Fast and Exact Fiber Surfaces for Tetrahedral Meshes

Pavol Klacansky, Julien Tierny, Hamish Carr, Zhao Geng

Fast and Exact Fiber Surfaces for Tetrahedral Meshes

Pavol Klacansky, Julien Tierny, Hamish Carr, and Zhao Geng

Abstract—Isosurfaces are fundamental geometrical objects for the analysis and visualization of volumetric scalar fields. Recent work has generalized them to bivariate volumetric fields with fiber surfaces, the pre-image of polygons in range space. However, the existing algorithm for their computation is approximate, and is limited to closed polygons. Moreover, its time performance does not allow instantaneous updates of the fiber surfaces upon user edits of the polygons. Overall, these limitations prevent a reliable and interactive exploration of the space of fiber surfaces. This paper introduces the first algorithm for the exact computation of fiber surfaces in tetrahedral meshes. It assumes no restriction on the topology of the input polygon, handles degenerate cases and better captures sharp features induced by polygon bends. The algorithm also allows visualization of individual fibers on the output surface, better illustrating their relationship with data features in range space. To enable truly interactive exploration sessions, we further improve the time performance of this algorithm. In particular, we show that it is trivially parallelizable and that it scales nearly linearly with the number of cores. Further, we study acceleration data-structures both in geometrical domain and range space and we show how to generalize interval trees used in isosurface extraction to fiber surface extraction. Experiments demonstrate the superiority of our algorithm over previous work, both in terms of accuracy and running time, with up to two orders of magnitude speedups and computations taking less than a second for each of data-sets. This improvement enables interactive edits of range polygons with instantaneous updates of the fiber surface for exploration purpose. A VTK-based reference implementation is provided as additional material to reproduce our results.

1 Introduction

Isosurfaces are geometric primitives that serve as the basis of many data-analysis and segmentation tasks. Regarding multivariate scalar fields, Scientific Visualization has historically had no counter parts to isosurfaces, even for bivariate fields. This was recently remedied by *fiber surfaces*, which generalize isosurfaces by taking the inverse image of a separating line, curve or polygon in the range [6]. Fiber surfaces have been shown to provide more flexible segmentation capabilities than sequences of isosurfacing/thresholding on the individual components of the data, as illustrated in Figure 1. However, due to its approximate nature and time requirement (several seconds of computation even for moderately small data-sets), the existing algorithm [6] currently prevents a usage of fiber surfaces as widespread as for isosurfaces. Thus, as with the original works on isosurfaces, two major issues need to be addressed: *(i)* accuracy (to allow for robust post-processing [15], [14]) and *(ii)* speed (to allow for interactive surface-based data exploration [25]).

This paper fills this gap by introducing the first algorithm that extracts provably exact fiber surfaces in tetrahedral meshes, with up to two orders of magnitude speedups in contrast to previous work [6]. Its result is shown to be exact, it assumes no restriction on the topology of the input polygon, handles degenerate cases and better capture sharp features induced by polygon bends. The algorithm also easily allows visualization of individual fibers on the output surface, which better illustrates their relationship with data features in range space. To reach interactive extraction rates, we investigate several speedup strategies. First, we show that our algorithm is trivially parallel and we report nearly linear scalings with the number of cores. Second, we generalize both domain-based and range-based isosurface extraction acceleration algorithms to fiber surfaces. In particular, we show how to generalize interval trees (widely used in isosurface extraction [8]) to the case of fiber surfaces and we describe how this problem reduces to the design of hierarchical partitioning data-structures efficiently supporting polygon collision detection tests. Experiments show the superiority of this approach with up to two orders of magnitude speedups over previous work and computations taking less than a second for each of our data-sets. Finally, we describe an interactive system for fiber surface exploration that combines and exploits these contributions.

Overall, our algorithm provides the robustness and speed required for a widespread usage of fiber surfaces, in automatic or interactive contexts. In the interest of reproducibility and rapid uptake of these methods, we provide a lightweight VTK-based C++ implementation as additional material that we hope will become a reference implementation for fiber surfaces. In summary, this paper makes the following new contributions:

1) **Accuracy and robustness:**
   - Exact extraction of fiber surfaces in tet-meshes;
   - Extension to input polygons of arbitrary topology.
2) **Interactive exploration:**
   - Generalization of domain-based and range-based isosurface acceleration methods to fiber surfaces;
   - Scalable parallel fiber surface extraction;
   - On-surface individual fiber visualization;
   - Interactive system for fiber surface exploration;
   - A VTK-based C++ reference implementation.

- *Klacansky is with the SCI Institute at the University of Utah, USA. E-mail: klacansky@sci.utah.edu*
- *Tierny is with Sorbonne Universites, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, France. E-mail: julien.tierny@lip6.fr*
- *Carr and Geng are with the University of Leeds, UK. E-mail: {h.carr, z.geng}@leeds.ac.uk*

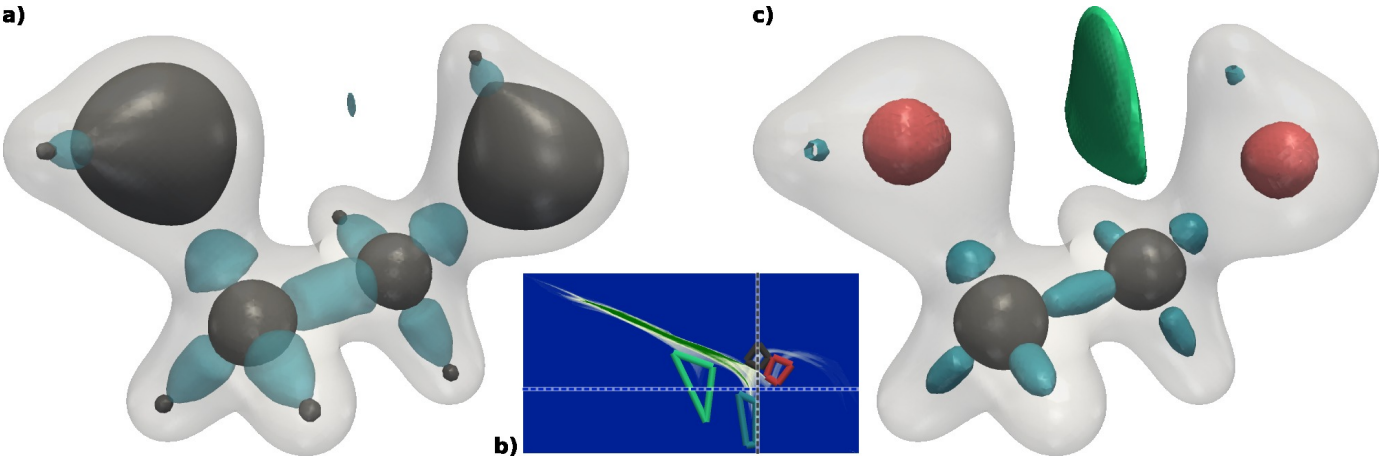

Fig. 1. Isosurfaces (a) and Fiber surfaces (c) of a bivariate field representing chemical interactions within an ethane-diol molecule ((b): continuous scatter plot, X: electron density, Y: reduced gradient [18]). While isosurfaces of the electron density capture regions of influence of atoms ((a): grey), they do not distinguish atom types. Similarly, isosurfaces of the reduced gradient capture regions of chemical interactions ((a): blue) but do not distinguish covalent from non-covalent interactions. In contrast, polygons isolating the main features of the continuous scatter plot (b) yield fiber surfaces (c) distinguishing atom types (red and grey) as well as interaction types (blue and green). Image adapted from [6].

2 Related Work

For this paper, there are three primary areas of relevant work: isosurface extraction, multifield visualization, and the recent paper introducing fiber surfaces [6]. For the former, we shall assume that the reader is broadly familiar with isosurface extraction except when the details are significant, otherwise directing the interested reader to a recent survey [25] and textbook [33]. For multifield visualization, we shall sketch the relevant literature, and use a separate section to sketch the principal results from the recent paper on fiber surfaces.

2.1 Isosurfaces

Given a scalar field $f : \mathbb{R}^3 \to \mathbb{R}$, contours and isosurfaces can be defined mathematically as the inverse image $f^{-1}(h) = \{x \in Dom f : f(x) = h\}$ of an *isovalue* $h \in Ran f$. For a simply connected domain, this has the useful property that it separates the domain into pieces: in particular, for many datasets, the isosurface is a closed surface which represents some sort of boundary in the phenomenon under study.

In practice, f is usually represented by a piecewise mesh with an interpolant over each cell of the mesh: extraction methods therefore depend on the type of cells.

For regular cubic meshes, a trilinear interpolant is normally assumed, for which the correct isosurfaces are hyperbolic sheets [26]. These, however, are expensive and difficult to extract and render, and in practice, a simpler approach is used.

Marching Cubes [22] therefore constructs a surface separately in each voxel, following four stages: I) classification (marking vertices as below or above the queried isovalue), II) triangle topology (given the previous classification, a lookup table is employed to retrieve the corresponding triangle mesh topology), III) vertex interpolation (given the previous triangle mesh, vertices' positions are obtained through interpolation), IV) normal vectors (given the triangle mesh embedding, its normals are computed). While efficient and easy to implement, the surfaces do not match the trilinear interpolant either topologically or geometrically [26], [15], [14].

Variants of this also exist for other mesh types, in particular for tetrahedral meshes with barycentric interpolation [4]. In this case, known as *Marching Tetrahedra*, the isosurface in a given cell is guaranteed to be planar and parallel to all other isosurfaces in the cell, and the surface extracted is thus known to be correct.

As a result of its simplicity, robustness and ease of implementation, Marching Cubes / Tetrahedra has become the standard approach for extracting isosurfaces. However, its cost of is $O(n)$ in the input size rather than $O(k)$ in the output size (the number of triangles). Since many techniques depend on interactive extraction of isosurfaces, considerable effort has therefore been devoted to accelerating Marching Cubes, in particular through parallelization, the adoption of geometric search structures, and through topological analysis. Of these three, parallelization is the simplest, since Marching methods compute independent surfaces in each cell of the mesh: thus, parallelisation is easily achieved, and carries over to fiber surfaces, as discussed in Section 8.

Geometric search for isosurface acceleration relies on the observation that only those cells which intersect a given isosurface (known as *active cells*) need to be processed. This can be restated by asking whether the desired isovalue h belongs to the image of each cell K in the range. Since for scalar fields, a cell's image is always an interval $[K_{min}, K_{max}]$, it can be stored in constant space, and tested with a point-in-interval intersection: is $h \in [K_{min}, K_{max}]$?

One of the simplest geometric search structures is the octree [24], in which the domain is recursively divided into octants. This was exploited for isosurface extraction by computing the image of each octant as the union of the images of its own octants, then storing the resultant interval at the corresponding node of the octree [34]. In searching the octree for cells intersecting a known isovalue h, any node whose interval does not include h can be ignored entirely.

In comparison, range-based queries such as span space [29] store each cell explicitly as an interval in a search structure, with nodes in the hierarchy generally representing isovalues. The most efficient range structure, the interval tree [11], [8], is a ternary

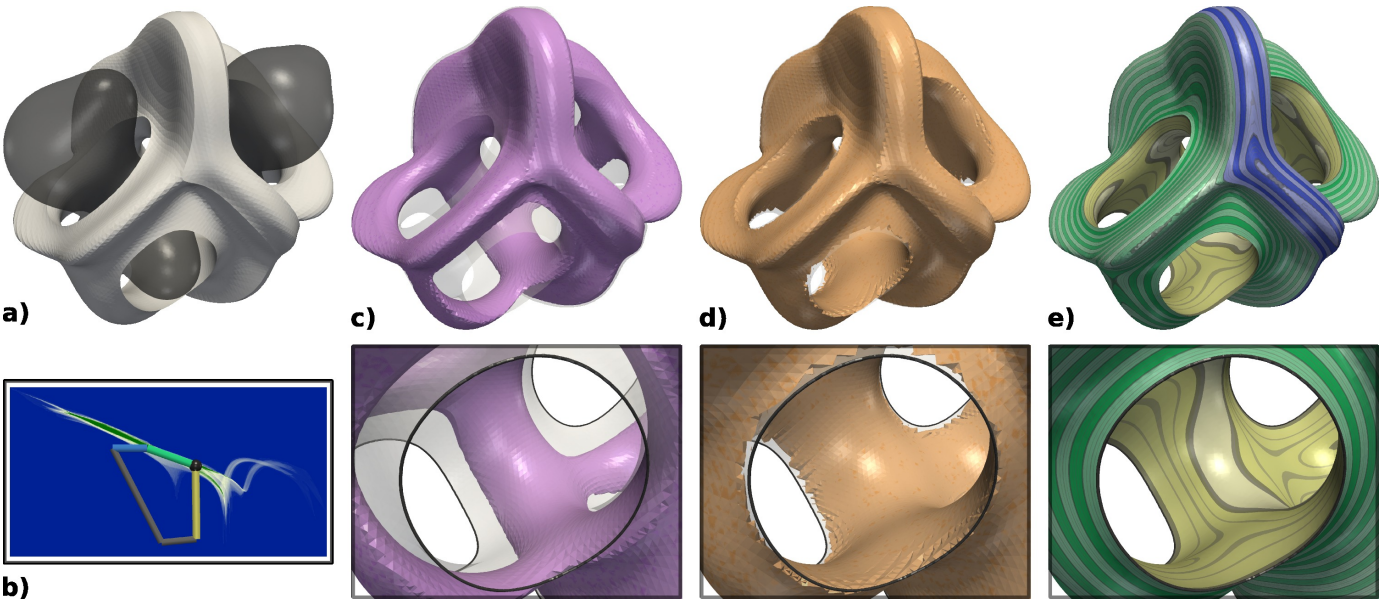

Fig. 2. Fiber surface extraction on a bivariate field (electron density and reduced gradient) representing chemical interactions within an ethane-diol molecule (dark surface in (a)). Fiber surfaces are defined as pre-images of polygons drawn in range space (i.e. the continuous scatter plot (b)). The existing algorithm for their computation [6] relies on a distance field computation on a rasterization of the range. While increasing the raster resolution results in more accurate fiber surfaces ((c): 16^2, (d): 1024^2), even for large resolutions, the distance field intrinsically fails at capturing sharp features of the fiber surface (here polygon bends in the range, black sphere (b)), as showcased in the zoom-views (bottom) where the corresponding fibers are displayed with black curves. Our work introduces the first algorithm for the exact computation of fiber surfaces on tetrahedral meshes. It accurately captures sharp features (e) and enhances fiber surfaces with polygon-edge segmentation (colors in (b) and (e)) and individual fibers (e, bottom) to better convey the relation between fiber surfaces and range features.

tree with an isovalue key and three child nodes at each node, of which the middle child stores cell intervals that contain the isovalue, and the other two store intervals below the isovalue and above the isovalue respectively. This allows the intersection test to be reduced to a set of scalar comparisons, allowing efficient descent through the tree. We will see in Section 9 that adapting these structures is non-trivial but possible, but will defer further discussion until fiber surfaces have been described.

Finally, the third branch of isosurface acceleration is based on topological analysis of the data to determine seed cells [31] from which propagation can be used to extract the isosurface [35]. For fiber surfaces, this depends on the topological analysis of functions of the form $\mathbb{R}^3 \to \mathbb{R}^2$, and while work has started on this [12], [5], it is not at present sufficiently advanced for use in fiber surface acceleration.

2.2 Multifield Visualization

Other than reduction to scalar fields or direct volume rendering, few general methods for bivariate visualization in $Dom f$ are known, except for the special case of complex-valued fields [32], where a complex value was chosen in the range of $f : \mathbb{C}^2 \to \mathbb{C}$, and the corresponding 2-manifold contour in $\mathbb{C}^2$ was constructed. If we treat $\mathbb{C}$ as $\mathbb{R}^2$, f can be restated as $f : \mathbb{R}^4 \to \mathbb{R}^2$, and these complex contours are then fibers of f, as described in the next section.

One method that is often used is to classify the data points statistically as "interior" or "exterior" then apply stage II. of Marching Cubes. However, this binary classification makes it difficult to apply stages III. and IV, which are usually resolved with heuristics[16], [28].

Multifields can be shown as multidimensional histograms, and recent work on continuous scatterplots [1] has shown the importance of the presumed mesh continuity. Subsequent work has focused on linear features [21] which are now [5] known to be related to the topology of the multifield. This has led to considerable work on the use of direct volume rendering (DVR) for visualizing two fields, commonly an isovalue and gradient pair. Since we do not rely on DVR in this paper, and the original fiber surface paper covers the use of DVR for bivariate visualization, we refer the interested reader to the treatment therein. For a broader view on visualization techniques for multivariate data, we refer the interested readers to a recent survey [20].

Recently, isosurfaces have been generalized to bivariate fields with the notion of fiber surface (pre-images of separating lines, curves or polygons in the range). However, the existing algorithm for their computation [6] is only approximate as it relies on a distance field computation on a rasterization of the range. While increasing the raster resolution results in more accurate fiber surfaces, even for large resolutions, the distance field intrinsically fails at capturing sharp features of the fiber surface (Figure 2 and 4). Moreover, our experiments show that it requires several seconds of computation even for moderately small data-sets, which prevents its usage in interactive exploration sessions where fiber surfaces should be instantaneously updated upon user edits of the input polygon.

3 Fibers and Fiber Surfaces

To generalise isosurfaces to bivariate fields, instead of taking a single value $h \in \mathbb{R}$, we take a single point $h \in \mathbb{R}^2 (= Ran f)$, and find its inverse image $f^{-1}(h) = \{x \in Dom f : f(x) = h\}$ to extract a *fiber* [27]. In the case of a bivariate volumetric field $f : \mathbb{R}^3 \to \mathbb{R}^2$, fibers are the intersection of the isosurfaces of each component

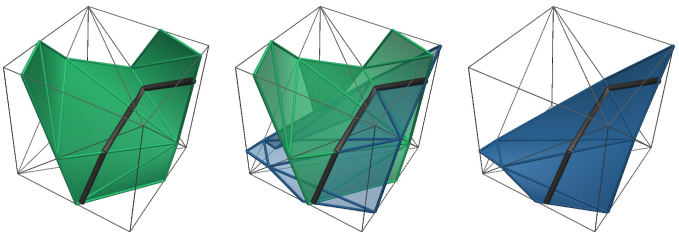

Fig. 3. Example of fiber construction. Left: an isosurface of f_1. Center: a fiber defined by the intersection of isosurfaces (black). Right: an isosurface of f_2. Both isosurfaces also show the fiber for reference.

of f, i.e. $f^{-1}(h) = f_1^{-1}(h_1) \bigcap f_2^{-1}(h_2)$, as illustrated in Figure 3. These are normally curves in space, and do not constitute 2-manifold boundaries the way isosurfaces do. This, however, can be remedied by taking the inverse image not of a 0-manifold point, but of a 1-manifold path in the range, which may be a curve, polyline or polygon.

If the curve separates the range of f, then the fiber surface separates the domain of f: i.e. it produces a boundary of some sort. Moreover, this leads to a simple algorithm [6] for extracting fiber surfaces: classify mesh vertices as inside or outside this boundary, then apply Marching Cubes tables to determine the local surface topology. Interpolating vertices along mesh edges is performed by computing the signed distance in the range from the curve to each vertex, and finding the zero-distance point along each edge. Finally, this computation can be accelerated by rasterising the distance field of the polygon for use as a lookup table. It therefore sufficed to deal with the case of a closed polygon, which we refer to as a *fiber surface control polygon* or FSCP.

Figure 4 illustrates configurations in tetrahedral meshes where the above strategy fails at capturing accurately the fiber surface. First, the interpolation based on the signed distance field fails at capturing bends in the FSCP, which are "shortcut" by its zero level-set (left). Note that since polygon bends are unlikely to coincide precisely in the range with the vertices of $Dom f$, this inaccuracy occurs for all bends. Second, the vertex classification (inside or outside) is insufficient when the FSCP is completely included within the image of a tetrahedron and that none of its vertices lie in the inside of the FSCP (middle). Third, an FSCP may cross the image of an edge of $Dom f$ multiple times, which may prevent the identification of intersections of the fiber surface with a tetrahedron, due the vertex classification (as illustrated in Figure 4, right). This latter configuration not only yields a poor approximation of the geometry of the fiber surface, but also an incorrect topology.

As discussed in the result section, these low-level configurations can have high-level impacts on the geometry and the topology of the extracted fiber surface. We describe in the following an algorithm that overcomes these difficulties and extract the correct fiber surface.

4 Correct Fiber Extraction

As noted in the previous section, a fiber in a bivariate volumetric field can be defined mathematically by the intersection of isosurfaces with respect to the two components of the field. In a function defined over a mesh, all that is required is to define

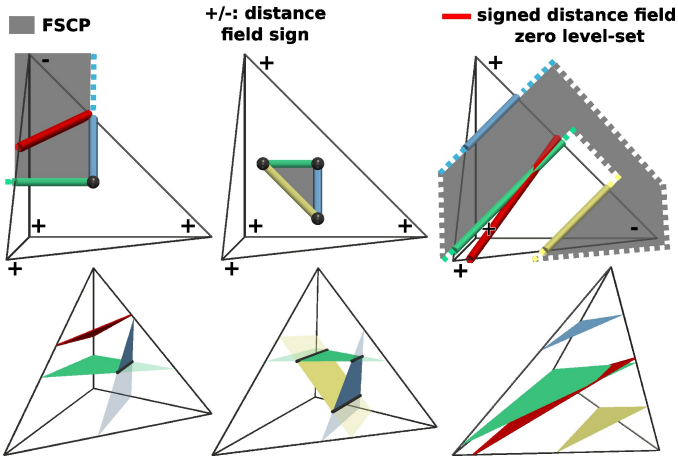

Fig. 4. Configurations inaccurately processed by a fiber surface extraction based on a signed distance field [6] (top: range, bottom: domain). Left: an FSCP bend lies inside a tetrahedron (black sphere). The resulting distance field yields a 0 level-set inaccurately capturing the fiber surface. Center: FSCP edges completely included inside a tetrahedron result in a distance field which yields an empty 0 level set. Right: an FSCP enters multiple times a tetrahedron. The corresponding distance field yields a 0 level-set which not only poorly approximates the fiber surface geometry but which also misses some connected components (blue and yellow).

a fiber for each cell separately. For a tetrahedral mesh with barycentric interpolation, this is straightforward, since we know that isosurfaces are simply planar cuts through the tetrahedron. If we therefore take one isosurface with respect to each component and intersect them, we expect to produce a line segment, as shown in Figure 3. Conveniently, any pair of fibers in a tetrahedron are co-planar and parallel within that plane, since the isosurface planes of each component are parallel to each other.

Instead of computing the intersection of two planes, we observe that a fiber is a contour line of the restriction of component 2 to an isosurface of component 1. We therefore compute the fiber by extracting the isosurface of component 1 explicitly using Marching Tetrahedra, interpolating the value of component 2 at each vertex of the resulting triangles, then using Marching Triangles to extract the exact fiber.

When we consider hexahedral cells with trilinear interpolation, however, this becomes impractical. To see this, recall that isosurfaces of the trilinear interpolant are hyperbolic sheets [26]. Thus, any given fiber is the intersection of two arbitrary hyperbolic sheets, and may have multiple connected components. Figure 5 (top) illustrates this: not only the fibers can be made of several connected components, but also their geometry is complex and cannot be accurately approximated with linear primitives.

Computing fibers for Marching Cubes cases is slightly easier, as each cell may have at most 5 triangles, leading to intersection tests between at most 25 pairs of triangles, but since the surfaces are not exact to the trilinear interpolant, this is hardly helpful. Finally, extracting an isosurface with Marching Cubes, then contouring the triangles to produce fibers may produce different results depending on which field we apply first (this ambiguity does not occur with Marching Tetrahedra).

When this is combined with FSCPs that induce an arbitrary number of intersections of a fiber surface with a given cube, it is clear that exact fiber surfaces for trilinear cubic meshes are

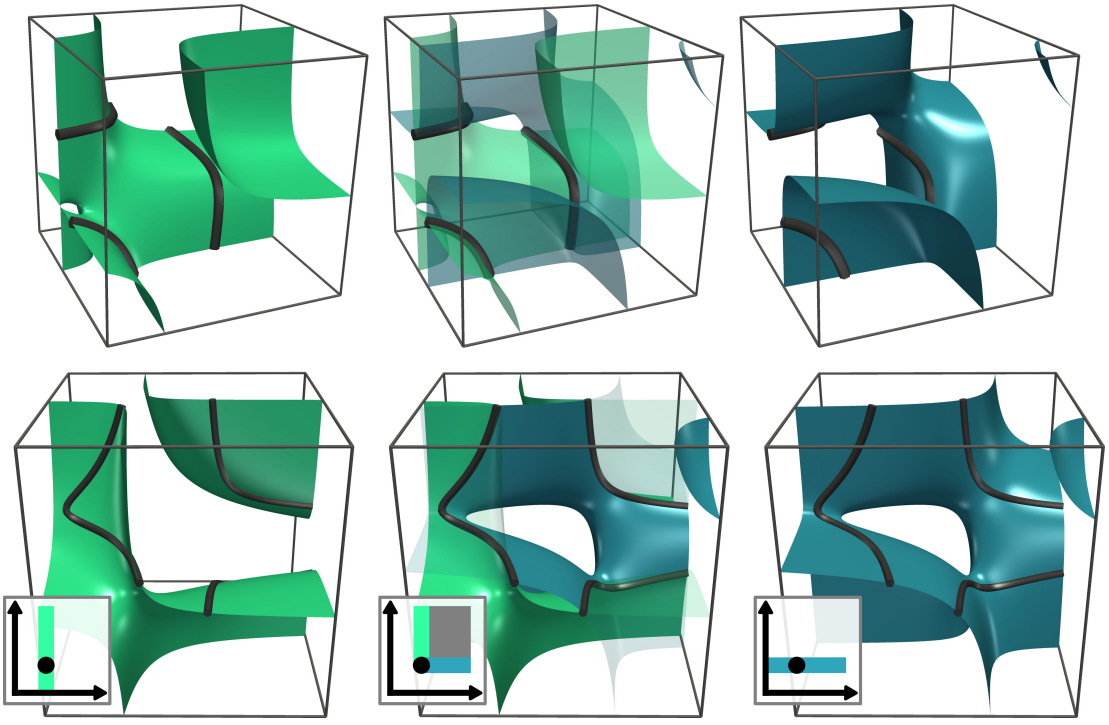

Fig. 5. Fiber surface extraction in a voxel with the trilinear interpolant. Top: fiber extraction (black curve), bottom: fiber surface extraction (inset: FSCP, black point: polygon bend). From left to right: isosurface of f_1 (green), fiber or fiber surface, isosurface of f_2 (blue). These results have been obtained with our algorithms on the tetrahedral mesh of an up-sampled voxel (256^3).

not presently tractable. Figure 5 (bottom) further exemplifies this with a simple, axis-aligned FSCP (bottom insets): due to the geometrical complexity of the trilinear interpolant, fiber surfaces can be possibly made of multiple connected components and their genus and the number of their boundary components can be large and can vary greatly, making their systematic extraction impractical.

5 Correct Fiber Surface Extraction

Once we can extract single fibers exactly, we look at exact extraction of fiber surfaces.

First, we observe that each line segment in an FSCP will locally induce planar segments of the fiber surface in each tetrahedron, as shown in Figure 6. For scalar fields, an isosurface can be interpreted as the zero level-set of the signed range distance field to the queried isovalue i: $f^{-1}(i) = \{p \in Domf \mid f(p) - i = 0\}$. Similarly, for bivariate functions, the pre-image of a line $\overleftrightarrow{l} \in Ranf$ can be interpreted as the zero level-set of the signed range distance field h to $\overleftrightarrow{l}$. Thus, as for any other scalar field, the pre-image of zero by h within each tetrahedron is indeed guaranteed to be planar due to the usage of the linear interpolant. In the following, we call such planar segments *base fiber surfaces*. This observation motivates the first stage of our algorithm (Algorithm 1).

Second, as seen in Figure 4, base fiber surfaces meet at the fibers induced by the vertices of the FSCP. We can therefore decompose the problem by considering each tetrahedron and each FSCP edge separately. In particular, to take an FSCP vertex v into account, one needs to clip the base fiber surfaces of each FSCP edge adjacent to v at the pre-image v. This observation motivates the second stage of our algorithm (Algorithm 2).

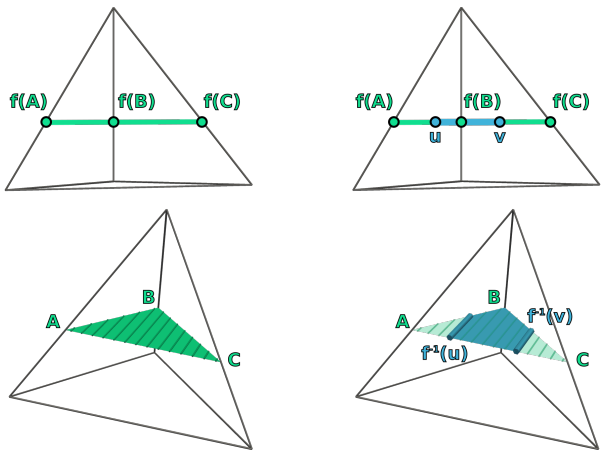

Fig. 6. Fiber surface extraction within a tetrahedron (top: range, bottom: domain). Left: Base fiber surface extraction (Algorithm 1, green surface). Right: Fiber clipping (Algorithm 2, thicker blue fibers, case 3 of Figure 7).

Therefore, our algorithm is composed of two stages (described in the following): base fiber surface extraction (Figure 6, left) and fiber clipping (Figure 6, right). In particular, each edge e of the FSCP is processed independently and for each of these, the tetrahedra of $Domf$ are traversed independently.

Base fiber surface extraction: Given a tetrahedron $T \in Domf$ and an FSCP edge $(u,v) \in Ranf$ living on a line $\overleftrightarrow{l}$, we ignore the endpoints u and v and extract the pre-image of $\overleftrightarrow{l}$ to produce the corresponding base fiber surface (in Figure 6, $\overleftrightarrow{l}$ is shown as a green line in the range (top)). This cut is found by the marching tetrahedra method by considering the zero level-set of the signed range distance field h to $\overleftrightarrow{l}$, using the Hesse normal form of the

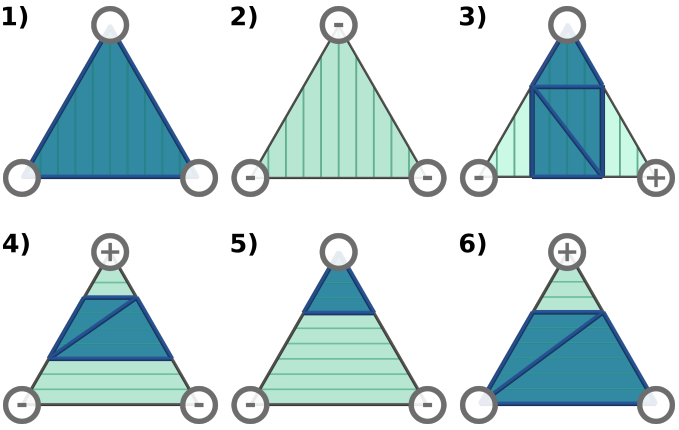

Fig. 7. Six rotationally and sign symmetric base cases for extracting fiber surface mesh (in red) from a single triangle. Plus denotes a vertex with $1 < t$, and minus $t < 0$. An empty circle denotes a vertex with $0 \leq t \leq 1$.

line (line 4 of Algorithm 1, where n and d stand for the line's unit normal and its distance to the origin respectively). Since the following stage relies on having correct function values f_1, f_2 for every vertex of the base fiber surface, we compute these values with linear interpolation when we extract the triangles.

Fiber clipping: We next clip the base fiber surface to obtain the segment between fibers $f^{-1}(u)$ and $f^{-1}(v)$. Given a triangle ABC of the base fiber surface, we recall that $A, B, C, f^{-1}(u)$ and $f^{-1}(v)$ are all coplanar in $Domf$ (and colinear in $Ranf$) in virtue of the linear interpolant yielding planar pre-images of the signed distance field h. The clipping procedure depends on whether $f(A), f(B)$ and $f(C)$ lie between u and v or not on $\overleftrightarrow{l}$. We parameterize $\overleftrightarrow{l}$ with u at $t = 0$ and v at $t = 1$, and test with linear interpolation the parameters t of $f(A), f(B), f(C)$ against $[0,1]$, such that $t < 0$ is interpreted as white (-), $t > 1$ as black (+), and $0 \leq t \leq 1$ is grey (=). For example, in Figure 6, $t(A) < 0$ (-), $t(C) > 1$ (+) and $0 \leq t(B) \leq 1$ (=). Since we have three vertices, each triangle has $3^3 = 27$ possibilities, and we can construct a lookup table with the base cases shown in Figure 7. Similarly to the Marching Triangles, such a lookup table enables to retrieve the appropriate fiber surface connectivity within each base fiber surface. The lookup table shown in Figure 7 has been constructed by enumerating all the 27 possibilities. For each possibility, a minimal triangulation of the set of points of the base fiber surface for which $0 \leq t \leq 1$ is verified has been performed. Note that the 27 possibilities can be retrieved from these 6 cases through rotations. For example, if A is white, B is grey and C is black, we get case 3, and extract the coloured portion of the triangle as the required segment of the fiber surface. Note that case 1 retains the entire triangle, while case 2 discards it.

We express this as shown in Algorithm 2, noting that this can be incorporated into Algorithm 1 if desired. We also note that the triangles used in Algorithm 2 were generated from lookup tables in Algorithm 1. Since no tetrahedra can have more than two triangles in its base fiber surface, and that the interpolant is guaranteed to be linear across the base fiber surface, it is possible to compute a lookup table with 3^4 entries, in which case some triangles can be combined. While we have done so in our implementation, this only reduces the number of triangles by about 2.5%, so we report the simpler solution for clarity.

Algorithm 1 Extracting Base Fiber Surface in Tetrahedron

Require: mesh M, functions $f = (f_1, f_2)$, line $\overleftrightarrow{l}$
1: **for all** tetrahedra $T \in M$ **do**
2: set case $C = 0$
3: **for all** vertices $w_i \in T$ **do**
4: compute vertex distance $h_i = n \cdot (f_1, f_2) - d$
5: **if** distance $h_i > 0$ **then**
6: set case $C = C|2^i$
7: **end if**
8: **end for**
9: **for all** triangles t in Marching Tetrahedra case C **do**
10: interpolate vertex positions
11: interpolate vertex values f_1 and f_2
12: **end for**
13: **end for**

6 Degenerate Cases

One of the practical difficulties with geometric algorithms is how to deal with degenerate cases. For isosurface extraction, Marching Cubes and Marching Tetrahedra assume a binary test: i.e. black vertices have $f \geq h$ while white vertices have $f < h$, or vice versa. This can be seen as a special case of simulation of simplicity [13], as it is equivalent to adding a small ε to the the function value before comparing with h.

For bivarate output, it is more difficult to have a simple robust test, so we instead use a ternary test [9], [2] (i.e. to check if a vertex is either *(i)* black (+), *(ii)* white (-) or *(iii)* grey (=)) in both phases of the algorithm. With this approach, the only concern that remains is a degenerate tetrahedron, all four of whose vertices belong to the inverse image. In this case, as with isosurfaces, there should be a volumetric bulge in the fiber surface.

However, unless all tetrahedra are degenerate, we are guaranteed a boundary between degenerate and non-degenerate tetrahedra. Each non-degenerate tetrahedron along this boundary will share three vertices with a degenerate tet, and the entire face will therefore be extracted for the base fiber surface. Thus, the boundary between degenerate and non-degenerate tetrahedra is guaranteed to be extracted without degeneracies, which is what is needed.

Algorithm 2 Fiber Clipping

Require: triangle $T = \{(w, (f_1, f_2)) | w \in Domf, (f_1, f_2) \in Ranf\}$, line segment $L = o + td$
1: mesh $M = \emptyset$
2: **for all** $(w_i, (f_1, f_2)) \in T$ **do**
3: set case $C = 0$
4: project (f_1, f_2) onto L
5: compute parameter t for vertex v
6: if $t < 0$, set $C = C + 0 * 3^i$ (minus)
7: if $0 \leq t \leq 1$, set $C = C + 1 * 3^i$ (neutral)
8: if $1 < t$, set $C = C + 2 * 3^i$ (plus)
9: **end for**
10: **for all** triangle T' in Fiber Segment case C **do**
11: interpolate vertex positions on edges of T
12: add T' to M
13: **end for**

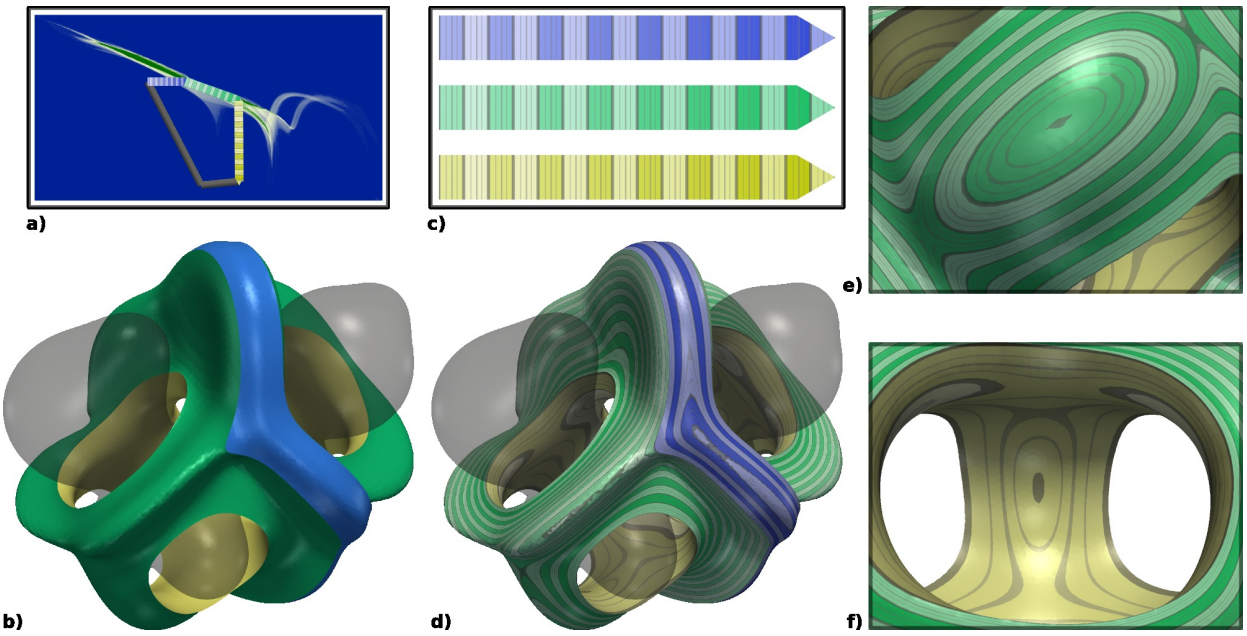

Fig. 8. Fiber surface texturing. (a) Continuous scatter plot. (b) Fiber surface segmented on a per FSCP edge basis (matching colors with (a)). (c) Employed textures. (d) The fiber-base texturing of the fiber surface provides further visual insights about the relation of the fibers constituting the surface and the corresponding points in the range, indicating possible transitions in the topology of fibers, (e) and (f).

7 Fiber Surface Textures

Sections 4 and 5 showed how to extract exact fibers and fiber surfaces. We next extend this to display fibers on a fiber surface, using colour to relate sections of the fiber surface to segments in the FSCP.

Since we extract portions of the surface separately for each FSCP segment, we can use the ID number of the segment to label each triangle extracted, then assign colours accordingly.

More generally, we observe that the FSCP is 1-parametrizable to the range $[0,1]$, either by assigning each vertex an integer, then normalizing, or by using a line-length parametrisation. Since the fiber surface is constructed from fibers, and all points on each fiber map to the same point on the FSCP, this can be used to assign colours or other properties to each fiber, using texture hardware on a video card.

Assigning a suitable texture parameter for each vertex of the fiber surface can be done easily given Algorithm 2. Recall that our algorithm processes each segment uv of the FSCP separately, and assigns u,v the parameters $0,1$ respectively. Since we then parameterize the vertices in each triangle to the same scale, it is then possible to map each vertex' parameter to the global range $[0,1]$ for use with the texture, as done in general with texture-based surface enhancement methods [17].

If we assign a different colour in the texture to each segment of the FSCP, we then get the same coloration as if we assign labels to each triangle based on the segment. More interestingly, we can store a dotted line in the texture map, with black values indicating a fiber to be drawn in black, and white values indicating no fiber. Combining these two ideas, as seen in Figure 8, simultaneously shows the viewer how the fibers change across the fiber surface, and which regions of the surface correspond to which values in the domain.

Two things now become visible: first, regions where the original fiber surface algorithm is inaccurate are precisely at sharp bends in the polygon, as shown in Figure 2. Second, the development of fibers across the surface indicates that topological analysis of the fibers is likely to provide further insight in the future.

8 Algorithm Complexity and Parallelism

As we observed in Section 2.1, fiber surfaces are nearly as parallelizable as Marching Tetrahedra, since the fiber surface is separately calculated in each cell of the mesh. From Section 5, we also see that the surface patch for each FSCP segment is separately calculated. As a result, we could on principle parallelize all cells and all FSCP segments, with $O(N \times E)$ independent calculations, where N and E stand for the number of tetrahedra and FSCP edges respectively. In practice, we expect E to be small, so we choose to parallelize over the cells, breaking them up into a number of regions based on the core count, then assigning each region to a separate thread.

Step 1: Our parallel algorithm starts by segmenting sequentially $Domf$ into n partitions of approximatively equal size.

Step 2: n threads are created. Each thread runs the fiber surface extraction algorithm (Algorithms 1 and 2) for its own region and progressively fills its own output surface data-structure.

Step 3: We now need to reduce n surface data-structures into one output. We sequentially allocate the output memory based on the sum of the number of triangles computed by each thread in step 2. In this process, we also identify n memory offset intervals, such that each interval will collect the triangles of a distinct thread.

Step 4: Finally, n threads copy the n sets of triangles computed in step 2 in each of the n offset intervals of the output data-structure.

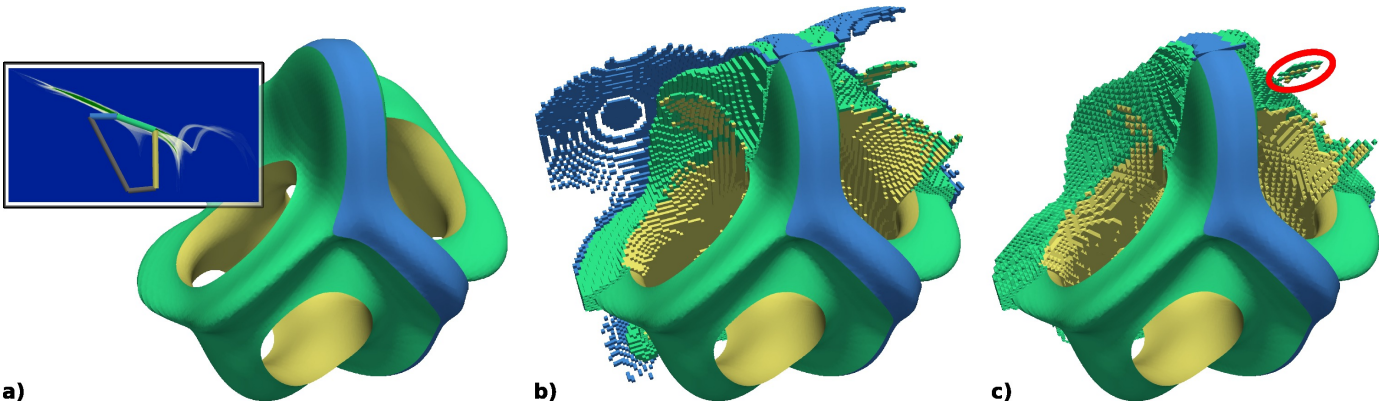

Fig. 9. Clipped view of the tetrahedra returned by our acceleration data-structures for each FSCP edge (matching colors). From left to right: (a) continuous scatter plot, FSCP, output fiber surface, queried tetrahedra with the octree (b, $\alpha = 0$) and the BVH respectively (c, $n_T = 1$).

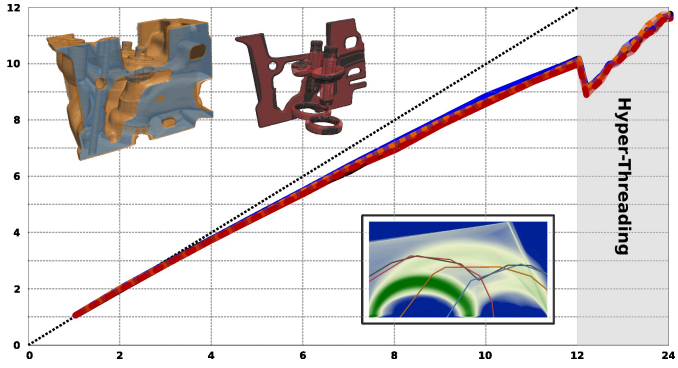

Fig. 10. Speedup of our parallel algorithm as a function of the number of threads on the up-sampled Engine data sets (285,927,495 tets). Each colored curve (continuous scatter plot, bottom right, X: scalar field, Y: gradient magnitude) corresponds to the fiber surface of the matching color (top left).

Note that this algorithm is fully parallel except for the synchronization at Step 3. In step 2, the threads only perform reading operations on the input data, hence requiring no synchronization between the threads. Similarly, no synchronization is required in Step 4 since each thread writes to distinct memory intervals.

9 Geometric Acceleration Techniques

Recall from Section 2.1 that geometric acceleration of isosurfaces can be reduced to point-in-interval tests by comparing the isovalue (a point) to the image of a cell (an interval). For fiber surface acceleration, the image of a tetrahedral cell in the range is known to be either a triangle or quadrilateral [30], or a more complex polygon for hexahedral cells [23]. For geometric search structures, the union of multiple such images will become a progressively more complex polygon, with inevitable implications for storage and runtime cost.

Since the FSCP is a polygon rather than an point, and the image of a cell is a polygon rather than an interval, this then replaces the simple point-in-interval inclusion test with a polygon-polygon intersection/inclusion test. While polygon-polygon intersection tests will be more common in practice, inclusion tests are still required since an FSCP can be completely included within the image of a tetrahedron while intersecting none of its edges (Figure 4, center). Once this is recognized, it becomes clear that general 2-D intersection tests are required, and the rich literature on collision detection can therefore be brought to bear on the problem.

In particular, polygon-polygon intersection tests can be replaced with a conservative test of axis-aligned bounding boxes of the polygons, albeit in the range of the function rather than the domain, at the expense of returning cells that do not intersect the fiber surface. We therefore show in the following how to extend two types of acceleration data-structures: domain-based (octree) and range-based (BVH), which allow us to reach interactive rates in our visualization.

9.1 Domain based acceleration data-structure

As we saw in Section 2.1, the octree can be used to store an interval representing the range of a scalar function at each node, then comparing the desired isovalue against this interval to determine which nodes can be discarded [34]. We have also observed that the corresponding exact test requires polygon-in-polygon tests, but can be replaced by a conservative test of axis-aligned bounding boxes in the range:

Off-line construction: The octree of $Domf$ is first computed in a top-to-bottom fashion, by recursive division of the domain into octants, yielding a tree data-structure [24]. At each node, we take the range bounding boxes (RBBs) of the child nodes, and compute the (min, max) with respect to each component in order to find the RBB of the entire node. For efficiency, we do not descend all the way to individual tetrahedra, instead providing a threshold n_T on the minimum number of tetrahedra per node, below which the base level RBB is computed from the vertices of the tetrahedra, but n_T can be set to 1 if desired.

On-line query: Given an FSCP edge e, the octree query starts at the root node and recursively visits each node's children only if the RBB intersects or overlaps e. Thus, a conservative over-estimate of the active cells is provided as input for fiber surface extraction (Algorithms 1 and 2). If an unneeded cell is returned by the octree, the ordinary operation of the fiber surface extraction will discard it in any event, so no additional geometry will be created.

The octree is not necessarily balanced in general, except for regular grids, and sub-trees can be arbitrarily deep (depending on the threshold n_T). To account for this, we use an additional

TABLE 1
Timings of our algorithms measured in seconds (1 thread).

Data set	Tets	Polygon	Pre-processing			Extraction			Output	Manifold	Fiber	Polygon
		Edges	Regular	Octree	BVH	Regular	Octree	BVH	Triangles	Post-processing	Texture	Segmentation
Tooth — Blue polygon	7,588,800	4	0	4.926	1.361	**0.790**	**0.277**	**0.164**	280,394	0.189	0.046	0.034
Tooth — Red polygon	—	5	—	—	—	**0.882**	**0.114**	**0.044**	48,974	0.035	0.011	0.005
Tooth — White polygon	—	5	—	—	—	**0.969**	**0.238**	**0.126**	226,248	0.150	0.004	0.011
Tooth — Yellow polygon	—	5	—	—	—	**0.962**	**0.249**	**0.129**	144,644	0.086	0.014	0.003
EthaneDiol — Black polygon	8,718,150	4	—	4.596	1.545	**0.783**	**0.020**	**0.011**	18,760	0.073	0.008	0.004
EthaneDiol — Blue polygon	—	4	—	—	—	**0.809**	**0.020**	**0.011**	11,252	0.009	0.014	0.017
EthaneDiol — Green polygon	—	3	—	—	—	**0.631**	**0.008**	**0.004**	10,956	0.005	0.011	0.017
EthaneDiol — Red polygon	—	4	—	—	—	**0.819**	**0.018**	**0.008**	12,724	0.016	0.008	0.015
EthaneDiol — Teaser polygon	—	5	—	—	—	**1.151**	**0.388**	**0.177**	259,160	0.148	0.009	0.006
Combustion	18,675,345	4	—	12.484	4.165	**1.898**	**0.321**	**0.177**	260,680	0.169	0.018	0.025
Engine — Black polygon	35,438,625	11	—	17.050	6.857	**9.121**	**0.741**	**0.341**	720,608	0.537	0.426	0.174
Engine — Blue polygon	—	6	—	—	—	**5.696**	**1.204**	**0.704**	1,734,388	1.284	0.212	0.022
Engine — Orange polygon	—	8	—	—	—	**7.438**	**1.425**	**0.798**	1,775,262	1.177	0.069	0.068
Engine — Red polygon	—	7	—	—	—	**6.103**	**0.649**	**0.325**	545,402	0.362	0.161	0.058
Enzo	82,906,875	8	—	39.052	16.451	**17.264**	**2.455**	**1.585**	3,460,562	2.486	0.392	0.302

termination criterion during off-line construction, stopping the recursion if a node's RBB is smaller than a fraction α of the RBB of the entire mesh. This criterion avoids deep sub-trees for parts of the mesh which concentrate to a small region of the range, yielding fewer line-bounding-box intersection tests and therefore faster online queries in these regions.

9.2 Range based acceleration data-structure

As discussed above, querying in the range for the set of cells that intersect a FSCP can be reduced to a collision detection test in 2-D. We therefore apply one of the most efficient collision detection tests, based on bounding volume hierarchies (BVH) [19], [10].

Off-line construction: The BVH of *Domf* is first computed in a top-to-bottom fashion, by recursively splitting the RBB in the middle along the horizontal and vertical axes. This is performed n_S times for each node, yielding a n_S-ary tree. In particular, each node of the BVH is given the list of tetrahedra whose RBB is completely included in its own RBB. The recursion stops if a node is given fewer tetrahedra than a given threshold n_T. Note that this data-structure differs from a quad-tree, as each node updates its RBB after its list of tetrahedra has been transferred from its parent node and before creating children nodes. This yields less regular but more refined range subdivision patterns.

On-line query: The query on the BVH is similar to that of the octree. Given a FSCP edge e, the query starts at the root and recursively visits each node's children if their RBB intersects or overlaps e. Only tetrahedra returned by the BVH are used for fiber surface extraction.

As with the octree, the BVH does not encode the precise polygonal projection of the tetrahedra (but only the RBBs), so it can also return tetrahedra that are not intersected by the fiber surface.

10 Experimental Results

In this section, we present experimental results obtained with a VTK-based (version 6.1) C/C++ implementation of our algorithms. Our implementation was evaluated on a desktop computer with two Xeon CPUs (2.6 GHz, 6 cores each) with 64 GB of RAM. Parallelism was implemented with OpenMP. All of our data sets are tetrahedral meshes obtained with 5-subdivisions of regular grids. All continuous scatter plots were computed using the original authors' implementation [1].

TABLE 2
Statistics for various computation parameters of our accelerating data-structures (1 thread, EthaneDiol data-set).

Parameters	Memory (MB.)	Pre-process (s.)	Queried Tetrahedra	Extraction (s.)
Octree ($\alpha = 0$)	768.810	4.345	11.835%	0.388
Octree ($\alpha = 0.001$)	649.736	2.362	35.260%	0.311
Octree ($\alpha = 0.005$)	634.104	1.799	58.984%	0.331
BVH ($n_T = 1$)	477.109	2.106	7.460%	0.171
BVH ($n_T = 8$)	116.728	1.478	12.654%	0.177
BVH ($n_T = 16$)	77.266	1.366	16.320%	0.203

10.1 Performance

Table 1 reports the execution times of our sequential implementation for various data sets and various user-defined FSCP. Our non-accelerated algorithm (column "Regular") has a time complexity of $O(N \times E)$ steps (N: number of input tets, E: number of FSCP edges). This complexity is verified in practice for a given data set as E increases (Tooth, Engine), and for a constant value of E across data sets of increasing size (Tooth - Blue polygon VS Combustion or Engine - Orange polygon VS Enzo).

Since our core algorithm extracts a triangle soup, it may be suitable to turn it into a manifold surface. This has been achieved by merging coincident points (using VTK's vtkMergePoints class). Alternatively, one could store in a map the list of output vertices for each input tet and use this information in a post-process. Throughout our tests, this feature has always been executed as an optional post-process. As detailed in Table 1 (column "Manifold post-processing"), this step takes a linear time with the size of the output. Finally, the visualization strategies discussed in Section 7 also require an overhead scaling approximatively linearly with the size of the output.

As expected, our acceleration algorithms (column "Octree" and "BVH") provide significant speedups, especially for small fiber surfaces which intersect only few tetrahedra (such as EthaneDiol - Green polygon). For the other data sets, these algorithms always improve over the non-accelerated algorithm, with average speedups of 18 and 36 for the octree and the BVH respectively.

Table 2 further investigates the behavior of our accelerating data-structures for our sequential algorithm. Since our input data sets

TABLE 4
Time performance comparison with [6] (measured in seconds).

Data set	Tets	Polygon Edges	Raster algorithm [6] 1024^2	Our algorithms (1 thread) Regular	Octree	BVH	**Max Speedup**	Our algorithms (24 threads) Regular	Octree	BVH	**Max Speedup**
Tooth — Blue polygon	7,588,800	4	2.080	0.790	0.250	0.157	**13**	0.072	0.167	0.037	**56**
Tooth — Red polygon	—	5	3.624	0.882	0.113	0.047	**77**	0.130	0.095	0.021	**173**
Tooth — White polygon	—	5	3.632	0.969	0.244	0.147	**25**	0.105	0.172	0.038	**96**
Tooth — Yellow polygon	—	5	2.388	0.962	0.245	0.161	**15**	0.106	0.195	0.041	**58**
EthaneDiol — Black polygon	8,718,150	4	2.162	0.783	0.020	0.011	**197**	0.069	0.021	0.013	**166**
EthaneDiol — Blue polygon	—	4	2.165	0.809	0.006	0.005	**433**	0.068	0.011	0.012	**197**
EthaneDiol — Green polygon	—	3	2.100	0.631	0.006	0.004	**525**	0.079	0.009	0.011	**233**
EthaneDiol — Red polygon	—	4	2.123	0.819	0.016	0.008	**265**	0.087	0.018	0.013	**163**
EthaneDiol — Teaser polygon	—	5	2.991	1.151	0.311	0.203	**15**	0.107	0.146	0.050	**60**
Combustion	18,675,345	4	4.468	1.898	0.299	0.169	**26**	0.160	0.227	0.041	**109**
Engine — Black polygon	35,438,625	11	13.121	9.121	0.785	0.370	**35**	0.879	0.506	0.090	**146**
Engine — Blue polygon	—	6	10.256	5.696	1.238	0.759	**14**	0.534	0.692	0.136	**75**
Engine — Orange polygon	—	8	11.277	7.438	1.351	0.810	**14**	0.804	0.870	0.167	**68**
Engine — Red polygon	—	7	10.590	6.103	0.603	0.366	**29**	0.613	0.415	0.085	**125**
Enzo	82,906,875	8	24.549	17.264	2.387	1.897	**13**	1.413	1.439	0.317	**77**

TABLE 3
Quantitative comparison with [6] for varying raster resolutions (EthaneDiol data-set).

Raster Resolution	Time (s.) Distance Field	Isosurface	Hausdorff Distance	Average Distance
16^2	0.555	1.242	5.268%	0.657%
32^2	0.523	1.214	3.00%	0.203%
64^2	0.523	1.220	1.752%	0.081%
128^2	0.540	1.222	1.256%	0.036%
256^2	0.587	1.220	1.212%	0.025%
512^2	0.802	1.222	1.027%	0.021%
1024^2	1.640	1.226	1.007%	0.020%
Exact signed distance field	3.471	1.227	1.007%	0.024%

have been obtained through a 5-subdivision of regular grids, the minimum number of tets per octree leaf has been set to 5 (corresponding to a single voxel). Due to this, the octree will necessarily return more candidate tets than necessary. For example, the range bounding box of a leaf can be intersected by a FSCP edge while none of its tets are actually intersected. The parameter α (Section 9.1) can be used to improve the depth of the octree, resulting in smaller memory footprint, shorter construction times and faster queries at the expense of returning more tets to the fiber surfacing procedure. We found in practice that $\alpha = 0.001$ offered the best trade-off. Since it is not domain-based, the BVH data-structure better captures the geometry of the input polygon in range-space, resulting in fewer returned tetrahedra, smaller memory requirements and faster queries. Similarly to the octree, the BVH depth can be tuned by adjusting the parameter n_T and n_S, which appeared to offer a best trade-off across all data sets for $n_T = 8$ and $n_S = 8$.

Figure 9 shows the tetrahedra returned by our accelerating data-structures, set up with parameters maximizing their depth. As suggested by Table 2, the BVH data-structure returns a sub-portion of the volume which better approximates the output surface. However, as discussed in Section 9.2, since the BVH does not encode precisely the polygonal range projection of each tetrahedron (but its range bounding box), it can still return tetrahedra which are not intersected by the fiber surface, as highlighted with the red ellipse.

Figure 10 reports the scaling performances of our parallel non-accelerated algorithm. For this experiment, we considered an up-sampled version of the Engine data set (512x512x220) prior to its 5-subdivision into a tetrahedral mesh (yielding 286 million tets). In practice, visited tetrahedra which are not intersected by the fiber surface will still be processed faster than intersected tetrahedra since no triangle will be created in the output. Also, whereas the threads are balanced input-wise, there is no guarantee that each thread produces the same number of triangles. This can lead to threads idling faster than others in the reduction step (step 4). Despite this, as showcased in Figure 10, our algorithm scales nearly linearly with the number of cores, irrespectively of the size of the output, achieving a maximum speedup of 11.71 in the hyper-threaded regime of our 12 cores, for a maximum throughput of 79 million tets per second.

10.2 Quantitative comparison

In this subsection, we provide a quantitative comparison with the existing algorithm for fiber surface extraction [6]. In particular, this algorithm approximates the fiber surface by extracting the 0 level-set in $Dom f$ of the signed range distance field to the FSCP. A faster approximation is also proposed in [6], where the authors perform a rasterization of the range, yielding fewer distance field evaluations and hence faster extractions. In our experiments, we adapted this algorithm, that we will call "Raster Algorithm", to tetrahedral meshes by using Marching Tetrahedra tables instead of Marching Cubes. We also evaluated the distance field value of each vertex of $Dom f$ in the range raster using bilinear interpolation. This yields smoother results than piecewise constant interpolation (as employed in the original algorithm) even for low raster resolutions.

Table 3 reports computation times for the raster algorithm (for the data set illustrated in Figure 2) as well as distance evaluations between its output and that of our algorithm (measured with the Hausdorff distance and the average of the minimum distance, expressed as a percentage of the bounding box diagonal). This table shows that, for small raster resolutions, the raster algorithm spends more time projecting the vertices of $Dom f$ into the range raster than evaluating the distance field: the timings in the column "Distance field" only starts to significantly increase for a resolution of 256^2. As predicted by the approximation nature of this algorithm, the Hausdorff distance to our exact computation decreases as the raster resolution increases (column "Hausdorff Distance"). The line "Exact signed distance field" reports the

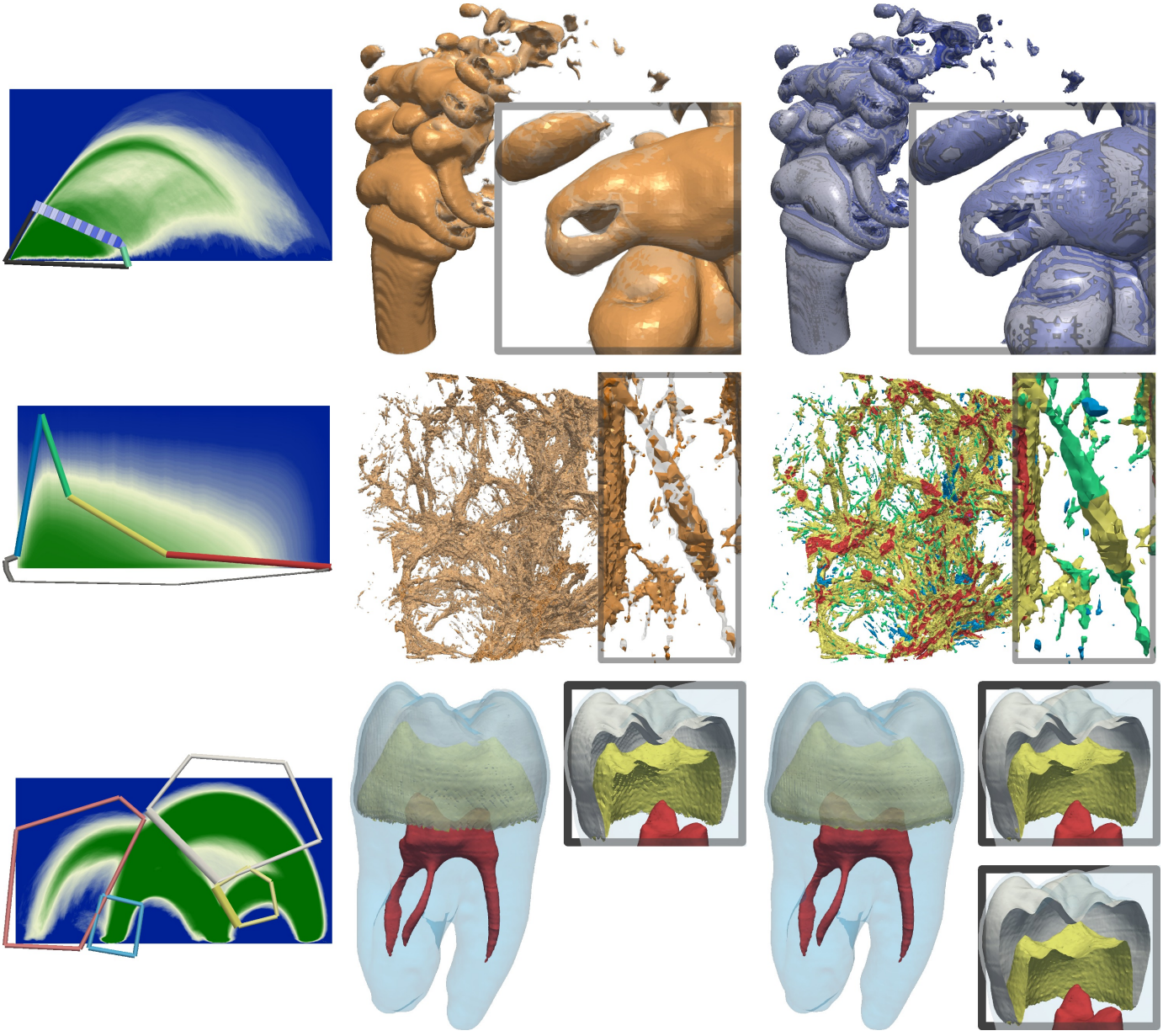

Fig. 11. Visual comparison with the raster algorithm [6]. Left: continuous scatter plot and FSCP. Center: fiber surfaces extracted with the raster algorithm. Right: fiber surfaces extracted with our algorithm.

statistics for the full signed range distance field computation (using no range raster), resulting in particular in slower running times.

Even for the exact range signed distance field, the Hausdorff distance to our result reaches a lower bound (1% of the bounding box diagonal). This is due to the fact that the distance field intrinsically fails to capture the configurations illustrated in Figure 4, in particular sharp surface features corresponding to FSCP bends. In practice, we saw that this lower bound was already reached at a raster resolution of 1024^2 (and did not improve with higher resolutions). We therefore use in the remainder a raster resolution of 1024^2 (implying faster computations, for an output of equivalent approximation quality).

Figure 2 provides a visual interpretation of Table 3. In particular, at a resolution of 16^2 (c), the fiber surface obtained with the raster algorithm (in purple) is far from our output (transparent). As the raster resolution increases, this distance decreases. For a resolution of 1024^2 (d), the only visual differences between the raster output (in orange) and that of our algorithm (transparent) occur in the vicinity of the fibers corresponding to FSCP bends. In particular, we illustrated these with black curves. While the orange surface produces a non-smooth surface that fails at capturing these features, our algorithm extracts them perfectly (e) while generating a smooth output.

Table 4 provides run-time comparisons between the raster algorithm and our algorithms. Our non-accelerated sequential algorithm (column “Regular”, 1 thread), was faster than the raster algorithm for all data sets (for an average speedup of 55%). This can be partly explained by the fact that the raster algorithm needs to compute the sign of the distance field, which requires an additional step to test if the vertices of $Dom f$, once projected in the range, are included within the FSCP.

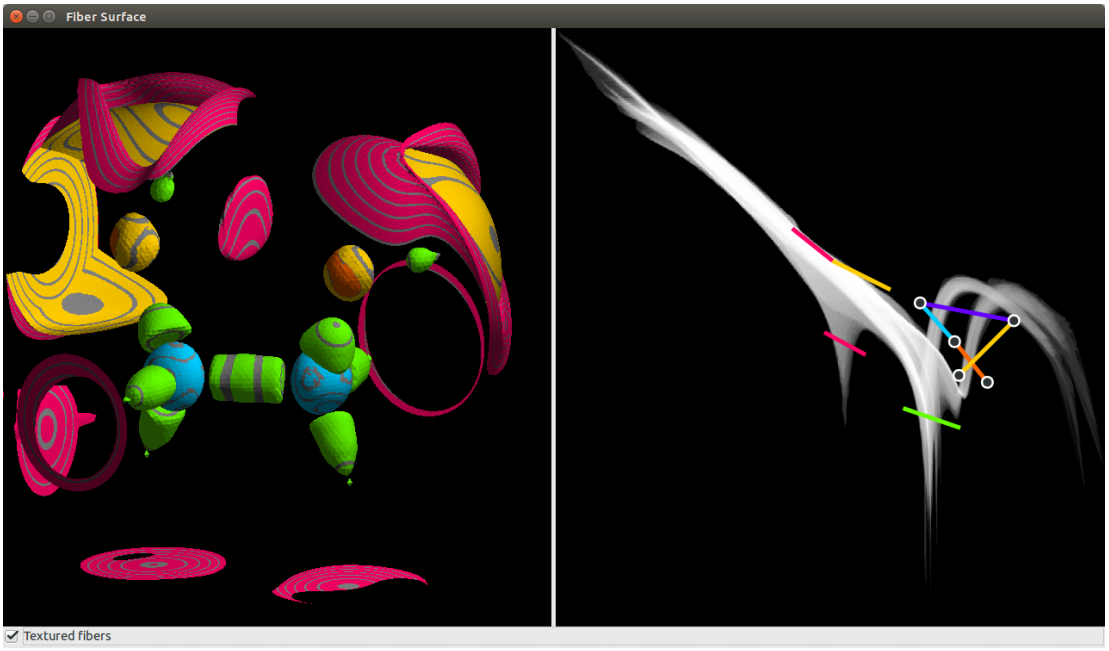

Fig. 12. Screen capture of our real-time fiber surface exploration user interface (left: fiber surface, right: continuous scatter plot and FSCP).

Our acceleration data-structures further improve our speedup in sequential mode, with an average speedup of 113. Further, we evaluate our parallel algorithm combined with our acceleration data-structures. For few data sets (especially for small fiber surfaces intersecting only few tetrahedra), the performance did not necessarily improve, since setting up the threads and reducing the result takes most of the time in these cases. Globally, our parallel algorithm combined with our acceleration data-structures provides the best time performance, leading to computations occurring in less than a second for all data sets (with BVH acceleration), for an average speedup of 120.

We therefore conclude that our algorithm is more accurate as well as faster, improving running time by up to 2 orders of magnitude.

10.3 Qualitative comparison

In this section, we provide further qualitative comparisons between the raster algorithm and our approach. Figure 11 provides side-by-side comparisons on several data sets: combustion (top row, X: scalar field, Y: gradient magnitude), enzo (middle row, X: matter concentration, Y: dark matter concentration), tooth (bottom row, X: scalar field, Y: gradient magnitude). As discussed in the previous subsection, the raster algorithm provides inaccurate geometrical approximations (orange surfaces, top and middle rows, center) in comparison to our algorithm (transparent surface), which can lead to disconnected structures, especially in the vicinity of polygon bends, as illustrated with the zoom-in views of the enzo data set, where polygon bends correspond to boundaries between segments of different colors (middle row, right). In these two data sets (combustion and enzo), our fiber surface texturing enables the visual identification of the location in the range of the individual fibers constituting the fiber surface (top row) or of the FSCP edges responsible for segments of the surface (middle row). The tooth data set (bottom row) illustrates the ability of fiber surfaces to segment material boundaries based on intensities and gradient magnitude. Such a segmentation was already achieved qualitatively in volume rendering with multi-dimensional transfer functions. However, fiber surfaces enable the explicit geometrical extraction of these boundaries. While our approach (right) tends to produce smoother surfaces than the raster algorithm (center), it also has the ability to handle non-closed FSCPs (in contrast to the raster algorithm). This is illustrated in the bottom right zoom-in view, where only the fiber surface corresponding to thick FSCP edges (left) have been extracted, yielding distinct open surfaces (white and yellow) revealing the boundary between the enamel and distinct materials. Note that, since our algorithm processes the FSCP on a per edge basis, it actually handles FSCPs of arbitrary topology. This is further exemplified in the accompanying video, where even self-intersecting polygons are demonstrated.

10.4 Interactive Fiber Surface Exploration

As documented in Section 10.2, our approach provides an overall speedup of up to two orders of magnitude over the raster algorithm, for an exact output. Such speedups enable processing times below a second for all of our data sets. This makes it possible to derive a user interface for the interactive exploration of fiber surfaces, with real-time updates of the surface upon user edits of the FSCP. Such an interface is illustrated in Figure 12 and further demonstrated in the accompanying video, which has been captured on a commodity laptop (Core2 Duo CPU at 2.4 GHz, 4 GB of RAM and an AMD 3650 mobility GPU). In contrast to the raster approach, our algorithm can process the FSCP on a per edge basis. Thus, since the user edits only a finite number of FSCP edges at a time, only the corresponding fiber surface patches are updated in the 3D view, which further improves the response time of the system. As illustrated in the accompanying video, our extraction algorithm enables instant updates of the fiber surface, allowing for a fully unconstrained exploration of the space of possible fiber surfaces.

10.5 Limitations

Our algorithm relies on the linear interpolant of tetrahedral meshes to extract an exact fiber surface. Other meshes and in particular hexahedral meshes have different interpolants, which can be handled by the approximate algorithm [6] or by subdivision into tetrahedra. However, different tetrahedral subdivision schemes will induce different linear interpolants [7], which will lead to topologically and geometrically different fiber surfaces. Future work will be required to obtain exact fiber surfaces directly (without tetrahedral subdivision), but as demonstrated in Section 4, this is likely to be a non-trivial task in its own right.

We also observe that the fiber surface does not strictly depend on the continuous scatterplot, which is used as the interface to define the FSCP. Since the ethanediol data set in particular has narrow spikes that are hard to capture manually in the continuous scatterplot, this indicates that automated definition or improved interfaces are worth examining in more detail.

11 Conclusion and Future Work

In this work we introduced the first algorithm for exact extraction of fiber surfaces in tetrahedral meshes. In contrast to the existing algorithm, our approach has no restriction regarding the topology of the control polygon, it has no parameter (such as the range raster resolution) and it handles degenerate cases. We showed that it is trivially parallelizable and scales nearly linearly with the number of cores. We described two acceleration strategies based on hierarchical data-structures. Overall, our approach improved previous work by up to two orders of magnitude at run-time, enabling real-time edits of the control polygon, with instantaneous updates of the fiber surface. We also provide as additional material a VTK-based source code that we hope will become a reference implementation for fiber surfaces.

Several future directions are apparent. Due to its resemblance to Marching Tetrahedra, our approach can be further improved with any of Marching Tetrahedra's extensions (such as dual contouring). Since we handle control polygons of arbitrary topology, it brings the necessary robustness for use with automatic range feature analysis (such as ridge extraction on the continuous scatter plot). A natural future direction is the extension of this work to higher dimensional data (both for the domain and the range), as investigated for isosurfaces [3], but also considering time-varying bivariate data.

Acknowledgements

Acknowledgements are due to EPSRC grant EP/J013072/1 for funding this work at Leeds, and to the grants LABEX Calsimlab ANR-11-LABX-0037-01 and DGA DT-SCAT-DA-IDF FD1300034MNRBC at UMPC.

References

[1] S. Bachthaler and D. Weiskopf. Continuous Scatterplots. *IEEE Transactions on Visualization and Computer Graphics (Proc of IEEE VIS)*, 2008.

[2] D. C. Banks and S. Linton. Counting Cases in Marching Cubes: Toward a Generic Algorithm for Producing Substitopes. In *Proceedings of IEEE Visualization*, pages 51–58. IEEE, 2003.

[3] P. Bhaniramka, R. Wenger, and R. Crawfis. Isosurface construction in any dimension using convex hulls. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):130–141, 2004.

[4] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341–355, 1988.

[5] H. Carr and D. Duke. Joint Contour Nets. *IEEE Transactions on Visualization and Computer Graphics*, 2013.

[6] H. Carr, Z. Geng, J. Tierny, A. Chattopadhyay, and A. Knoll. Fiber surfaces: Generalizing isosurfaces to bivariate data. *Computer Graphics Forum (Proc. of EuroVis)*, 2015.

[7] H. Carr, T. Möller, and J. Snoeyink. Artifacts Caused by Simplicial Subdivision. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):231–242, March/April 2006.

[8] P. Cignoni, P. Marino, C. Montani, E. Puppo, and R. Scopigno. Speeding up isosurface extraction using interval trees. *IEEE Transactions on Visualization and Computer Graphics*, 1997.

[9] P. Cignoni, C. Montani, and R. Scopigno. Tetrahedra based volume visualization. In *Mathematical Visualization*, pages 3–18. Springer, 1998.

[10] H. Dammertz, J. Hanika, and A. Keller. Shallow Bounding Volume Hierarchies for Fast SIMD Ray Tracing of Incoherent Rays. In *Proceedings of the Nineteenth Eurographics Conference on Rendering*, EGSR '08, pages 1225–1233, 2008.

[11] H. Edelsbrunner. Dynamic Data Structures for Orthogonal Intersection Queries. Technical report, Inst. Informationsverarb, Tech. Uniz. Graz, Graz, Austria, 1980.

[12] H. Edelsbrunner and J. Harer. Jacobi Sets of Multiple Morse Functions. In *Foundations in Computational Mathematics*, pages 37–57, Cambridge, U.K., 2002. Cambridge University Press.

[13] H. Edelsbrunner and E. P. Mucke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 1990.

[14] T. Etiene, L. Nonato, C. E. Scheidegger, J. Tierny, T. J. Peters, V. Pascucci, R. M. Kirby, and C. T. Silva. Topology verification for isosurface extraction. *IEEE Transactions on Visualization and Computer Graphics*, 2012.

[15] T. Etiene, C. E. Scheidegger, L. G. Nonato, R. M. Kirby, and C. T. Silva. Verifiable visualization for isosurface extraction. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2009.

[16] R. Huang and K.-L. Ma. RGVis: region growing based techniques for volume visualization. In *11th Pacific Conference on Computer Graphics and Applications*, pages 355–363, 2003.

[17] V. Interrante, H. Fuchs, and S. M. Pizer. Conveying the 3d shape of smoothly curving transparent surfaces via texture. *IEEE Transactions on Visualization and Computer Graphics*, 1997.

[18] E. R. Johnson, S. Keinan, P. Mori-Sanchez, J. Contreras-Garcia, A. J. Cohen, and W. Yang. Revealing noncovalent interactions. *Journal of the American Chemical Society*, 2010.

[19] T. L. Kay and J. T. Kajiya. Ray tracing complex scenes. In *Proc. of ACM SIGGRAPH*, volume 20, pages 269–278. ACM SIGGRAPH, 1986.

[20] J. Kehrer and H. Hauser. Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):495–513, 2013.

[21] D. J. Lehmann and H. Theisel. Discontinuities in Continuous Scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1291–1300, 2010.

[22] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Computer Graphics*, 21(4):163–169, 1987.

[23] N. Max. Hexahedron projection for curvilinear grids. *Journal of Graphics, GPU, and Game Tools*, 12(2):33–45, 2007.

[24] D. Meagher. Geometric Modeling Using Octree Encoding. *Computer Graphics and Image Processing*, 19(2):129–147, 1982.

[25] T. S. Newman and H. Yi. A Survey of the Marching Cubes Algorithm. *Computers And Graphics*, pages 854–879, 2006.

[26] G. M. Nielson. On Marching Cubes. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):283–297, 2003.

[27] O. Saeki. *Topology of Singular Fibers of Differentiable Maps*. Number 1854 in Lecture Notes in Mathematics. Springer, 2004.

[28] P. Sereda, A. Bartroli, I. Serlie, and F. Gerritsen. Visualization of boundaries in volumetric data sets using LH histograms. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):208–218, March 2006.

[29] H.-W. Shen, C. D. Hansen, Y. Livnat, and C. R. Johnson. Isosurfacing in Span Space with Utmost Efficiency (ISSUE). In *Proceedings of Visualization 1996*, pages 287–294, 1996.

[30] P. Shirley and A. Tuchman. A Polygonal Approximation to Direct Scalar Volume Rendering. *Computer Graphics*, 24(5):63–70, 1990.

[31] M. van Kreveld, R. van Oostrum, C. L. Bajaj, V. Pascucci, and D. R. Schikore. Contour Trees and Small Seed Sets for Isosurface Traversal. In *Proceedings of the 13th ACM Symposium on Computational Geometry*, pages 212–220, 1997.

[32] C. Weigle and D. Banks. Complex-valued contour meshing. In *IEEE Visualization*, 1996.

[33] R. Wenger. *Isosurfaces: Geometry, Topology & Algorithms*. CRC Press, 2013.

[34] J. Wilhelms and A. Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227, 1992.

[35] G. Wyvill, C. McPheeters, and B. Wyvill. Data Structure for Soft Objects. *The Visual Computer*, 2:227–234, 1986.