

Jacobi Set Driven Search for Flexible Fiber Surface Extraction

Mohit Sharma*

Vijay Natarajan†

Indian Institute of Science, Bangalore

ABSTRACT

Isosurfaces are an important tool for analysis and visualization of univariate scalar fields. Earlier works have demonstrated the presence of interesting isosurfaces at isovalues close to critical values. This motivated the development of efficient methods for computing individual components of isosurfaces restricted to a region of interest. Generalization of isosurfaces to fiber surfaces and critical points to Jacobi sets has resulted in new approaches for analyzing bivariate scalar fields. Unlike isosurfaces, there exists no output sensitive method for computing fiber surfaces. Existing methods traverse through all the tetrahedra in the domain. In this paper, we propose the use of the Jacobi set to identify fiber surface components of interest and present an output sensitive approach for its computation. The Jacobi edges are used to initiate the search towards seed tetrahedra that contain the fiber surface, thereby reducing the search space. This approach also leads to effective analysis of the bivariate field by supporting the identification of relevant fiber surfaces near Jacobi edges.

Index Terms: Human-centered computing—Visualization—Visualization techniques; Human-centered computing—Visualization application domains—Scientific visualization;

1 INTRODUCTION

Data from science and engineering disciplines is often represented as a scalar field over a geometric domain. The scalar field maps each point of the domain to a scalar value. A univariate field refers to a single scalar field and the term multivariate refers to the case of multiple scalar fields defined over the domain. Several topological structures have been introduced for analyzing a univariate field and its critical points – contour tree, Reeb graph, Morse-Smale complex, extremum graph [9]. Extensions to multivariate fields are also known – Jacobi set [6], Reeb space [7], and pareto sets [10]. In addition, the notion of fibers and fiber surfaces is introduced as a generalization of isosurfaces to bivariate fields [4]. A fiber is the preimage of a given pair of scalar values. A fiber surface is a collection of fibers, the preimage of a line segment or polygon in the range space of the bivariate field. Visualizing interesting features within multivariate fields in an automated manner and fast computation of these structures has remained a challenge. Extensive work has been done towards the identification of interesting isovalues and for computing the corresponding isosurfaces efficiently. Contour trees and Reeb graphs help locate the seed cells for a particular isosurface, thereby enabling their efficient extraction. The contour tree also serves as a guide for identifying interesting isovalues. However, there is limited work on counterparts for performing similar tasks for bivariate fields, namely for identifying fiber surfaces of interest.

Our work targets two problems: (a) Computing fiber surfaces in an output sensitive manner, and (b) Identifying and computing interesting fiber surfaces in the vicinity of a Jacobi edge. These problems

are inspired by the univariate counterparts, where critical point information and contour trees are used to identify isosurface components of interest and to compute them with correctness guarantees.

Related work. Methods and approaches for computing isosurfaces have been extensively studied during the past three decades [14, 16]. Oostrum et al. [17] introduce a seed set based approach for isosurface extraction, that provides significant advantages over the domain and range search based methods that were proposed earlier. The contour tree is an abstract representation of the connectivity of isosurfaces, a topological structure that capture the connected components of isosurfaces. It also helps locate critical points whose value is close to an input or query isovalue [9]. The located critical points facilitate faster traversal to the isosurface seed cell. Sharma et al. [23] describe an application of augmented contour trees that helps fast location of a seed cell. In this paper, we extend the seed set based approach to bivariate fields towards the computation of a generalization of the isosurface. The Jacobi set [6] is a bivariate counterpart of critical points. Edelsbrunner et al. [6] show the applicability of Jacobi sets to the study of protein interactions and Lagrange points in the solar system. The Jacobi set is typically a large and complex network consisting of multiple edges, and it is difficult to identify important Jacobi edges. Multiple algorithms are available for simplifying the Jacobi set [2, 15].

Carr et al. [4] generalize the notion of isosurfaces to bivariate fields and introduce fiber surfaces. They demonstrate the use of fiber surfaces to chemistry and medical imaging data. Carr et al. [4] use the continuous scatterplots [1], generalization of discrete scatter plots [21], as an intermediate tool to find bivariate range values that may represent interesting fiber surfaces. The fiber is defined as a generalization of an isosurface. A collection of fibers result in a fiber surface. Jankowai et al. [11] introduce feature level-sets, which generalizes isosurfaces and fiber surfaces. Sane et al. [20] extend the idea of univariate confidence isosurfaces to multivariate feature level-sets. Klacansky et al. [13] present an algorithm for fiber surface computation with correctness guarantees and improved speed. Various applications [3, 18] have used the fiber surface to study and extract interesting features in multifields. Tierny et al. [24] use Jacobi fiber surfaces for bivariate Reeb space computation and show the application of Reeb space based domain segmentation to peel continuous scatterplot layers. Sakurai et al. [19] introduce an approach towards flexible fiber surface extraction that does not require computation of the Reeb space. However, the method is limited to the exploration of fiber surfaces within the vicinity of a given fiber surface. In this paper, we introduce a method for flexible computation of individual fiber surface components and hence a flexible exploration of the fiber surface while also not requiring expensive computation of the Reeb space.

Contributions. We introduce a novel output sensitive approach for computation of fiber surfaces for bivariate fields defined on tetrahedral meshes. We present an approach for fast identification of tetrahedra that contain the fiber surface and utilize existing implementation for extracting the fiber surface within each such tetrahedron. The method works by reducing the search space using the Jacobi set of the bivariate field. Key contributions of our work include

1. An output sensitive algorithm for fiber surface computation in tetrahedral meshes using the Jacobi set.

*e-mail: mohitsharma@iisc.ac.in

†e-mail: vijayn@iisc.ac.in

2. An approach for flexible computation of individual fiber surface components.
3. Interactive guided exploration of fiber surfaces within the vicinity of Jacobi edges.

2 BACKGROUND

This section presents a brief introduction to topological analysis of bivariate fields and the mathematical preliminaries necessary for describing our algorithm. For a detailed description of Morse theory, topological descriptors for scalar fields and the relevant definitions, we refer the reader to surveys and books on the topic [9, 26].

Univariate field. A *scalar field*, also called a *univariate field*, maps each point of a spatial domain \mathcal{M} to a single scalar value.

$$f_u : \mathcal{M} \rightarrow \mathbb{R}.$$

Generally, the scalar value represents a physical quantity like temperature, pressure, speed, height, or distance. Non-physical quantities such as probability distributions may also be represented as a scalar field. Fig. 1(a) shows a univariate field where each domain point is mapped to z-coordinate and visualized using a color map.

Isosurface. Given a scalar value a , the *isosurface* \mathcal{I} represents all points in \mathcal{M} that map to a ,

$$\mathcal{I} = f_u^{-1}(a).$$

The scalar a is referred to as an *isovalue*.

Bivariate field. A collection of scalar fields defined over a common domain is called as *multivariate field* or *multifield*. A *bivariate field* is a special instance of a multifield when two scalar fields are defined over \mathcal{M} ,

$$f = \{f_1, f_2\} : \mathcal{M} \rightarrow \mathbb{R}^2.$$

Analyzing a bivariate field helps explore the relationship between the individual scalar fields f_1 and f_2 and to study the features impacted by both fields simultaneously. For example, electron density field and its gradient magnitude may help in the extraction of atoms or bonds in a molecule. Fig. 1(a) and Fig. 1(b) show a synthetic dataset [25]. Each point of the spatial domain is mapped to a bivariate field $\{f_1, f_2\}$ where f_1 is the z-coordinate and f_2 is the distance of each point from $\{0, 0, 0\}$.

Continuous scatterplot. A CSP is a generalization of the discrete scatterplot to spatially continuous multifields [1]. In this paper, we only consider bivariate fields. The CSP maps a point from the 2D range space of the bivariate field to the density of that point. Density is the continuous counterpart of frequency of data points in a discrete scatterplot. A point in the CSP with high density implies a high number of occurrence for the corresponding pair of values ($f_1 = s_1, f_2 = s_2$). Fig. 1(c) shows the CSP of a bivariate field. Visual inspection of the CSP helps identify interesting values in the range space, say high or low density values or a unique pattern.

Fiber. A fiber [4] is the multivariate counterpart of an isosurface. Given a bivariate isovalue (s_1, s_2) , a *fiber* \mathcal{F} is the collection of points in \mathcal{M} that map to (s_1, s_2) under f ,

$$\mathcal{F} = f^{-1}(s_1, s_2)$$

Fig. 1(d) shows the isosurface $f_1^{-1}(s_1)$ in purple and $f_2^{-1}(s_2)$ in yellow. The intersection of these two isosurfaces shown in black, is the fiber \mathcal{F} . \mathcal{F} is the preimage of the black point shown in Fig. 1(c).

Fiber surface. The preimage of a collection of points that lie along a continuous curve in the CSP is called a *fiber surface*. So, the fiber surface is a special collection of fibers. The blue fiber surface in

Fig. 1(e) corresponds to the 2-edge polygon (blue) in Fig. 1(c). The black fiber belongs to this fiber surface.

Control polygon. The set of two blue edges shown in Fig. 1(c) is called a *control polygon*. A *control polygon* is a polygon embedded in the range space of the bivariate field f . It may be open or closed and serves as an interface to specify interesting fiber surface in the spatial domain. The topology of the fiber surface depends on the control polygon. A closed control polygon corresponds to a closed fiber surface unless it intersects the domain boundary.

Jacobi set. The Jacobi set [6] of two Morse functions defined on a d-manifold is the set of critical points of the restrictions of one function to the isosurfaces of the second function. It can also be considered as the set of points where the gradients of both functions are parallel or one of the gradients vanish. Let $\Delta f_1(x)$ and $\Delta f_2(x)$ denote the gradients of scalar fields f_1 and f_2 , then the Jacobi set is defined as

$$\mathcal{J} = \{x \in M \mid \Delta f_1(x) + \lambda \Delta f_2(x) = 0 \text{ or} \\ \lambda \Delta f_1(x) + \Delta f_2(x) = 0\}$$

We follow Klacansky et al. [13] to present an illustration of the Jacobi set using a simple example. Fig. 1(f) shows the contour tree of f_2 restricted to an isosurface of f_1 . If contour trees for all isosurfaces of f_1 are stacked together then the set of critical points of the restrictions of f_2 form individual edges of the Jacobi set. A Jacobi edge can be considered as a bivariate analog of the critical point. In a univariate setting, the topology of isosurfaces change in the vicinity of critical points. Similarly, topology of fibers change in the vicinity of Jacobi edges. A Jacobi edge can be categorized as a saddle or extremum edge. The categorization is based on the connectedness of its lower and upper links. The star $st(s)$ of a simplex s is defined as the set of simplices that have s as a face. Link $Lk(s)$ of a simplex s is the set of faces of $st(s)$ that do not intersect with s . In the univariate setting, a critical point partitions the 1D range space into two halves, smaller scalar values go to the lower half and higher value to the upper. Lower and upper links are subsets of the link that correspond to this partition. In the bivariate scenario, a Jacobi edge also partitions the 2D range space in two halves. Any point from the domain fall on one side of the edge as shown in Fig. 2. Simulation of Simplicity [8] is employed to ensure that no two edges e_1 and e_2 have collinear images and no edge can have end points mapped to the same bivariate value. In order to categorize an edge e as Jacobi edge, a signed distance measure is defined for each point p

$$d_s(p) = \overrightarrow{f(p)} \overrightarrow{f(v_e)} \cdot \overrightarrow{N_{f(e)}}.$$

d_s is the dot product of two vectors – the normal vector $N_{f(e)}$ to the image $f(e)$ of edge e and the vector connecting an end point $f(v_e)$ of $f(e)$ to $f(p)$. The dot product d_s is positive for points lying on one side of $f(e)$ and negative for points lying on the other side. A point p is categorized in lower link $Lk^-(e)$ if $d_s(p) < 0$ else in $Lk^+(e)$. The edge e is classified as an extremum Jacobi edge if either of $Lk^-(e)$ or $Lk^+(e)$ is empty, a saddle Jacobi edge if any of $Lk^-(e)$ or $Lk^+(e)$ consists of more than one connected component, else as a regular edge. In Fig. 2, p_1 and p_2 are identified as $Lk^+(e)$; p_3, p_4, p_5 and p_6 as $Lk^-(e)$. Both the lower and upper link have two disconnected components and hence e is identified as the saddle edge. The topology of fibers in the vicinity of Jacobi edges in the range space changes in the similar manner. Topology stays the same upon crossing a regular edge, refer Fig. 3(a). The fibers originate or vanish around an extremum edge, refer Fig. 3(b). Fibers split or merge in the neighborhood of a saddle edge, refer Fig. 3(c). Fig. 1(g) shows the computed Jacobi edges, the saddle in green and extremum edges in blue, the inset shows a finite number of stacked critical points of f_2 on different level sets of f_1 .

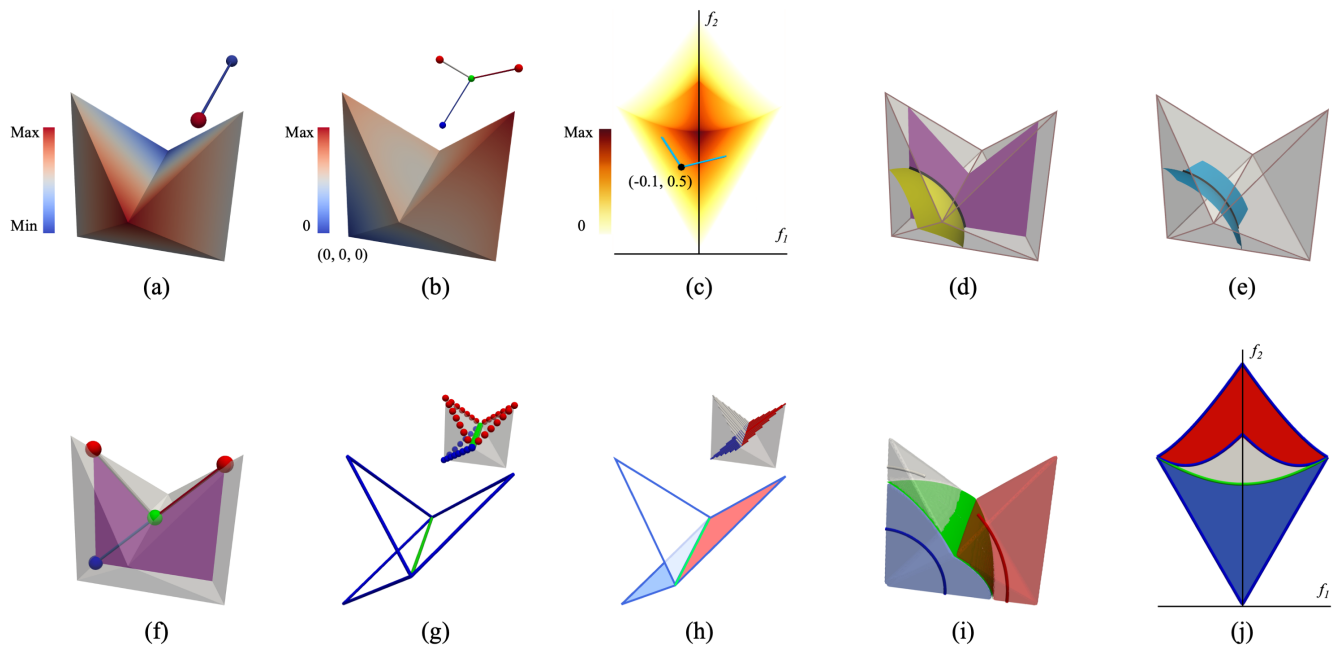



Figure 1: (a,b) A synthetic bivariate field. (a) f_1 is the z-coordinate. (b) f_2 is a distance field, the distance of a point from $(0,0,0)$. (c) Continuous Scatter Plot (CSP) of the bivariate field. A blue open control polygon is selected. One of its vertices $(-0.1, 0.5)$ is shown in black. (d) The purple isosurface of f_1 corresponds to isovalue -0.1 and the yellow isosurface of f_2 corresponds to isovalue 0.5 . The two isosurfaces intersect along the black curve, the fiber corresponding to $(-0.1, 0.5)$. (e) The blue fiber surface corresponding to the control polygon passing through the black fiber. (f) Contour tree of f_2 for level set (purple) $f_1 = -0.1$ (g) Jacobi set of the bivariate field. Extremum edges shown in blue and saddle edge shown in green, inset shows stacked critical points of f_2 for finite level sets of f_1 . (h) Reeb space with different sheets shown in different colors, inset shows the stacked contour trees (i) Reeb space based domain segmentation, each segment represents same fiber topology (J) Segmentation projected onto the 2D range space. All CSPs in this paper use a yellow-red color map () in log scale to visualize density.

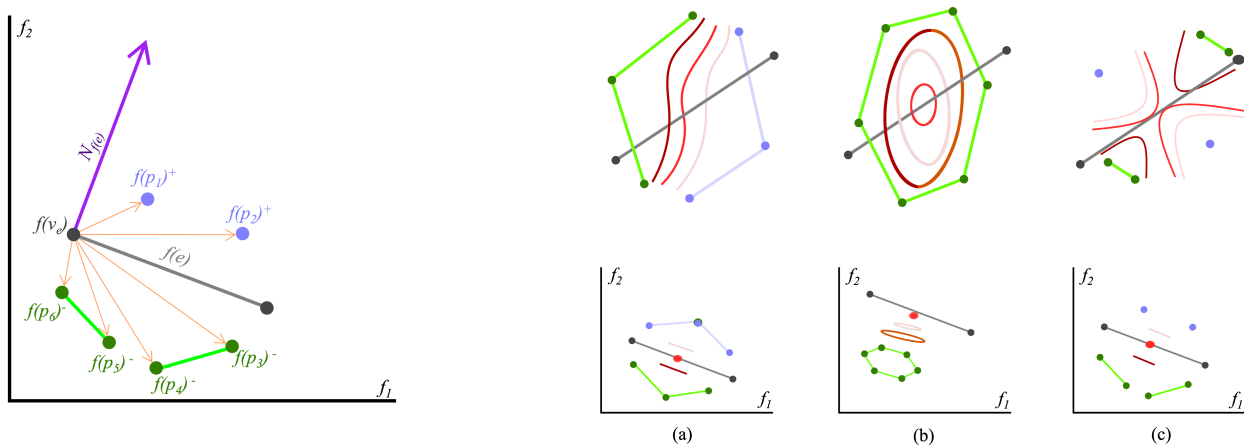


Figure 2: Classifying a Jacobi edge based on $Lk^-(e)$ and $Lk^+(e)$. The green points are identified in $Lk^-(e)$ and blue in $Lk^+(e)$. Both $Lk^+(e)$ and $Lk^-(e)$ have two disconnected components and hence the edge is classified as a saddle Jacobi edge.

Reeb space. Reeb space is the bivariate analog of the Reeb graph. In univariate scenario, if a range of scalar values is swept from minimum to maximum and each level set component is contracted to a point, the resulting structure is called the Reeb graph. For the bivariate counterpart, if each fiber component is contracted to a point, the resulting structure is called the Reeb space. Alternatively, it can be computed by stacking the Reeb graphs of one scalar field restricted to the infinite family of isosurfaces of the second function. Fig. 1(h) shows the Reeb graph computed for a synthetic bivariate field. Inset shows the stacked Reeb graphs of f_2 computed over a finite number of isosurfaces of f_1 . Boundary of each arc in the

Figure 3: (Top) Fiber topology in the neighborhood of an edge (black) in the domain. (Bottom) Corresponding projections in range space. The lower link is represented in green and upper link in blue. (a) Fiber topology does not change upon crossing a regular edge in the range space. (b) Fibers originate or vanish at an extremum Jacobi edge. (c) Fibers split or merge at a saddle Jacobi edge.

Reeb graph are the critical points. The arc forms a surface in the Reeb graph and critical points form the boundary of that surface called Jacobi edges. We refer these surfaces as “sheets” in rest of the discussion. Each sheet of the Reeb graph represents fibers of same topology lying inside corresponding 3D segment in the domain. The synthetic bivariate field has three different segments shown as blue, red, and white in Fig. 1(i) corresponding to three surfaces of Reeb graph bounded by Jacobi edges.

Jacobi fiber surfaces. Image $f(e)$ of a Jacobi edge e is an edge in the range space. If $f(e)$ is used as a control polygon, the resulting fiber surface passes through e and is called a Jacobi fiber surface. Saddle fiber surface correspond to saddle Jacobi edge and extremum fiber surface correspond to extremum Jacobi edge. Topology of fibers change only in the vicinity of a Jacobi fiber surface. A fiber can have multiple components similar to an isosurface. A fiber corresponding to a point lying in the vicinity of $f(s)$, where s is the saddle Jacobi edge, may have more than one component. Jacobi fiber surfaces partition the domain into 3D segments. Each segment is a collection of fibers with the same topology. In Fig. 1(i), the blue fiber in the blue segment corresponds to bivariate value $\{0, 0.4\}$. As f_2 increases for a fixed value of $f_1 = 0$, the blue fiber remains a single component until it crosses the green fiber surface when it splits into two components within the red and white segments. The map of each 3D segment in the range space is bounded by the corresponding Jacobi edges. In Fig. 1(j), the green saddle Jacobi edge separates the blue segment from the other segments. White and red segments are also bounded by corresponding Jacobi edges but overlap in the range space due to overlapping bivariate values.

3 JACOBI SET BASED FIBER SURFACE EXTRACTION

We now describe an algorithm for computing the fiber surface using an efficient directed search to locate seed tetrahedra followed by a surface traversal step. The algorithm is designed based on the intuition gained from the univariate counterpart. In the case of a univariate field, the augmented contour tree is queried to gain direct access to seed tetrahedra that correspond to an isovalue [5]. A direct extension to bivariate fields would use the Reeb space to identify seeds. Given the computational challenges in constructing and storing the Reeb space, we propose the use of the simpler Jacobi set. The Jacobi set (\mathcal{J}) of a bivariate field f defined over a tetrahedral mesh (\mathcal{M}) and a control polygon (\mathcal{C}) defined on the range space of f are given as input. \mathcal{J} remains the same for a particular bivariate field, hence it can be precomputed and reused for different control polygons. The Jacobi set is computed using TTK [25]. Our algorithm locates and extracts all tetrahedra that contain the fiber surface corresponding to an edge (u, v) of \mathcal{C} . The fiber surface computation for each control polygon edge can be executed independently. We now describe a four-step algorithm that computes the fiber surface for a control polygon edge (u, v) . This algorithm can be iteratively executed for each edge of \mathcal{C} . Algorithm 1 shows pseudo code for the three steps that locate all tetrahedra that contain the fiber surface.

(A) Jacobi intersections. In the univariate setting, the input isovalue is used to traverse the contour tree and identify the arcs that span the isovalue. A directed search to find the seed cell is initiated from one end point of that arc, both of which are critical points. In the bivariate scenario, the aim is to locate the Reeb space sheet(s) containing (u, v) and extract the set (J_s) of Jacobi edges bounding the sheet(s). Any edge from J_s may be used to initiate the directed search. The Reeb space is not available to our algorithm. Instead, we compute the Jacobi edges that intersect the line L containing the edge (u, v) , see Fig. 4. The set of intersected Jacobi edges J_{int} contains J_s together with additional edges. In Fig. 4, edge 3 or edge 4 would yield the required seed cell since (u, v) lies inside the blue segment, but edges 1 and 2 are also identified as intersected Jacobi edges. Line 3 in Algorithm 1 implements this step. The running time of this Jacobi edge intersection computation step is $O(|\mathcal{J}|)$.

(B) Directed search. Each intersected Jacobi edge may or may not yield a seed tetrahedron. In order to find out the seed tetrahedron, a directed breadth first search (BFS) is initiated from the cell (tetrahedron) containing the Jacobi edge. Lines 6 to 14 in Algorithm 1 show the pseudo code for this directed search. In each iteration of the directed BFS, a neighbor tetrahedron whose edge intersects L and has the closest intersection point to either endpoint of (u, v) , is added to the BFS queue (Line 14 in Algorithm 1). The aim is to

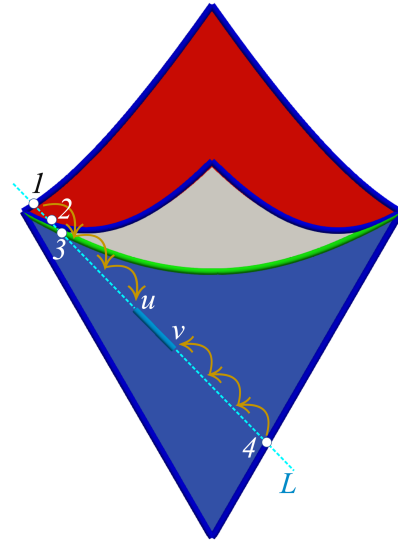


Figure 4: Control edge (u, v) overlapping with blue Reeb space sheet and enclosed by intersections with Jacobi edges. Line L (dashed) extended through (u, v) (solid) intersects with four Jacobi edges. Directed search is initiated from all, to find the seed cell overlapping with either u or v .

select a neighbor tetrahedron that helps to move towards the seed cell intersecting with (u, v) . Each edge e of a tetrahedron maps to an image $f(e)$ in the range space. Intersection of L and $f(e)$ for all edges e of every neighbor tetrahedron is computed. If there is no intersection then clearly the tetrahedron will not lead to (u, v) and can be discarded. Else, the euclidean distance inside range space between the intersection point to u and v is computed. The neighbor tetrahedron with the edge having minimum distance is chosen. The directed search terminates either upon reaching a cell intersected by (u, v) in the range space (Line 9) or when there exist no new cell to explore *i.e.*, upon reaching a dead end or boundary point is reached (Line 15). The BFS inherently provides an optimization, namely that no directed search is initiated from a tetrahedron containing an intersected Jacobi edge if it is visited during a directed search initiated elsewhere. The time taken for the directed search depends on the number of visited tetrahedra $|D_t|$. Clearly, D_t cannot exceed the total number of tetrahedra T . Hence, the running time is $O(|D_t|) = O(|T|)$.

(C) Restricted BFS Next, we extract the tetrahedra containing the fiber surface by initiating a restricted BFS from each seed cell. The BFS explores only those cells that overlap with (u, v) . All cells yielded in this step contain some part of the fiber surface. BFS initiated from each of the seed cells will yield at least one fiber surface component but multiple seeds may belong to the same component. Hence, seeds that are visited during a restricted BFS initiated from another seed cell are discarded. If the required fiber surface passes through $|Z|$ tetrahedra then this step traverses exactly $O(|Z|)$ tetrahedra.

(D) Fiber surface extraction. The set of tetrahedra (Z) extracted in the previous step contain the required fiber surface. We extract these tetrahedra from the domain and supply them as input to TTK, to reuse their implementation for computing the fiber surface within each tetrahedron [13]. The running time for this step is linear in the number of input tetrahedra, $O(|Z|)$.

Correctness. The restricted BFS traverses through all cells reachable from a seed cell while ensuring that the range of scalar values within the traversed cell overlap with the control polygon edge (u, v)

Algorithm 1 Extract all tetrahedra containing the fiber surface

```
1: procedure EXTRACTFIBERSURFACETETS( $\mathcal{M}, f, \mathcal{J}, (u, v)$ ) ▷ Returns Z: Tetrahedra containing the fiber surface of  $(u, v)$ 
2:    $L \leftarrow$  line passing through  $(u, v)$ 
3:    $J_{int} \leftarrow$  Jacobi edges intersected by  $L$ 
4:    $Z \leftarrow$  NULL
5:   for all  $j_i$  in  $J_{int}$  do ▷ Initiate directed search from each intersected Jacobi edge
6:      $start \leftarrow$  tetrahedron containing  $j_i$ 
7:      $seedT \leftarrow$  NULL
8:     while true do
9:       if  $start$  contains either  $u$  or  $v$  then
10:         $seedT \leftarrow start$ 
11:        break
12:       end if
13:        $prevStart \leftarrow start$ 
14:        $start \leftarrow$  traverseToNeighborClosestToUV( $start$ ) ▷ Directed traversal
15:       if  $start == prevStart$  then ▷ Dead end
16:         break
17:       end if
18:     end while
19:      $Z \leftarrow Z +$  restrictedBFS( $seedT$ )
20:   end for
21:   return  $Z$ 
22: end procedure
```

in the range space. We prove the correctness of the algorithm by showing that it locates at least one seed cell for each connected fiber surface component. If a control polygon edge (u, v) maps to n connected fiber surface components then it will overlap with exactly n Reeb space sheets. In the projected range space, the overlapping segments of (u, v) will be enclosed by bounding Jacobi edges of the corresponding sheets. The infinite line L extended through (u, v) will intersect with at least one of those enclosing Jacobi edges. As discussed in Sect. 2, each surface in the Reeb space represents fibers having same topology without any split or merge. Hence, a directed search initiated from a valid Jacobi edge of that particular sheet will locate the seed cell.

Runtime analysis. Adding the time taken for the four steps, the total running time of the algorithm is $O(|\mathcal{J}| + |D_t| + |Z|) = O(T)$. In practice, the run time crucially depends on the number of Jacobi edges and the number of intersected Jacobi edges.

4 RESULTS

We now describe results of experiments conducted on four different datasets: Ethanediol, Thiophene-Quinoxaline, Tooth, and Combustion. First, we study the correctness of the results via comparisons against results obtained using a previous method implemented in TTK [25]. We also compare the runtime performance of our Jacobi set driven search against a previous method that employs domain search [13]. Next, we highlight specific applications that are supported by our output sensitive approach to fiber surface extraction.

4.1 Quality

Fig. 5 shows a visual comparison of fiber surfaces generated using our Jacobi set driven search algorithm and the existing algorithm [13], implemented in TTK. We have selected a control polygon containing a single edge for simplicity. The algorithm can be executed independently for each edge of a generic control polygon, as mentioned in Sect. 3. control polygons and their corresponding fiber surfaces are shown using a common color. Visually, the fiber surfaces computed by our algorithm and TTK look the same. We also compare the expected number of tetrahedra intersected by the fiber surface against the number of tetrahedra reported by our algorithm. The expected number of tetrahedra intersecting with fiber surface is computed by traversing the entire domain. Table 1 shows

Dataset	Control polygon	#Tets containing FS	#Tets using Jacobi set driven search
Ethanediol ¹	Blue	12804	12804
	Green	3524	3524
	Red	4718	4718
	Black	7612	7612
Ethanediol ²	Pink	138239	138239
Thiophene-Quinoxaline	Green	4032	4032
	Red	946	946
Tooth	Blue	223145	223145
	Yellow	47796	47796
	Red	27645	27645
	Grey	87185	87185
Combustion	Blue	157287	157287

Table 1: Validating the algorithm by comparing the number of tetrahedra containing the fiber surface (FS) against an exhaustive search.

the corresponding results. We observe an exact match, signifying that no tetrahedron containing the fiber surface is missed by our algorithm.

4.2 Runtime performance

The runtime performance of our algorithm depends on the number of topological features in the input dataset, specifically the number of Jacobi edges. Table 2 shows the computation time taken by individual traversal steps of the algorithm for different datasets and corresponding control polygons. #Cells represents the total number of tetrahedra in the domain, #Jacobi edges represents number of Jacobi edges, #Jacobi intersections represents number of Jacobi edges intersected by the line L corresponding to a control polygon, #Tets containing FS represents number of tetrahedra that contain (intersect) the output fiber surface. Time taken by individual steps of the algorithm is specified in columns A, B, and C.

Ethanediol¹ and Ethanediol² are the same dataset represented by the same bivariate field (f_1 : electron density, f_2 : reduced gradient [12]), but associated with two different query control polygons. The time taken to compute the Jacobi intersections is similar for all control polygons of a particular dataset because it primarily

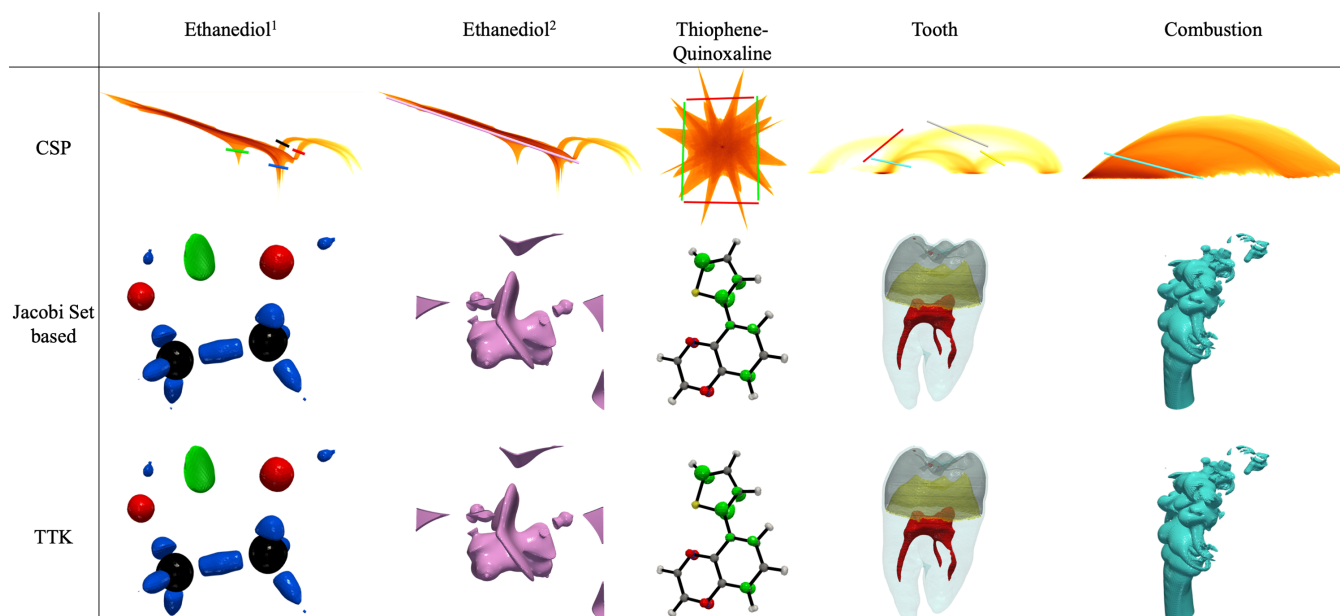


Figure 5: Visual comparison. Fiber surfaces computed by the proposed Jacobi set based search algorithm match with those computed using the fast and exact algorithm implemented in TTK [13].

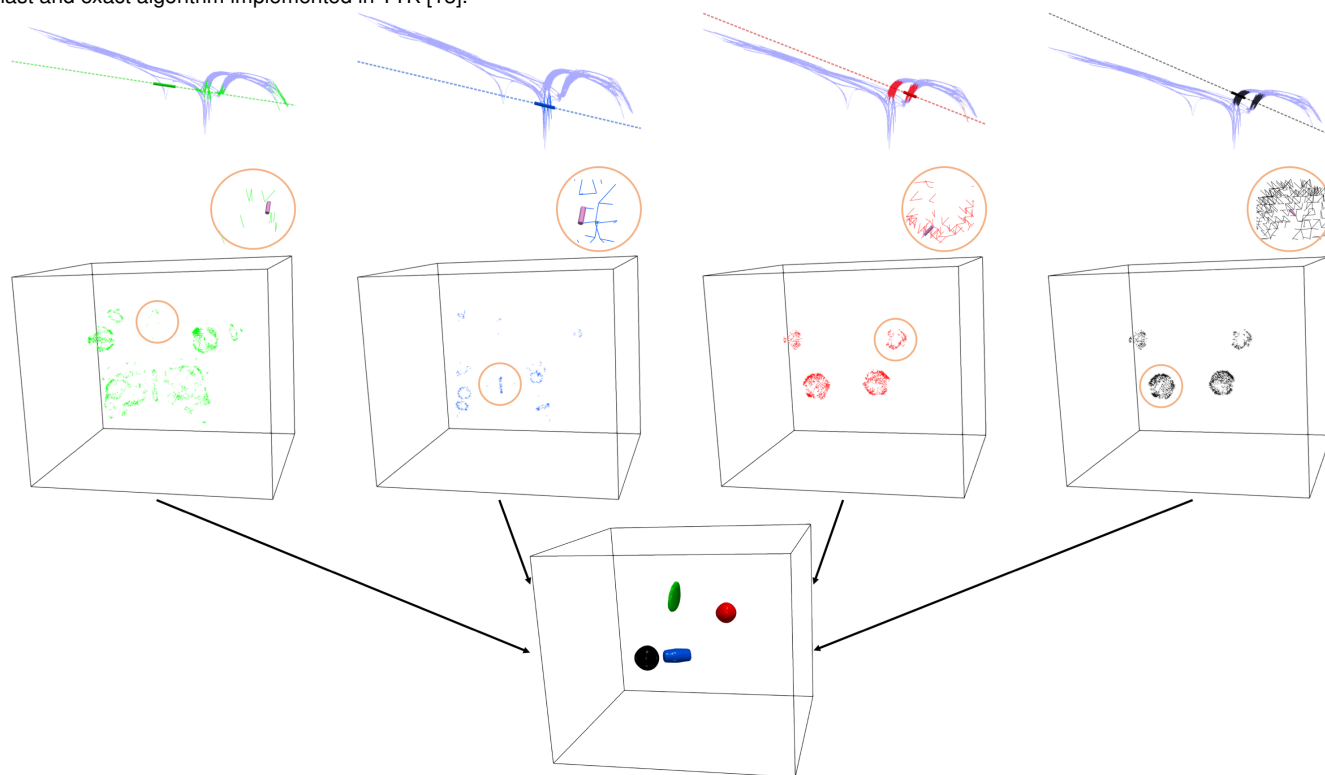


Figure 6: Flexible fiber surfaces extracted from the Ethanediol dataset. (Top) Jacobi edges in the range space. Intersected Jacobi edges are highlighted in corresponding color. (Middle) Intersected Jacobi edges are shown in domain space to facilitate interactive selection of an edge, with the objective of exploring fiber surface in its neighborhood. Inset shows the selected edge in pink. (Bottom) Fiber surface components extracted for the selected Jacobi edges.

depends on the number of Jacobi edges. We observe that this step is the computational bottleneck. The high total computation times for Tooth and Combustion is due to the large number of Jacobi edges in these datasets. The time taken for directed search depends on the number of Jacobi edges that intersect the line L because it

corresponds to the number of directed searches that may be initiated. We observe that the location of the intersected Jacobi edges plays an equal if not more important role in determining the time for directed search. The blue control polygon in the Tooth dataset intersects with a larger number of Jacobi edges than the yellow polygon but

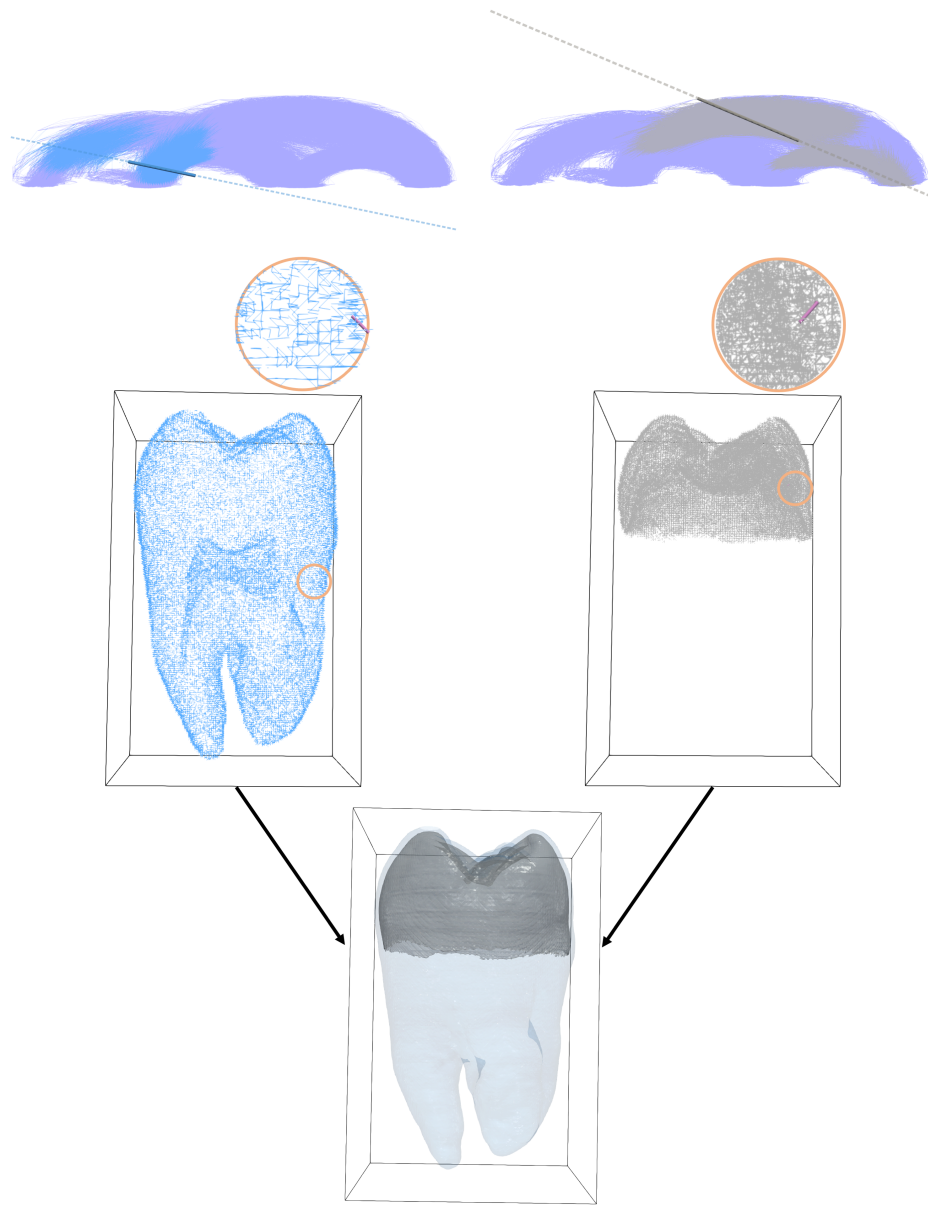


Figure 7: Flexible fiber surfaces extracted from the Tooth dataset. (Top) Jacobi edges in the range space. Intersected Jacobi edges are highlighted in corresponding color. (Middle) Intersected Jacobi edges shown in domain space to facilitate interactive selection of an edge, with the objective of exploring fiber surface in its neighborhood. Inset shows the selected edge in pink. (Bottom) Fiber surface components extracted for the selected Jacobi edges.

requires lesser amount of time for directed search because several intersected Jacobi edges are visited during directed search initiated from another edge. As expected, we observe that the time taken for the restricted BFS step depends on the number of tetrahedra that contain the fiber surface.

We collect these tetrahedra and pass them to TTK in order to reuse its implementation of the numerical computation of fiber surface within a tetrahedron. The time taken for extracting this collection of tetrahedra, storing them in a data structure, and transferring the collection to TTK is mentioned under the data transfer column in Table 2. This data transfer step can be avoided if the proposed search algorithm is integrated in TTK, wherein the fiber surface can be computed as soon as a tetrahedron is marked during the restricted BFS step. Table 3 shows the total time taken to compute the fiber surface *i.e.*, traversal time from Table 2 and fiber surface extraction

time for the identified tetrahedra. TTK implementation traverses through all tetrahedra passed to it, computes the fiber surface section enclosed in a particular tetrahedron if the fiber surface passes through it. Since all tetrahedra identified by our algorithm contain the fiber surface, TTK processes all tetrahedra. The running time increases with the number of tetrahedra passed to TTK, refer Table 3. The running time of our algorithm is comparable with the exhaustive search in TTK, particularly when the number of Jacobi edges in the dataset is small (Ethanediol¹). The octree based domain search implemented in TTK is faster in all cases. For Ethanediol², the number of tetrahedra that contain the fiber surface is large and seem scattered throughout the octree, causing the octree version to perform worse than the exhaustive search.

The restricted BFS step of our algorithm approach relies on triangulation data structure implemented in TTK. Accessing cell neigh-

Dataset	#Cells	#Jacobi edges	Control polygon	#Jacobi intersections	#Tets containing FS	Computation times (msec)				
						Jacobi intersections (A)	Directed Search (B)	Restricted BFS (C)	Traversal (T = A + B + C)	Data transfer
Ethanediol ¹	8718150	95935	Blue	394	12804	31	0*	57	88	454
			Green	2066	3524	31	23	15	69	446
			Red	1318	4718	31	11	21	63	448
			Black	1556	7612	30	5	33	68	446
Ethanediol ²	8718150	95935	Pink	1048	138239	31	1	629	661	565
Thiophene-Quinoxaline	2479680	20737	Green	38	4032	17	0*	18	35	258
			Red	26	946	14	0*	4	18	252
Tooth	7588800	2732204	Blue	44821	223145	897	95	1021	2013	586
			Yellow	39185	47796	874	273	217	1364	428
			Red	7236	27645	879	11	123	1013	406
			Grey	57149	87185	885	351	397	1633	463
Combustion	18675345	2449505	Blue	19668	157287	819	30	757	1606	1093

Table 2: Time taken by Jacobi set driven search algorithm to locate all tetrahedra containing the required fiber surface (FS) and time to transfer to TTK for fiber surface computation. Time taken to compute the Jacobi intersections is directly proportional to number of Jacobi edges. Time for executing the restricted BFS increases with number of tetrahedra that contain the fiber surface. Ethanediol¹ and Ethanediol² are identical datasets; the control polygon chosen in Ethanediol² results in a larger sized fiber surface. 0* represents values smaller than 1 msec.

Dataset	#Cells	Control polygon	#Tets containing FS	Using Jacobi set (msec)			TTK (msec)	
				Traversal (T)	FS Extraction (D)	Total (T+D)	Without octree	With octree
Ethanediol ¹	8718150	Blue	12804	88	21	109	472	28
		Green	3524	69	5	74	472	10
		Red	4718	63	8	71	461	11
		Black	7612	68	13	81	471	18
Ethanediol ²	8718150	Pink	138239	661	237	898	675	726
Thiophene-Quinoxaline	2479680	Green	4032	35	6	41	259	8
		Red	946	18	1	19	257	2
Tooth	7588800	Blue	223145	2013	386	2399	767	519
		Yellow	47796	1364	83	1447	485	113
		Red	27645	1013	47	1060	424	81
		Grey	87185	1633	151	1784	579	195
Combustion	18675345	Blue	157287	1606	271	1877	1211	339

Table 3: Total time taken by our algorithm for fiber surface (FS) computation. The algorithm performs better in datasets with fewer number of Jacobi edges.

bors of each traversed cell increases the computation time by a large amount. The computation times are still comparable. For Thiophene-Quinoxaline, the two scalar fields are x :hole.nto and y :particle.nto. Hole.nto represents the charge lost and particle.nto represents the charge gained during electronic transition of molecule. The green control polygons highlight the fiber surfaces corresponding to region that donated charge and red highlights the acceptor region in molecule [22]. Since the number of Jacobi edges are two orders of magnitude smaller than the number of cells, Jacobi set driven search performs fairly well. For Tooth (f_1 : scalar field, f_2 : gradient magnitude), the number of Jacobi edges and number of cells in the dataset are of same order. Computing Jacobi intersections takes more time than traversing all cells in an exhaustive search. The difference is lower for Combustion (f_1 : scalar field, f_2 : gradient magnitude) because the number of Jacobi edges are approximately one-tenth the total number of cells. The results are computed on a machine with 2.10GHz Intel(R) Xeon(R) Gold 6130, 32 core processor and 345 GB RAM. All runtimes are reported for a serial implementation. The runtimes are average of five runs out of seven. The runs with maximum and minimum runtime are discarded.

4.3 Applications

The algorithm supports exploration of fiber surfaces in the vicinity of Jacobi edges and flexible computation of individual components of a

fiber surface. The top row in Fig. 6 shows the Jacobi edges projected onto the range space with four different control polygons. The Jacobi edges intersected by the line extended through the control polygon edge are highlighted with the same color as the control polygon edge. Visual inspection enables the user to identify edges that intersect the control polygon. Visualizing the intersecting Jacobi edges in the domain (middle row) provides good insight into the spatial distribution of features in the data. This insight helps the user interactively select a Jacobi edge and explore its neighborhood using fiber surfaces. The insets show the selected control polygon edge in pink. The bottom row shows fiber surface components that are extracted by initiating a directed search from the selected Jacobi edges and launching a restricted BFS from the resulting seed cell. We observe that the selected components correspond to individual atoms and bond structure in Ethanediol. In Fig. 7, two control polygons are identified in a similar manner for the Tooth dataset resulting in two fiber surface components. The set of all intersecting Jacobi edges form a network in the vicinity of the output fiber surfaces. These networks or clusters act as a guide to select an intersecting Jacobi edge. Selecting one edge from each cluster is sufficient to explore the fiber surface component covered by other Jacobi edges within the connected cluster. The user may proceed with the exploration by visually analyzing these networks and selecting few Jacobi edges.

Table 4 presents a comparison between the time taken for flexible

Dataset	Control polygon	#Tets containing FS component	#Tets containing FS	FS component computation (msec)			FS computation (msec)		
				Jacobi intersections (A)	Directed Search (B)	Restricted BFS (C)	Jacobi intersections (A)	Directed Search (B)	Restricted BFS (C)
Ethanediol ¹	Blue	2660	12804	30	0*	12	31	0*	57
	Green	3524	3524	31	0*	16	31	23	15
	Red	2372	4718	31	0*	10	31	11	21
	Black	3796	7612	31	0*	17	30	5	33
Tooth	Blue	223076	223145	899	0*	1028	897	95	1021
	Grey	87185	87185	873	0*	399	885	351	397

Table 4: Time taken by our algorithm to extract a component of the fiber surface. Time to compute the Jacobi intersections is independent of the component. Time required for directed search to locate the seed tetrahedron is negligible for component computation due to interactive selection of Jacobi edge. Time required to execute restricted BFS depends on the number of tetrahedra containing the fiber surface component, and is hence comparatively small relative to the total time. 0* represents values smaller than 1 msec.

Dataset	Control polygon	#Tets containing FS component	#Tets containing FS	FS component computation (msec)			FS computation (msec)		
				Traversal (T = A+B+C)	FS Extraction (D)	Total (T+D)	Traversal (T = A+B+C)	FS Extraction (D)	Total (T+D)
Ethanediol ¹	Blue	2660	12804	42	4	46	88	21	109
	Green	3524	3524	47	5	52	69	5	74
	Red	2372	4718	41	4	45	63	8	71
	Black	3796	7612	48	6	54	68	13	81
Tooth	Blue	223076	223145	1927	389	2316	2013	386	2399
	Grey	87185	87185	1272	151	1423	1633	151	1784

Table 5: Overall time to compute a particular fiber surface component is always lower than time taken to compute complete fiber surface due to interactive selection of Jacobi edges and due to the smaller number of tetrahedra that contain the selected fiber surface component.

computation of a fiber surface component that corresponds to an interactively selected Jacobi edge against time taken to compute the complete fiber surface corresponding to a control polygon edge. The time taken for directed search is almost negligible in the former case because only one directed search is initiated for the interactively selected Jacobi edge. Time taken for restricted BFS is less in cases where the component is significantly smaller than the complete fiber surface. In Table 5, fiber surface extraction time (D) also depends on the number of tetrahedra that contain the fiber surface component or complete fiber surface. Total time (T+D) for component calculation is less than time required for complete fiber surface computation thanks to a negligibly small directed search time and reduced time for steps C and D.

5 CONCLUSIONS

In this paper, we presented an output sensitive approach to compute the fiber surface for a bivariate field defined over a tetrahedral mesh. The method does not require explicit computation of the Reeb space and runtime depends on the number of Jacobi edges. The runtime performance is good, with runtimes less than a few seconds, even though it is slower than TTK. A key benefit of the approach is that it enables exploration of fiber surfaces within the vicinity of Jacobi edges. It supports flexible computation of individual components of the fiber surface. In future work, we plan to improve the runtime of the algorithm via Jacobi set simplification and to devise methods for identifying interesting Jacobi edges, thereby avoiding the need for user interaction with the CSP or the projected range space. We observe that the Jacobi set intersection step dominates the runtime. We plan to explore space partitioning techniques like quad tree and employ parallelization strategies to reduce the time taken by this step. Further, we also plan to analyze the performance with respect to memory requirement and preprocessing time.

ACKNOWLEDGMENTS

This work is partially supported by MoE Govt. of India, a Swarnajayanti Fellowship from SERB India (DST/SJF/ETA-02/2015-16),

an Indo-Swedish joint network project (DST/INT/SWD/VR/P-02/2019), and a Mindtree Chair research grant.

REFERENCES

- [1] S. Bachthaler and D. Weiskopf. Continuous scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1428–1435, 2008. doi: 10.1109/TVCG.2008.119
- [2] H. Bhatia, B. Wang, G. Norgard, V. Pascucci, and P.-T. Bremer. Local, smooth, and consistent jacobi set simplification. *Computational Geometry*, 48(4):311–332, 2015. doi: 10.1016/j.comgeo.2014.10.009
- [3] C. Blecha, F. Raith, G. Scheuermann, T. Nagel, O. Kolditz, and J. Maßmann. 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)*, pp. 189–201, 2019. doi: 10.1109/PacificVis.2019.00030
- [4] H. Carr, Z. Geng, J. Tierny, A. Chattopadhyay, and A. Knoll. Fiber surfaces: Generalizing isosurfaces to bivariate data. *Computer Graphics Forum*, 34(3):241–250, 2015.
- [5] H. Carr, J. Snoeyink, and M. Van De Panne. Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree. *Computational Geometry*, 43(1):42–58, 2010.
- [6] H. Edelsbrunner and J. Harer. Jacobi sets of multiple morse functions. *Foundations of Computational Mathematics - FoCM*, 01 2004.
- [7] H. Edelsbrunner, J. Harer, and A. K. Patel. Reeb spaces of piecewise linear mappings. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, pp. 242–250, 2008.
- [8] H. Edelsbrunner and E. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms, 1994. doi: 10.48550/ARXIV.MATH/9410209
- [9] C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. De Floriani, G. Scheuermann, H. Hagen, and C. Garth. A survey of topology-based methods in visualization. *Computer Graphics Forum*, 35(3):643–667, 2016.
- [10] L. Huettenberger, C. Heine, and C. Garth. Decomposition and simplification of multivariate data using pareto sets. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2684–2693, 2014.

- [11] J. Jankowai and I. Hotz. Feature Level-Sets: Generalizing Iso-surfaces to Multi-Variate Data. *IEEE Transactions on Visualization and Computer Graphics*, 26(2):1308–1319, 2020. doi: 10.1109/TVCG.2018.2867488
- [12] E. R. Johnson, S. Keinan, P. Mori-Sánchez, J. Contreras-García, A. J. Cohen, and W. Yang. Revealing noncovalent interactions. *Journal of the American Chemical Society*, 132(18):6498–6506, 2010. PMID: 20394428. doi: 10.1021/ja100936w
- [13] P. Klacansky, J. Tierny, H. Carr, and Z. Geng. Fast and exact fiber surfaces for tetrahedral meshes. *IEEE Transactions on Visualization and Computer Graphics*, 23(7):1782–1795, 2016.
- [14] W. E. Lorensen. History of the marching cubes algorithm. *IEEE computer graphics and applications*, 40(2):8–15, 2020.
- [15] S. N and V. Natarajan. Simplification of jacobi sets. In V. Pascucci, X. Tricoche, H. Hagen, and J. Tierny, eds., *Topological Methods in Data Analysis and Visualization: Theory, Algorithms, and Applications*, pp. 91–102. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi: 10.1007/978-3-642-15014-2_8
- [16] T. S. Newman and H. Yi. A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5):854–879, 2006.
- [17] R. Oostrum, V. Kreveld, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. *13th ACM Symposium on Computational Geometry*, 04 1999. doi: 10.1145/262839.269238
- [18] F. Raith, C. Blecha, T. Nagel, F. Parisio, O. Kolditz, F. Günther, M. Stommel, and G. Scheuermann. Tensor field visualization using fiber surfaces of invariant space. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1122–1131, 2019. doi: 10.1109/TVCG.2018.2864846
- [19] D. Sakurai, K. Ono, H. Carr, J. Nonaka, and T. Kawanabe. *Flexible Fiber Surfaces: A Reeb-Free Approach*, pp. 187–201. Mathematics and Visualization. Springer Science and Business Media Deutschland GmbH, Germany, 2020. doi: 10.1007/978-3-030-43036-8_12
- [20] S. Sane, T. M. Athawale, and C. R. Johnson. Visualization of Uncertain Multivariate Data via Feature Confidence Level-Sets. In M. Agus, C. Garth, and A. Kerren, eds., *EuroVis 2021 - Short Papers*. The Eurographics Association, 2021. doi: 10.2312/evs.20211053
- [21] A. Sarikaya and M. Gleicher. Scatterplots: Tasks, data, and designs. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):402–412, 2017.
- [22] M. Sharma, T. B. Masood, S. S. Thygesen, M. Linares, I. Hotz, and V. Natarajan. Segmentation driven peeling for visual analysis of electronic transitions. In *Proc. IEEE Visualization Conference, IEEE VIS 2021 - Short Papers*, pp. 96–100. IEEE, 2021.
- [23] M. Sharma and V. Natarajan. On-demand augmentation of contour trees. In *Proceedings of the 11th Indian Conference on Computer Vision, Graphics and Image Processing, ICVGIP 2018*. Association for Computing Machinery, New York, NY, USA, 2018. doi: 10.1145/3293353.3293384
- [24] J. Tierny and H. Carr. Jacobi fiber surfaces for bivariate reeb space computation. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):960–969, 2016.
- [25] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. The topology toolkit. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):832–842, 2018. doi: 10.1109/TVCG.2017.2743938
- [26] L. Yan, T. B. Masood, R. Sridharamurthy, F. Rasheed, V. Natarajan, I. Hotz, and B. Wang. Scalar field comparison with topological descriptors: Properties and applications for scientific visualization. *Computer Graphics Forum*, 40(3):599–633, 2021.