \to \mathbb{R}^d$. Then we define the fiber surface of s with regard to M as

$$f_M := s^{-1}(M), \tag{3}$$

i.e., the preimage of M under s. The name fiber surface is only justified if this is a surface in D, at least for regular cases. To see this, let us consider a point

$$c = s(a) \in \mathbb{R}^d \tag{4}$$

that lies on M with the local normal $n(c)$. We assume c to be a regular value of s, so the derivative $Ds(a)$ has full rank, i.e.

$$\mathrm{rk}(Ds(a)) = 3. \tag{5}$$

Since s is smooth, $s(D)$ is a smooth three-manifold inside some neighborhood $U \subset \mathbb{R}$ of c and Ds has full rank in the whole neighborhood. Furthermore, we assume that $s(D)$ intersects M at c transversally. This means that the defining normal $n(c)$ is not parallel to $s(D)$ at c or simply

$$\dim(s(D) \cap M) = 2 \tag{6}$$

in some neighborhood $V \subset U$ of c. Since Ds has full rank in this neighborhood, the preimage

$$s^{-1}(V \cap s(D) \cap M) \subset f_M \tag{7}$$

is also a two-manifold. Therefore, our fiber surface is indeed a surface around regular points if the image of s intersects the defining manifold transversally. This also means, that it splits the domain D locally into two parts and behaves similarly to an isosurface.

4. Fiber Surfaces for many Variables: The Piecewise-Linear Case

The previous section uses the usual terms of analysis. However, our current implementation is based on a piecewise linear setting. We

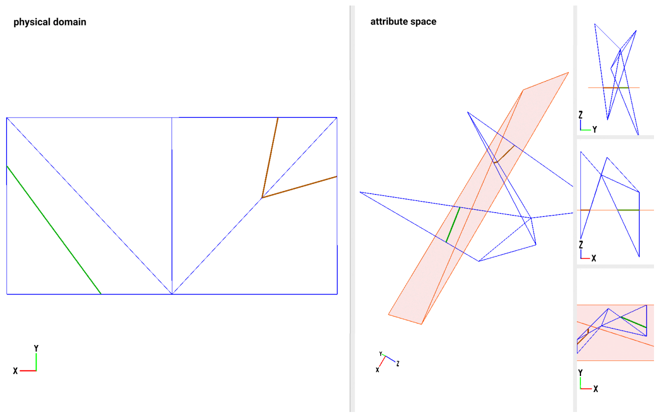

Figure 1: *Example of a projection of a three-dimensional attribute space into a two-dimensional physical domain. The green and orange-brown lines are the intersections of the red hypersurface with the blue grid.*

assume that our data is given on a tetrahedral grid

$$\mathcal{T} = \{T_i | i = 1, \ldots, m\}, \tag{8}$$

defining the domain

$$D = \bigcup_{i=1}^{m} T_i \subset \mathbb{R}^3 \tag{9}$$

where T_i describes the different tetrahedra of the grid. d scalar fields are given by d scalar values at each vertex of the complex. By linear interpolation over each tetrahedron, we obtain a piecewise linear multi-scalar field

$$s : D \to \mathbb{R}^d. \tag{10}$$

In the attribute space $\mathbb{R}^d$, we define our fiber surface by a convex interactor. We describe this interactor $M \subset \mathbb{R}^d$ as intersection of n half spaces

$$M = \bigcap_{j=1}^{n} H_j. \tag{11}$$

Each half space is defined by an outward pointing d-dimensional unit normal n_j and a scalar distance to the origin b_j, i.e.,

$$H_j := \{x \in \mathbb{R}^d | n_j \cdot x - b_j \leq 0\} \tag{12}$$

Please note, that b_j needs to be negative if the origin is not inside H_j.

In the range $\mathbb{R}^d$, $s(D)$ is an immersed three-dimensional piecewise linear grid consisting of tetrahedra $s(T_i) \subset \mathbb{R}^d, i = 1, \ldots, m$, i.e., a set of tetrahedra where tetrahedra may intersect each other in the interior. The fiber surface

$$f_M := s^{-1}(M) \tag{13}$$

is the preimage of the intersection of $s(D)$ with all half spaces H_j defining M. In the regular case, s maps a tetrahedron T_i to a 3D-volume $s(T_i)$ in the range $\mathbb{R}^d$. (This is the case if the four d-dimensional vectors of scalar values at the vertices of T_i are linearly independent.) For a single half space H_j, we may look at the interesting case, where it actually intersects our mapped tetrahedron $s(T_i)$. In general, the three-dimensional subspace of $\mathbb{R}^d$ spanned by $s(T_i)$ will not lie inside H_j. This is called a transversal intersection. In this case, H_j intersects $s(T_i)$ like a two-dimensional plane. The preimage of this intersection is a planar piece of our fiber surface if it is inside all other half spaces. Therefore, we repeat this intersection for each half space and each tetrahedron. The actual computation of this intersection is described in the next section.

To illustrate these concepts a little bit more, one may look at Fig. 1. We consider the simpler case of a two-dimensional simplicial grid on the left, where there are three piecewise linear scalar fields given by assigning three scalar values to each vertex of the grid. This defines a piecewise linear mapping s of the whole grid into the three-dimensional range on the right. Now, we use a hypersurface consisting of two coplanar red triangles to define our "fiber surface". As can be easily seen, our red hypersurface intersects the two-dimensional grid on the right at two piecewise linear curves marked in orange-brown and green. The preimage of these two curves can be seen as two piecewise linear curves on the left. This is our "fiber surface", which is actually a fiber line in this case. It can be easily seen that the situation on the left is indeed very similar to isolines.

5. Fiber Surface Extraction Algorithm

Our algorithm is based on the idea to reduce all computations to the intersection of a plane with a tetrahedron in three dimensions. This may be a bit surprising at first glance, but it should be clear that this is a geometric problem that can be solved efficiently. Furthermore, we ensure in the process that all intersections result in replacing the old tetrahedron by new tetrahedra that have the (hyper-)surface as part of their faces but do not intersect it in the interior. This further simplifies the algorithm. Before we can do so, we need some preparations, and after we are finished, we do some post-processing to ensure that we are always dealing with tetrahedra.

5.1. Reducing the Problem to Plane-Tetrahedron Intersection

As mentioned before, in the range, we have a tetrahedral mesh immersed into d dimensions. Furthermore, we want to compute the intersection of all tetrahedra T_i with a $(d-1)$-dimensional hyperplane H_j. We have a d-dimensional normal n_j for each hyperplane that defines which side of the hyperplane is considered inside and which side is considered outside. As we assume that we have a rather small number of hyperplanes compared to the number of tetrahedra, we intersect all tetrahedra with one hyperplane after the other (see Alg. 1). We keep all hyperplanes in a list and take out one at a time. Furthermore, we intersect each tetrahedron with the current hyperplane, one by one. This leaves us with the task of intersecting a tetrahedron with a hyperplane in d dimensions. The next step is that our tetrahedron lives in a three-dimensional subspace of the range $\mathbb{R}^d$. Therefore, we project the hyperplane into this three-dimensional subspace. This results in three cases:

1. If the hyperplane does not intersect the subspace, it cannot intersect the tetrahedron, so we are done.

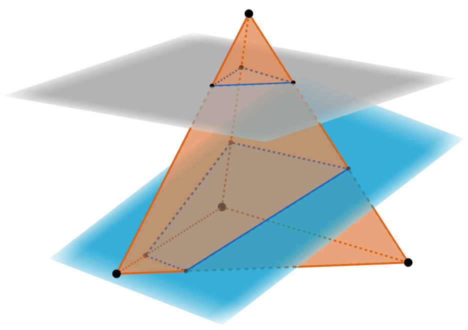

Figure 2: *Intersection of a tetrahedron with two planes, which produces a triangular and a quadrilateral cut.*

2. If the hyperplane contains the three-dimensional volume of the subspace, it contains the whole tetrahedron and we are done as well.
3. We have the interesting case: The hyperplane intersects the subspace as a plane.

Overall, we only need to look at a plane intersecting a tetrahedron in three dimensions. We project the d-dimensional normal of the hyperplane into the three dimensions to know what is inside and what outside. Concerning the algorithm there is no projection. It only helps to understand that the intersection of a tetrahedron with a hyperplane will always be planar and happens inside a three-dimensional subspace, so there is no need to think about higher dimensional geometry here.

The intersection of all hyperplanes with all tetrahedra is numerically more stable as each intersection point is only computed once. This guarantees to produce no holes inside our fiber surfaces which could happen by computing intersection points multiple times with very small differences due to numerical errors. Also, this leads to a smaller effort during the retessellation as we have all required vertices before creating any new tetrahedra.

5.2. Plane-Tetrahedron Intersection

We have to intersect a plane with a tetrahedron in three dimensions. It should be noted that we could adapt the method by Raith et al. [RBN*19] in this case. Nevertheless, we operate slightly differently as our hypersurface is defined in another way as we are using hyperplanes instead of triangles. We compute all vertices in barycentric coordinates relative to the tetrahedron in question, so we can project these points back into physical space as well as the range $\mathbb{R}^d$. Each half space is defined using a d-dimensional normal n_j and the (signed) distance b_j to the origin. These two elements are used to form the Hesse normal form (HNF) for this specific hyperplane. The following equation allows to check for each vertex of the tetrahedron if it is inside or outside:

$$\vec{x} \cdot \vec{n}_j - b_j \leq 0, \quad \vec{x} \in \mathbb{R}^d, b_j \in \mathbb{R}. \tag{14}$$

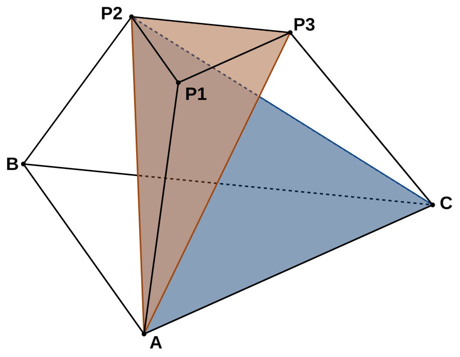

Figure 3: *Prism-type polyhedron as result of the intersection of the tetrahedron ABCD with a plane, where vertex D is clipped. The lines $\overline{AD}$, $\overline{BD}$, and $\overline{CD}$ intersect the plane in the points P_1, P_2, P_3. All quadrilateral faces of the frustum were triangulated, which results in the dark orange lines $\overline{P_2A}$ and $\overline{P_3A}$ as well as in the blue line $\overline{P_2C}$. These new lines were used to create the three new tetrahedra $ABCP_2$, ACP_2P_3, and $AP_1P_2P_3$.*

This leads to five different cases (except for the ordering of the vertices):

1. All vertices lie on the inside.
2. One vertex lies outside.
3. Two vertices lie in- and two on the outside of the hyperplane.
4. Three of the vertices are lying outside.
5. All four vertices are marked as outside.

Cases 1 and 5 are simple as either the tetrahedron is copied to the new grid or it is excluded from this list of new tetrahedra. Case 2 and 4 produce the same intersection during the clipping of a tetrahedron with a plane (see intersection of orange tetrahedron and grey plane in Fig. 2). The only difference is whether the tetrahedrization of the result is trivial, as in case 4 where the result of the clipping is already a tetrahedron, or not. In case 2, the result of the clipping is a prism-type polyhedron (see Fig. 2) with a triangular intersection. Case 3 describes the only case where the intersection, which is quadrilateral, has to be triangulated as well as the lateral surfaces (see intersection with blue plane in Fig. 2).

The position of the intersection of the tetrahedron with the plane is calculated on the basis of the intersection of the edges with the current plane (see Alg. 2). For the relative position of both vertices P_1 and P_2 of an edge, the distance to the plane of each vertex along the normal is calculated and the exact location of the intersection point is examined using the following equation:

$$\lambda = \frac{\vec{n_j} \cdot \vec{OP_1} - b}{\vec{n_j} \cdot \vec{OP_1} - \vec{n_j} \cdot \vec{OP_2}} \tag{15}$$

which calculates λ as barycentric coordinate of the intersection point of the edge $\overline{P_1P_2}$ starting in P_2 with $\lambda = 0$, and with the origin O.

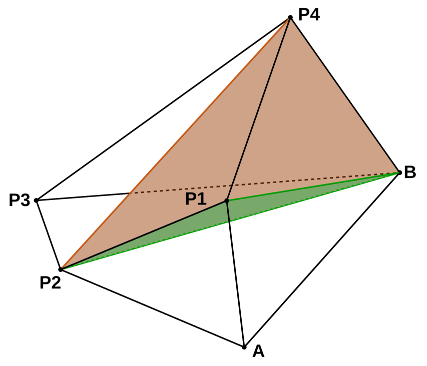

Figure 4: *One of two prism-type polyhedra as result of the intersection of the tetrahedron ABCD with a plane, where vertex C and D are clipped. The lines $\overline{AC}$ and $\overline{AD}$ intersect the plane in the points P_2 and P_1 and the lines $\overline{BC}$ and $\overline{BD}$ in the points P_3 and P_4. All quadrilateral faces of the frustum were triangulated, which results in the green lines $\overline{P_2B}$ and $\overline{P_1B}$ as well as in the red line $\overline{P_2P_4}$. These new lines were used to create the three new tetrahedra ABP_2P_1, $BP_2P_1P_4$, and $BP_2P_4P_3$. The other prism-type polyhedron could be split in the same way.*

5.3. Splitting the Result into Tetrahedra

Our algorithm assumes that we always have a list of tetrahedra to process with the remaining hyperplanes. Therefore, we need to split the current tetrahedron, after computing the intersection with the plane, into several tetrahedra. The general principle is that all new tetrahedra should not intersect the current hyperplane. In all cases, we split the tetrahedron with the plane into two parts. As illustrated in Fig. 2 and mentioned before, we have a triangular or a quadrilateral intersection. In the triangular case, one part of the tetrahedron is already a tetrahedron, so there is nothing to do. The remaining part is shown in Fig. 3. We can easily create three tetrahedra by splitting the three resulting quadrilateral faces of the remaining prism-type polyhedron.

In the case of a quadrilateral intersection, we are left with two prism-type polyhedra as shown in Fig. 4. Again, we end up with three quadrilateral surfaces (one is the actual intersection), and two triangular faces. We split the three (always convex) quadrilaterals and create three tetrahedra.

As final step, we replace the current tetrahedron in our list of tetrahedra with the new set. It should be noted that the so retessellated grid could also be used as input for other analysis and visualization techniques as it could reduce a big data set to a small subset of tetrahedra. In addition, we mark the triangular faces of the actual intersection as intersection faces to allow for drawing the fiber surface later.

It's worth pointing out that also the triangles of the previous intersection of the tetrahedra with a hyper plane must be considered in each of the following refinement steps. The loop in line 12 of Algorithm 1 is used to retriangulate the before marked fiber surface triangles as they have to be refined also if the corresponding tetrahedra will be refined. This is necessary because if a fiber surface triangle is added to the list, it loses its connection to the tetrahedron it is a face of. The video of the intersection of a tetrahedron with a sphere in the supplemental material shows a good use case because due to the curvature of the sphere earlier fiber surface triangles frequently lie outside another hyperplane.

5.4. Drawing the Fiber Surface

After all tetrahedra and all hyperplanes are processed, we have a set of tetrahedra in the d-dimensional attribute space that do not intersect the hypersurface M in the interior. Furthermore, the intersections between the tetrahedral mesh and the hypersurface consist of a list of faces that are marked. Also, whenever we split a tetrahedron into several tetrahedra, we compute the coordinates of the new vertices in the physical domain. We draw the fiber surface by projecting these faces back to physical domain. It should be noted that the fiber surface will only split the domain into parts, if the hypersurface separates the immersed tetrahedra into parts. This means that our fiber surface might end somewhere in the middle of the physical domain, which is correct in a theoretical sense. If users require the separation of the mesh into two parts, they have to extend the defining hypersurface so that it cuts the immersed tetrahedral mesh in the range into two parts.

5.5. Avoiding Numerical Errors

If the algorithm is implemented as described above, it works in principle but it will show some small gaps due to numerical problems. This happens if the intersection points along lines or faces of the tetrahedra are computed multiple times for all neighboring tetrahedra and numerical errors create slightly different coordinates. Furthermore, it is not guaranteed that we are dealing with a valid triangulation of the domain at all times. This is the case if we split a quadrilateral on a face into two triangles and do so in different ways for the two neighboring tetrahedra.

We avoid these problems by keeping lists of all vertices, edges, faces, and tetrahedra of the mesh including connectivity and a list of values defined on the vertices at all times (see Alg. 1). In addition, the intersections are calculated based on the edges (see line 8 in Alg. 1 and Alg. 2). Whenever we compute an intersection of an edge or split a face into triangles, we look for an existing intersection or split and use it. If it is not there, we make our choice and store it, for example, to avoid different triangulations of quadrilateral faces of neighboring tetrahedra. There may be more efficient ways to resolve the problem but this solution worked for us.

5.6. Complexity

In normal use cases we would expect a complexity of (number-of-vertices + number-of-edges + number-of-tetrahetra) * number-of-hyperplanes but extreme cases where each tetrahedron is intersected by each hyperplane which produces alternating one and three new tetrahedra can be created but are not the normal use case. This would lead to a duplication of the number of tetrahedra in each step which results in a complexity where the effort (the number of tetrahedra to process) increases exponentially with $2^{number-of-hyperplanes}$. Normally, in each step some tetrahedra were added due to the retessellation and some were dropped because they lie completely on the outside.

Algorithm 1 Generalized Fiber Surface Extraction Algorithm

```
Require: tetrahedral grid $Tet$ in attribute space, list $M$ of hyperplanes of the interactor(s)
 1: compute set of grid vertices $Vertices$ of $Tet$
 2: compute set of grid edges $Edges$ of $Tet$
 3: compute set of vertex values $Values$
 4: for all hyperplanes $H_j \in M$ do
 5:     let $n_j$ be the outward normal of $H_j$
 6:     let $a_j$ be the center of the corresponding interactor on which the hyper plane is defined on
 7:     let $b_j$ be the distance of $H_j$ to $a_j$
 8:     for all edges $e_k \in Edges$ do
 9:         IntersectEdgePlane($e_k$, $n_j$, $a_j$, $b_j$, $Tet$)
10:     end for
11:     create a new list $NewMarkedFaces$
12:     for all faces $f_m \in OldMarkedFaces$ do
13:         if all vertices of $f_m$ lie on the outside of $H_j$ then
14:             do nothing
15:         else if all vertices of $f_m$ lie on the inside of $H_j$ then
16:             add $f_m$ to the new list of $NewMarkedFaces$
17:         else if $H_j$ intersects $2D$ subspace of $f_m$ as line then
18:             extract the inner and outer points of $f_m$
19:             extract the intersection points of $f_m$
20:             triangulate the result of the intersection with $H_j$
21:             add the new triangels to the list $NewMarkedFaces$
22:         end if
23:     end for
24:     create a new tetrahedral grid $NewTets$
25:     for all tetrahedra $T_i$ of $Tet$ do
26:         if all vertices of $T_i$ lie on the outside of $H_j$ then
27:             do nothing
28:         else if all vertices of $T_i$ lie on the inside of $H_j$ then
29:             add $T_i$ and the values defined on its vertices to the new grid $NewTets$
30:         else if $H_j$ intersects $3D$ subspace of $T_i$ as plane then
31:             extract the inner and outer points of $T_i$
32:             extract the intersection points of $T_i$
33:             triangulate the quadrilateral faces
34:             add the faces on the intersection between $T_i$ and $H_j$ to the list $NewMarkedFaces$
35:             create new tets using the inner and the intersection points
36:             add the new tets to the new grid $NewTets$
37:         end if
38:     end for
39:     set the list $OldMarkedFaces$ to $NewMarkedFaces$
40: end for
41: project all tetrahedra of $NewTets$ and marked faces back to the domain $D$
42: draw all faces of the list $NewMarkedFaces$
```

6. Interactive Fiber Surface Definition

This section explains the multiple views of the range and the interactive definition of the fiber surfaces through interactors in these views. As defined previously, the constraining hypersurface in d dimensions is a convex object which is defined as intersection of multiple half spaces given by a set of hyperplanes. It has to be convex as our algorithm works analogously to a classical clipping technique where it cuts off parts of a tetrahedron. This is not possible with a non-convex interactor. In such a non-convex case, clipping with one half space H_i may cut off parts of the mesh that are inside the interactor. However, as the algorithm refines the tetrahedral grid incrementally in each step, cut off parts are not available in a later refinement step. Nevertheless, a non-convex interactor may be split into convex parts which can be processed separately. However, we do not show the high-dimensional range in one view, so the convex object is not manipulated directly. Instead, we let users select regions of interest in several three-dimensional subspaces of the range. In each space, we offer the same solids as interactors as Raith et al. [RBN*19] in their work, i.e., tetrahedra, cubes, triangulated balls, regular cylinders, pyramids, and prisms. All other usual primitives of constructive solid geometry (CSG) could also be used as interactors because they are all convex. They only have to be approximated by a triangulation. If CSG set operators, like intersection, union, etc., are used to create more complex solids, one would need to compute the effects on the fiber surface as well. An alternative would be to split the CSG result into convex parts. The final hypersurface is defined as intersection of all the hyperplanes (from interactors in all views), so they basically serve as constraints on the region of interest in the various subspaces.

Algorithm 2 Calculate Edge Plane Intersection

```
Require: edge $e$, hyperplane normal $n$ in $d$ dimensions, center point of the corresponding interactor $a$, distance of the hyperplane to the center $b$, global grid $Tet$ with lists $Vertices$, $Edges$, $Values$, $EdgeIntersectionMapping$
 1: calculate distances of vertices of edge $e$ to hyperplane
 2: if both distances are smaller or equal to 0 then
 3:     add both vertices to $Vertices$
 4:     add edge $e$ to $Edges$
 5:     add values defined on both vertices to $Values$
 6: else if only one distance is smaller or equal to 0 then
 7:     compute intersection point of edge $e$ with plane
 8:     compute corresponding value for the intersection point
 9:     add the inner vertex to $Vertices$
10:     add new intersection point to $Vertices$
11:     add value defined on the inner vertex to $Values$
12:     add value defined on the intersection point to $Values$
13:     create new edge using the inner and the intersection point
14:     add new edge to $Edges$
15:     add mapping for edge $e$ to intersection point and value to $EdgeIntersectionMapping$
16: else if both distances are greater 0 then
17:     do nothing
18: end if
```

6.1. Fiber Surface Definition for Many Variables

The actual definition of our fiber surfaces starts with the creation of the first subspace defined as combination of three attributes. Following this, the tetrahedral grid is mapped from the physical domain into this subspace. The structure of the grid and the

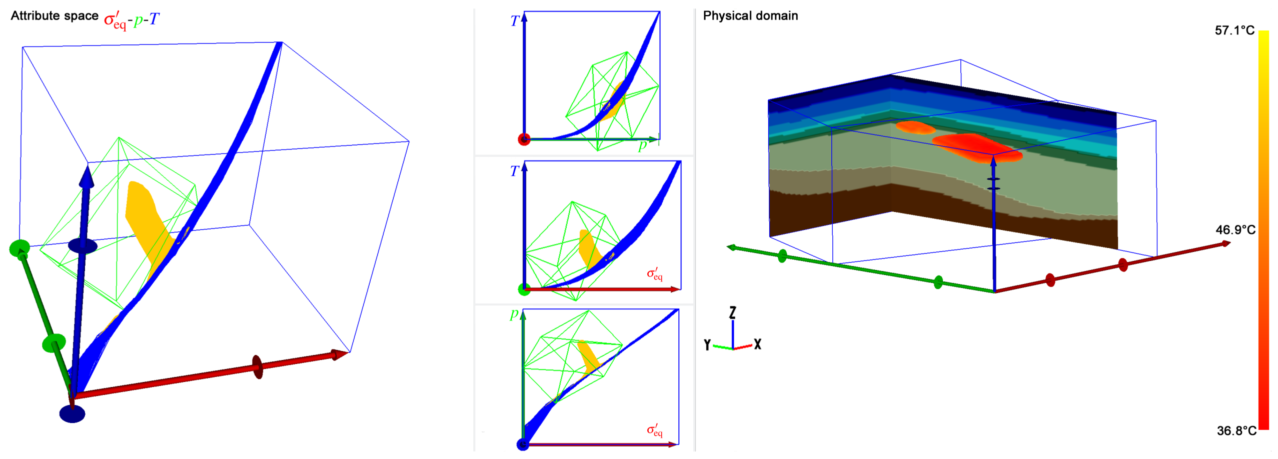

Figure 5: *Extraction of a fiber surface in a generic model of a nuclear waste repository* 160 *years after the start of the simulation, i.e., after the start of the construction of the repository. The left-hand side of the image shows the* σ'_{eq}-p-T *attribute space of the system* (σ'_{eq}: $9.13 \cdot 10^5$ *Pa* - $8.52 \cdot 10^7$ *Pa, p:* $-4.38 \cdot 10^2$ *Pa* - $3.73 \cdot 10^7$ *Pa, T:* 8.0°*C* - 142.8°*C). The distribution of all values of the data set used is shown as blue wireframe and the tetrahedra with values which differ from their initial values at time step 0 are marked in yellow. An interactor (green box) is used to select the yellow bulge, which has formed. The right-hand side illustrates in the physical domain (Euclidean space) the bounding surface of the region marked by the interactor in attribute space. The background colors clarify the geological layers for context. For information on the geological layers, see [JBB*17, JBH*17, JBJ*17]. This image and further visualizations of the nuclear waste repository can be found in the paper by Blecha et al. [BRS*19].*

neighborhood of the cells are preserved during this projection, which also could be done in the opposite direction. Basically, this means that we do piecewise linear interpolation on our tetrahedral grid. For further information about the mathematical definition and the feasibility of this mapping, we refer to the paper of Raith et al. [RBN*19]. The next step is to create an interactor inside the current subspace and use it to select a region of interest. If we stop here, we get the same fiber surface as the algorithm of Raith et al. would produce under the same conditions.
To create fiber surfaces for more than three restricted attributes, a new three-dimensional subspace must be created. This one could be based on any combination of scalar variables. Each subspace benefits from the zooming, panning, and rotation of its own view with respect to the identification of attractive regions. A new interactor could be defined in this new subspace, whereby the intersection of all interactors defines one restraining hypersurface. Of course, each interactor may also define its own hypersurface to get a visual impression of its impact on the data.

7. Examples

In the first two cases we demonstrate our method on a data set from geotechnical engineering. This data set has also been used by Blecha et al. [BRS*19]. It shows a generic, i.e., not site-specific, model of a nuclear waste repository in a claystone/ clayey marl formation (see Fig. 2.6 [JBH*17, p. 21] and Fig. 2.7 [JBH*17, p. 22]). The numerical simulation of the physical processes occurring in this repository has been realized within the framework of the ANSICHT project [JBB*17, JBH*17, JBJ*17]. The simulation began with the excavation of the repository area (see Fig. 2.4 [JBH*17, p. 18]). This is followed by the storage of the waste and the subsequent post-closure phase. The representation of the heat producing waste packages in the repository was carried out with the aid of prescribed heat sources. For the simulation of the processes, a thermo-hydro-mechanical model is chosen as the process model, since it can capture the rock deformations and fluid movements in response to temperature changes. The numerical model consisted of eleven geological layers, each with its own material properties. The system was simulated with the scientific, open-source, finite element framework OpenGeoSys [KBB*12, BFK*19]. For more details on the model, its context, and the parameters used, see [JBB*17, JBH*17, JBJ*17]. We show the correctness and the possibilities of our algorithm using this data. In the first case, we compare the results of our new algorithm with the results of Blecha et al. [BRS*19]. Their results serve as a reference to verify the correctness of the new approach. In the second case, we show what is possible with the new algorithm. The third example applies our fiber surfaces to data from a regional weather simulation. Each use case ends with some statistics, for example, about the time to produce the results, and the number of intersected tetrahedra. All these information were acquired using a standard workstation with two Intel Xeon E5-2630 v3 with 2.40 GHz, 32 GB RAM and a Nvidia GeForce GTX 980. The algorithm was implemented inside our framework using C++.

7.1. Coupled thermo-hydro-mechanical simulations of a generic nuclear waste repository in clay rock

The first case shows the correctness of our new method with the data set from the work of Blecha et al. [BRS*19]. Here a three-dimensional attribute space is mapped to a three-dimensional physical domain. For the analysis we choose three physical variables to represent the thermo-hydro-mechanical model which are also used in the work of Blecha et al. [BRS*19]. We use the temperature T

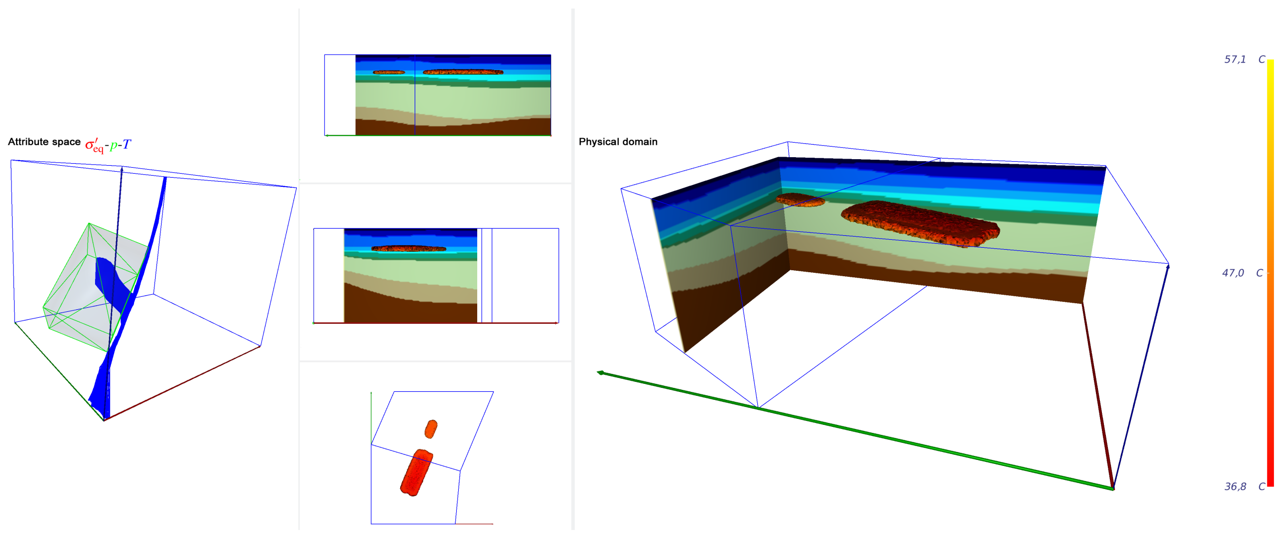

Figure 6: *Extraction of a fiber surface with the new extraction algorithm in the same setting and with the same interactor as in Fig. 5.*

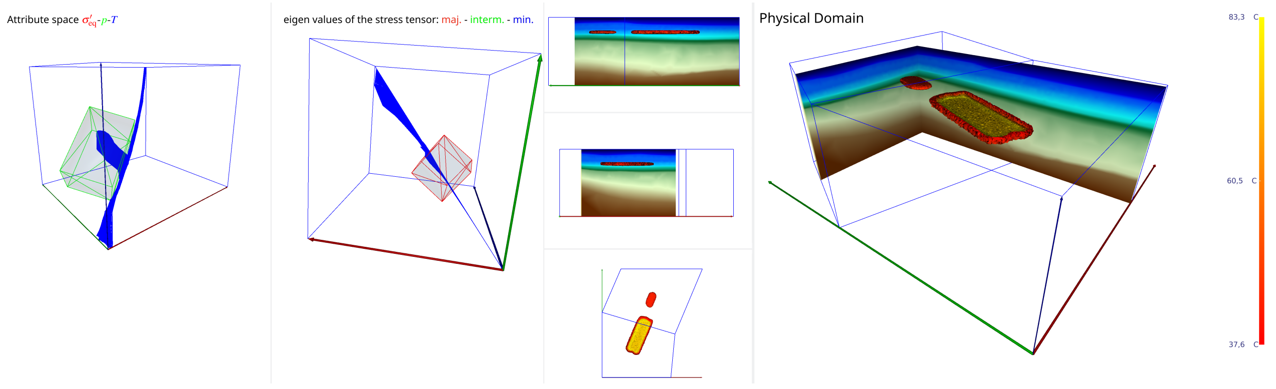
Attribute space σ'_{eq}-p-T
eigen values of the stress tensor: maj. - interm. - min.
Physical Domain
83,3 C
60,5 C
37,6 C

Figure 7: *Extraction of a fiber surface with the new extraction algorithm in the same setting as in Fig. 5. Furthermore, a second interactor is used to restrain the region of interest even further. The second subspace is built of the eigenvalues of the stress tensor in the order: major, intermediate, and minor.*

in °C (thermodynamics), the fluid pressure p in Pa (hydrodynamics), and the equivalent effective stress σ'_{eq} in Pa (solid mechanics) defined by:

$$\sigma'_{\text{eq}} = \sqrt{\frac{3}{2}\text{dev}\,\boldsymbol{\sigma}' : \text{dev}\,\boldsymbol{\sigma}'} \qquad (16)$$

where : is the double "Überschiebung". Each of these variables describes a scalar field defined on the given grid.

Fig. 6 shows the state of the repository after 160 years. The attribute space with the tetrahedral grid in blue and the cuboid interactor as a green, shaded wire-frame is shown in the left part of Fig. 6. Regarding the used visualization techniques and the discussion of their usefulness in Raith et al. [RBN*19], it would also be possible to visualize the attribute spaces using these techniques as the the algorithm is implemented inside the same framework. Nevertheless, in our opinion the wire-frame visualization is currently the best one, as sharp features can be seen inside the attribute space which were blurred by a density-based volume rendering or a related visualization in the spirit of continuous scatterplots. The right part of Fig. 6 shows the physical domain with the extracted fiber surfaces. The fiber surface is colored by the temperature field. Additionally, the geological layers are displayed in the background in order to better classify the detected areas for data analysis. In this time step, the inserted disturbance is recognizable as a bulge separated from the normal curvature/ natural gradient in the attribute space. In this space, we set the interactor to the same position as in the paper by Blecha et al. (see attribute space in Fig. 5). The position of the repository in the physical domain can be clearly identified, which can also be seen in Fig. 6. An exact comparison of the surfaces shows that the surfaces are the same except for minimal numerical errors.

The data set has 498 712 tetrahedra which were reduced to 216 849 tetrahedra in about 1:08 minutes on average (starting with the import of the data and ending with visualization of the resulting fiber surface) using 12 hyperplanes specified by the interac-

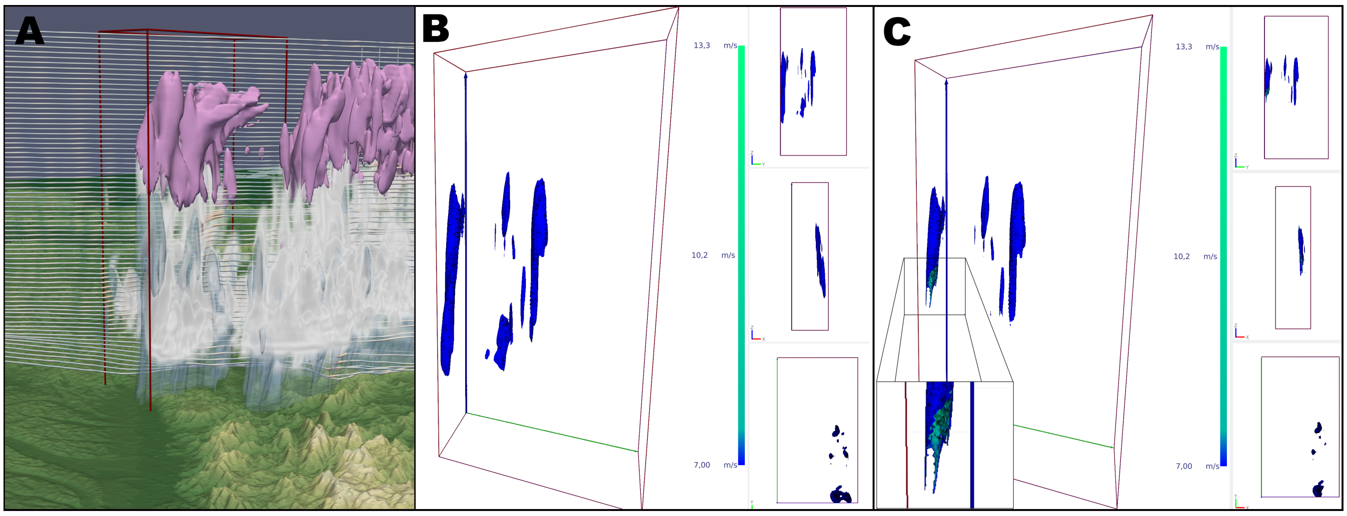

Figure 8: *Application of fiber surfaces to atmospheric model data.* ***A:*** *The red box denotes the selected subset of the data used for our further analysis. The ParaView rendering shows cloud water (grey volume rendering), cloud ice (magenta isosurface), and rain water (semi-transparent blue isosurface). A set of streamlines illustrates the wind field. The vertical wind velocity is shown by the color mapping on the streamlines.* ***B:*** *After filtering the data in attribute space with regard to strong positive vertical velocity, regions of updraft are visible in physical space.* ***C:*** *In the second step, the data is further cut with respect to high values of cloud water and cloud ice in attribute space. The resulting fiber surfaces show regions with high upward moisture transport.*

tor (green box on the left side of Fig. 6). The resulting fiber surface consists of 94 464 triangles. During the refinement of the grid and the calculation of the fiber surface 121 723 tetrahedra were clipped (only tetrahedra which were intersected by a hyperplane were counted) and no triangle of the fiber surface had to be refined.

7.2. Extended analysis of the THM data set

The second case shows the possibilities of the new method. The same data set as before in subsection 7.1 is used. The simulation contains further variables of interest for the engineers, which motivated an extended analysis compared to the one in 7.1. In addition to the previously used variables T, p, and σ'_{eq}, the eigenvalues of the stress tensor are used, which correspond to the principal stresses. Consequently, we have a setting with six scalar variables on a three-dimensional physical domain, creating a six-dimensional attribute space. In principle, many more variables can be considered in a coupled THM simulation, for which the interpretation of the results themselves can be an iterative process. Having visual data exploration tools at one's disposal can thus be of great benefit.

In Fig. 7, we show both attribute spaces. In the left part of the image, the attribute space for the three variables σ'_{eq} (equivalent effective stress), p (fluid pressure), and T (temperature) is shown. Here, we select again the region with the introduced disturbance. Thereupon in the middle part of Fig. 7, the second attribute space consisting of the principal stresses (eigenvalues of the stress tensor) is visualized. With the help of a new cube-shaped interactor in red, a subdomain is again selected. Finally the restricted area is mapped back into the physical domain. The extracted (generalized) fiber surface can be seen in the right part of Fig. 7. The differences between the extracted fiber surface with one and two interactors can be seen in the right parts of Fig. 6 and Fig. 7. It can be seen, that a smaller region is selected, if the region is further constrained by the second interactor in the space of principal stresses. This smaller region is in the interior of the repository and, therefore, very close to the emplacement area, whereby a higher temperature is displayed on the fiber surface. This example illustrates the general purpose of this method: The generalized fiber surface allows engineers to extract areas based on constraints with respect to all scalar variables of interest in an explorative manner with immediate feedback.

In this example the algorithm reduces the number of tetrahedra to 206 131 tetrahedra in about 1:39 minutes on average using 24 hyperplanes specified by the two interactors (green and red boxes on the left side of Fig. 7). The resulting fiber surface consists of 124 466 triangles. During the refinement of the grid and the calculation of the fiber surface 160 431 tetrahedra were clipped and 1 547 triangles of the fiber surface were refined.

7.3. Application of Fiber Surfaces in Atmospheric Modeling

The third example uses data from a regional weather simulation that was performed at the DKRZ using the ICON (ICOsahedral Non-hydrostatic) model, and which was made available as part of the IEEE SciVis Contest 2017 [dkr]. The data originates from the HD(CP)^2 project and shows the weather situation above Germany for April 26, 2013. ICON is jointly developed by the Max Planck Institute for Meteorology (MPI-M) in Hamburg and the German Weather Service (DWD) in Offenbach. It is a framework that is defined on an icosahedron with an equal area projection on which data is sampled via primal triangular cells, dual hexagonal cells and hybrid quadrilateral cells [LL11]. Beneficial for employing ICON is that it exhibits no poles, allows an easy mesh refinement in local areas, and – for running as Earth-System-model – the coupling between the oceanic and atmospheric components is much easier, as both models share the same grid layout, albeit with different resolutions. Within the last years, ICON has been extended to permit large eddy simulations at cloud resolving resolutions as part of the HD(CP)^2 project [HDH*17]. The project's aim was high resolution, regional simulations that could explicitly resolve clouds and precipitation processes. More recently, as part of the DYAMOND initiative and within the EU funded project of ESiWACE, the atmosphere has been globally simulated with a resolution of 2.5 km [SSA*19]. As goal of the recently started project ESiWACE2, the spatial resolution will be further refined down to 1.25 km horizontal grid resolution. In order to access the simulation output for data analysis or visualization, alternatives to the classic post-visualization strategy have to be developed. Here the DKRZ is exploring in-situ visualization strategies using ParaView/Catalyst [RE19], as well as a progressive rendering approach using a wavelet decomposition and the visualization software Vapor [JAR*16].

For this application case, we chose to focus on a regional subset of the simulation data. In the selected region, a strong cumulonimbus system has formed. The visualization is shown in Fig. 8 and illustrates the meteorological and spatial setting. The red box denotes the subset of the data we used for our fiber surface analysis. The image depicts cloud water (grey volume rendering), cloud ice (magenta isosurface), and rain water (semitransparent blue isosurface). A set of streamlines illustrates the wind field. The vertical wind velocity is visualized via colormapping on the streamlines.

We intended to apply the fiber surface methodology to visualize relations between hydrometeorological quantities and the wind field. In the first step, we used a three-dimensional attribute space which consists of the three wind velocity components u, v, and w. Here, we selected only the upward wind (of 7 m/s and above) to extract updraft regions. The visualization in Fig. 8 B shows the extracted fiber surfaces.

In the second step, we extended our attribute space by the cloud water, cloud ice and rain water, where we selected high values for cloud ice and cloud water to further cut the fiber surface derived in the first step. The resulting chimneys shown in Fig. 8 C can be understood as the central "power supply" of this actual storm; here large amounts of energy and moisture are transported to higher altitudes. To illustrate the difference between the fiber surfaces extracted in the first and the second step, we have color mapped the vertical wind velocity onto the resulting fiber surfaces.

The extracted fiber surfaces in Fig. 8 B and C were calculated using one interactor with 12 hyperplanes, and two interactors with 12 hyperplanes each. For the results in Fig. 8 B the algorithm needs about 5:33 minutes to compute only the intersection of each hyperplane with the tetrahedra, and about 6:59 minutes starting with the import of the data, some prepossessing, and ending with the visualization of the resulting fiber surfaces. For the results in Fig. 8 C the algorithm needs about 5:54 minutes to compute only the intersection of each hyperplane with the tetrahedra, and about 7:19 minutes in total. The number of tetrahedra were reduced from 4 522 500 to 56 630 tetrahedra in B and to 39 513 tetrahedra in C. During the refinement in B, 22 085 tetrahedra were clipped to produce the fiber surfaces with 27 512 triangles where 40 of these triangles had to be refined. In C 27 935 tetrahedra were clipped to produce the fiber surfaces with 24 423 triangles where 1 392 of these triangles had to be refined.

8. Conclusion and Future Work

In this article, we generalized the fiber surface extraction algorithm by Raith et. al [RBN*19] from 3 to d dimensions in the attribute space to extend the concept of isosurfaces even further. To define a generalized fiber surface, we use a convex $(d-1)$-dimensional hypersurface. It is represented as intersection of multiple hyperplanes in the d-dimensional attribute space (the range) with a grid defined by the values of all scalar fields at each vertex of a tetrahedral grid. Our algorithm clips each tetrahedron against each hyperplane, one by one. We show that we can reduce the complexity of this calculation even more by using the fact that each tetrahedron defines a three-dimensional subspace where the hyperplane typically appears as plane. Thus, the actual intersection reduces to the intersection of a tetrahedron and a plane in three dimensions. Furthermore, we refine the tetrahedral mesh to represent the fiber surfaces as unions of faces in the mesh, which makes the algorithm even simpler to implement. Our algorithm does not only produce fiber surfaces but also extracts and refines the tetrahedral grid to the part inside the hypersurface. Therefore, it can be easily inserted into a pipeline or used to select a part of the grid with specific constraints. Users define the hypersurface using multiple three-dimensional views of the attribute space. As scientists and engineers are used to three-dimensional views, this works quite nicely and avoids any direct representation of higher-dimensional spaces in a single view.

We presented three examples to evaluate and demonstrate our algorithm. In the first example, we used the same data set and interactor as Blecha et al. [BRS*19] to confirm that our output is identical (up to numerical errors) with the results of the method from Raith et al. [RBN*19]. In the second example, we extended the hypersurface of the first example by defining a second interactor in another subspace defining a hypersurface in six dimensions as intersection of these two interactors. In the last example, we applied our new algorithm to a regional weather simulation, where two interactors in two different three-dimensional subspaces helped us to select the central "power supply" of the simulated storm inside a cumulonimbus cloud.

Directions for further research and improvement are a generalization of our piecewise linear scalar field requirement and a better visualization technique for the attribute spaces. The subspaces of the attribute space suffer from self-intersection of the tetrahedra, which was also mentioned by Raith et al. [RBN*19]. Another point of further research may concern improvement of the whole algorithm to enable a fast interactive exploration of large data sets and to improve the usability. One may also investigate the structure of the attribute space itself and its relationship to the extracted fiber surfaces. Also the visualization of multiple variable distributions on the extracted fiber surface(s), like the work by Nagaraj et al. [NN11], is of interest.

9. Acknowledgements

The Federal Institute for Geosciences and Natural Resources (BGR) is a technical and scientific agency of the Federal Ministry for Economic Affairs and Energy (BMWi). The authors would like to thank the BMWi for funding the ANSICHT project, which provided the data base for this study. 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

[BFK*19] BILKE L., FLEMISCH B., KALBACHER T., KOLDITZ O., HELMIG R., NAGEL T.: Development of Open-Source Porous Media Simulators: Principles and Experiences. *Transport in Porous Media 130*, 1 (oct 2019), 337–361. 8

[BRS*19] BLECHA C., RAITH F., SCHEUERMANN G., NAGEL T., KOLDITZ O., MASSMANN J.: Analysis of Coupled Thermo-Hydro-Mechanical Simulations of a Generic Nuclear Waste Repository in Clay Rock Using Fiber Surfaces. In *2019 IEEE Pacific Visualization Symposium (PacificVis)* (April 2019), pp. 189–201. 1, 2, 3, 8, 11

[Car14] CARR H.: Feature Analysis in Multifields. In *Scientific Visualization: Uncertainty, Multifield, Biomedical, and Scalable Visualization*, Mathematics and Visualization. Springer, 2014, pp. 197–204. 2

[CGT*15] CARR H., GENG Z., TIERNY J., CHATTOPADHYAY A., KNOLL A.: Fiber Surfaces: Generalizing Isosurfaces to Bivariate Data. *Computer Graphics Forum 34*, 3 (2015), 241–250. 1, 2, 3

[CLKH14] CHUNG D. H., LARAMEE R. S., KEHRER J., HAUSER H.: Glyph-Based Multi-field Visualization. In *Scientific Visualization*. Springer, 2014, pp. 129–137. 2

[DHCRJ*14] D. HANSEN C., CHEN M., R. JOHNSON C., E. KAUFMAN A., HAGEN H.: *Scientific Visualization: Uncertainty, Multifield, Biomedical, and Scalable Visualization*. Mathematics and Visualization. Springer, 01 2014. 2

[DKG15] DASGUPTA A., KOSARA R., GOSINK L.: VIMTEX: A Visualization Interface for Multivariate, Time-Varying, Geological Data Exploration. *Computer Graphics Forum 34*, 3 (2015), 341–350. 3

[dkr] Partner der Klimaforschung. `https://www.dkrz.de/`, publisher=DKRZ. Accessed: 2019-12-04. 10

[EH04] EDELSBRUNNER H., HARER J.: *Jacobi sets of multiple Morse functions*. Cambridge University Press, 2004. 2

[EHP08] EDELSBRUNNER H., HARER J., PATEL A. K.: Reeb Spaces of Piecewise Linear Mappings. In *Proceedings of the Twenty-Fourth Annual Symposium on Computational Geometry* (New York, NY, USA, 2008), SCG '08, Association for Computing Machinery, pp. 242–250. 2

[FH09] FUCHS R., HAUSER H.: Visualization of Multi-variate Scientific Data. *Computer Graphics Forum 28*, 6 (2009), 1670–1690. 2

[HDH*17] HEINZE R., DIPANKAR A., HENKEN C. C., MOSELEY C., SOURDEVAL O., TRÖMEL S., XIE X., ADAMIDIS P., AMENT F., BAARS H., ET AL.: Large-eddy simulations over Germany using ICON: A comprehensive evaluation. *Quarterly Journal of the Royal Meteorological Society 143*, 702 (2017), 69–100. 11

[HW13] HEINRICH J., WEISKOPF D.: State of the Art of Parallel Coordinates. In *Eurographics (STARs)* (2013), pp. 95–116. 3

[JAR*16] JUBAIR M. I., ALIM U. R., RÖBER N., CLYNE J. P., MAHDAVI-AMIRI A.: Icosahedral Maps for a Multiresolution Representation of Earth Data. In *VMV* (2016). 11

[JBB*17] JOBMANN M., BEBIOLKA A., BURLAKA V., HEROLD P., JAHN S., LOMMERZHEIM A., MASSMANN J., MELESHYN A., MRUGALLA S., REINHOLD K., RÜBEL A., STARK L., ZIEFLE G.: Safety assessment methodology for a German high-level waste repository in clay formations. *Journal of Rock Mechanics and Geotechnical Engineering 9*, 5 (oct 2017), 856–876. 8

[JBH*17] JOBMANN M., BURLAKA V., HEROLD P., KUATE SIMO E., MASSMANN J., MELESHYN A., RÜBEL A., ZIEFLE G.: *Projekt ANSICHT Systemanalyse für die Endlagerstandortmodelle Methode und exemplarische Berechnungen zum Sicherheitsnachweis*. Tech. rep., 2017. 8

[JBJ*17] JOBMANN M., BEBIOLKA A., JAHN S., LOMMERZHEIM A., MASSMANN J., MELESHYN A., MRUGALLA S., REINHOLD K., RÜBEL A., STARK L., ZIEFLE G.: *Sicherheits- und Nachweismethodik für ein Endlager im Tongestein in Deutschland. Synthesebericht.* Tech. rep., 2017. 8

[JF16] JOHANSSON J., FORSELL C.: Evaluation of Parallel Coordinates: Overview, Categorization and Guidelines for Future Research. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 579–588. 3

[JH20] JANKOWAI J., HOTZ I.: Feature Level-Sets: Generalizing Iso-Surfaces to Multi-Variate Data. *IEEE Transactions on Visualization and Computer Graphics 26*, 2 (Feb 2020), 1308–1319. 2

[JWSK07] JANICKE H., WIEBEL A., SCHEUERMANN G., KOLLMANN W.: Multifield visualization using local statistical complexity. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1384–1391. 2

[KBB*12] KOLDITZ O., BAUER S., BILKE L., BÖTTCHER N., DELFS J. O., FISCHER T., GÖRKE U. J., KALBACHER T., KOSAKOWSKI G., MCDERMOTT C. I., PARK C. H., RADU F., RINK K., SHAO H., SHAO H. B., SUN F., SUN Y. Y., SINGH A. K., TARON J., WALTHER M., WANG W., WATANABE N., WU Y., XIE M., XU W., ZEHNER B.: OpenGeoSys: an open-source initiative for numerical simulation of thermo-hydro-mechanical/chemical (THM/C) processes in porous media. *Environmental Earth Sciences 67*, 2 (sep 2012), 589–599. 8

[KTCG17] KLACANSKY P., TIERNY J., CARR H., GENG Z.: Fast and Exact Fiber Surfaces for Tetrahedral Meshes. *IEEE Transactions on Visualization and Computer Graphics 23*, 7 (July 2017), 1782–1795. 2, 3

[LL11] LEONIDAS LINARDAKIS DANIEL REINERT A. G.: *ICON Grid Documentation*. Tech. rep., Max-Planck-Institut for Meteorology, Hamburg, Germany, Tech Report, 2011. 10

[LS16] LIU X., SHEN H.-W.: Association Analysis for Visual Exploration of Multivariate Scientific Data Sets. *IEEE Transactions on Visualization and Computer graphics 22*, 1 (2016), 955–964. 2

[LS17] LU K., SHEN H.-W.: Multivariate volumetric data analysis and visualization through bottom-up subspace exploration. In *2017 IEEE Pacific Visualization Symposium (PacificVis)* (2017), IEEE, pp. 141–150. 3

[NN11] NAGARAJ S., NATARAJAN V.: Relation-Aware Isosurface Extraction in Multifield Data. *IEEE Transactions on Visualization and Computer Graphics 17*, 2 (Feb 2011), 182–191. 2, 11

[NNN11] NAGARAJ S., NATARAJAN V., NANJUNDIAH R. S.: A Gradient-Based Comparison Measure for Visual analysis of Multifield Data. *Computer Graphics Forum 30*, 3 (2011), 1101–1110. 2

[OP14] OBERMAIER H., PEIKERT R.: Feature-Based Visualization of Multifields. In *Scientific Visualization: Uncertainty, Multifield, Biomedical, and Scalable Visualization*, Mathematics and Visualization. Springer, 2014, pp. 189–196. 2

[PIG*16] PERDIKARIS P., INSLEY J. A., GRINBERG L., YU Y., PAPKA M. E., KARNIADAKIS G. E.: Visualizing multiphysics, fluid-structure interaction phenomena in intracranial aneurysms. *"Parallel Computing" 55* (2016), 9 – 16. Visualization and Data Analytics for Scientific Discovery. 3

[RASS17] ROCHA A., ALIM U., SILVA J. D., SOUSA M. C.: Decal-Maps: Real-Time Layering of Decals on Surfaces for Multivariate Visualization. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (Jan 2017), 821–830. 3

[RBN*19] RAITH F., BLECHA C., NAGEL T., PARISIO F., KOLDITZ

O., GÜNTHER F., STOMMEL M., SCHEUERMANN G.: Tensor Field Visualization using Fiber Surfaces of Invariant Space. *IEEE Transactions on Visualization and Computer Graphics 25*, 1 (Jan 2019), 1122–1131. 1, 3, 5, 7, 8, 9, 11

[RE19] RÖBER N., ENGELS J. F.: "In-Situ Processing in Climate Science". In *"High Performance Computing"* (Cham, 2019), Weiland M., Juckeland G., Alam S., Jagode H., (Eds.), "Springer International Publishing", pp. 612–622. 11

[Rob07] ROBERTS J. C.: State of the Art: Coordinated & Multiple Views in Exploratory Visualization. In *Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2007)* (2007), IEEE, pp. 61–71. 3

[SOC*19] SAKURAI D., ONO K., CARR H., NONAKA J., KAWANABE T.: Flexible Fiber Surfaces: A Reeb-Free Approach. In *Topological Methods in Data Analysis and Visualization V*, Carr H., Fujishiro I., Sadlo F., Takahashi S., (Eds.). 2019. 2

[SSA*19] STEVENS B., SATOH M., AUGER L., BIERCAMP J., BRETHERTON C. S., CHEN X., DÜBEN P., JUDT F., KHAIROUTDINOV M., KLOCKE D., ET AL.: DYAMOND: the DYnamics of the Atmospheric general circulation Modeled On Non-hydrostatic Domains. *Progress in Earth and Planetary Science 6*, 1 (2019), 61. 11

[SSC*16] SAKURAI D., SAEKI O., CARR H., WU H., YAMAMOTO T., DUKE D., TAKAHASHI S.: Interactive Visualization for Singular Fibers of Functions f : R3 → R2. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (Jan 2016), 945–954. 3

[STS06] SAUBER N., THEISEL H., SEIDEL H.: Multifield-graphs: An approach to visualizing correlations in multifield scalar data. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (Sep. 2006), 917–924. 2

[TC17] TIERNY J., CARR H.: Jacobi Fiber Surfaces for Bivariate Reeb Space Computation. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (Jan 2017), 960–969. 2

[TFL*18] TIERNY J., FAVELIER G., LEVINE J. A., GUEUNET C., MICHAUX M.: The Topology ToolKit. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (Jan 2018), 832–842. 2

[WKI*17] WU K., KNOLL A., ISAAC B. J., CARR H., PASCUCCI V.: Direct Multifield Volume Ray Casting of Fiber Surfaces. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (Jan 2017), 941–949. 2