Direct Multifield Volume Ray Casting of Fiber Surfaces

Kui Wu, Aaron Knoll, *Member, IEEE*, Benjamin J Isaac, Hamish Carr, *Member, IEEE*, and Valerio Pascucci, *Member, IEEE*

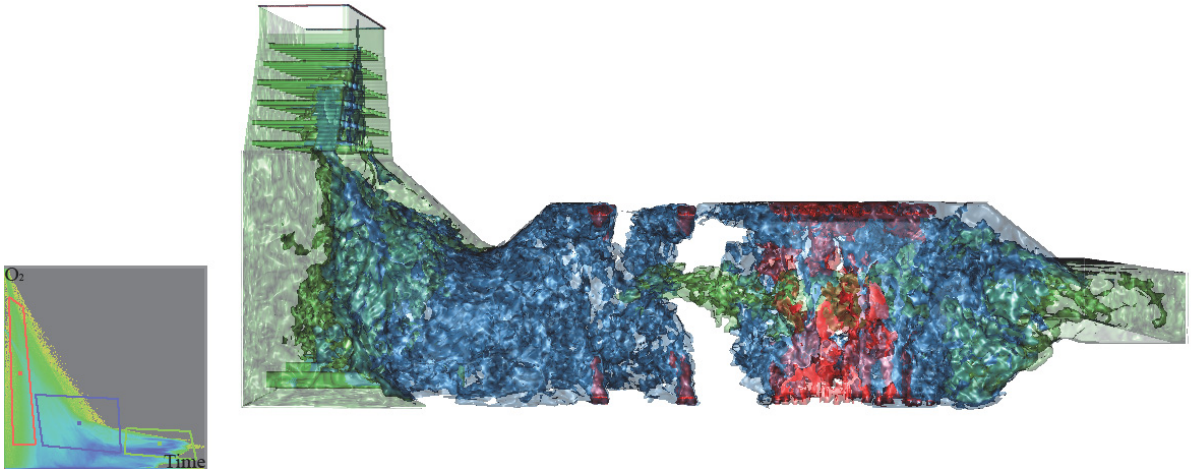

Fig. 1. Left: 2D (joint) histogram and three fiber surface control polygons (FSCPs), specified by red, blue and green annotations. Right: Corresponding fiber surfaces. Let us compare residence time and oxygen across both data range and spatial domain, in a simulation of coal combustion in GE-Alstom's 15 MW_{th} Boiler Simulation Facility (BSF). These surfaces let us show low and high regions of oxygen as they occur over the entire course of the simulation, classified by annotating the 2D scatterplot (joint histogram) with FSCPs. Direct ray casting allows users to explore and manipulate fiber surfaces interactively on larger datasets; in this case at 16 fps at 1024×1024 on an NVIDIA Geforce GT 650M mobile GPU.

Abstract— Multifield data are common in visualization. However, reducing these data to comprehensible geometry is a challenging problem. Fiber surfaces, an analogy of isosurfaces to bivariate volume data, are a promising new mechanism for understanding multifield volumes. In this work, we explore direct ray casting of fiber surfaces from volume data without any explicit geometry extraction. We sample directly along rays in domain space, and perform geometric tests in range space where fibers are defined, using a signed distance field derived from the control polygons. Our method requires little preprocess, and enables real-time exploration of data, dynamic modification and pixel-exact rendering of fiber surfaces, and support for higher-order interpolation in domain space. We demonstrate this approach on several bivariate datasets, including analysis of multi-field combustion data.

Index Terms—Volume Rendering, Isosurface, Multidimensional Data

1 Introduction

Multifield volume data are ubiquitous in scientific computing. Simulations frequently compute several variables, for the purposes of driving the computation itself or understanding underlying physical phenomena. However, most visualizations of 3D volume data consider only a single field in a given image, using either isosurfaces or direct volume rendering. This is due in equal parts to audiences' familiarity with single-field metaphors, and the relative lack of concise techniques for defining and describing multifield data.

Fiber surfaces are multifield equivalents of isosurfaces for univariate 3D volume data. Just as isovalues define contours in single-field data, *fibers* [34] define contours over tuples in multifield data. For bivariate (two-field) volume data, fibers are defined as points in two-dimensional range space. A curve composed of fibers in the range defines a fiber surface in the domain. These features classify spatial volume data as a linear combination of two fields, and provide a powerful tool for defining contours in terms of multiple attributes. As shown in Fig. 1, fiber surfaces allow us to restrict our classification of contours to subregions of bivariate range space. This not only reduces clutter, but allows us to identify features that isosurfaces of these respective fields could not. Fiber surfaces are a recent contribution to the visualization literature [5], and were implemented as a straightforward extension to Marching Cubes [28]. Though effective, surface extraction presents two main limitations. First, the resulting mesh is a piecewise-linear approximations of higher-order analytical implicit surfaces. Second, the extraction process is costly and potentially non-interactive for larger volume data. While methods exist for accelerating marching cubes, for sufficiently large and complex data, direct rendering methods are needed to enable interactive exploration.

Direct isosurface ray casting is a well-known alternative to mesh extraction. Surface ray casting is attractive for its sublinear time complexity – with the appropriate acceleration structure, small and large volume data render at similar speed, constrained only by memory. Moreover, ray casting of implicit surfaces can employ a wide range of

- *Kui Wu is with University of Utah. E-mail: kwu@cs.utah.edu.*
- *Aaron Knoll is with SCI Institute, University of Utah, and Argonne National Laboratory. E-mail: knolla@sci.utah.edu*
- *Benjamin J Isaac is with ICSE, University of Utah. E-mail: benjamin.j.isaac@utah.edu.*
- *Hamish Carr is with School of Computing, University of Leeds. E-mail: H.Carr@leeds.ac.uk.*
- *Valerio Pascucci is with SCI Institute, University of Utah. E-mail: pascucci@sci.utah.edu*

Manuscript received 31 Mar. 2016; accepted 1 Aug. 2016. Date of publication 15 Aug. 2016; date of current version 23 Oct. 2016.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2016.2599040

root-finding techniques. In this paper, we contribute a method for direct ray casting of fiber surfaces by solving for the ray's intersection point with a fiber in both domain and range space, using either explicit definition of fiber control polygons or approximation as a distance field. In addition, we contribute a method for accelerating volume traversal via a uniform grid of multivariate intervals, and a strategy for analytically computing the gradient of the implicit fiber surface. We use our method to explore bivariate data from cosmology and chemistry simulations, and use it to explore a computational fluid dynamics coal combustion dataset in depth. In all, we show that direct fiber surface ray casting can achieve higher-quality visual results with lower preprocess time than fiber surface meshing or multifield direct volume rendering.

2 Related Work

In applying ray-tracing to fiber surfaces, several sets of literature become significant. First, we will give a brief sketch of the relevant work on isosurfaces and Marching Cubes. Second, we canvass the principal elements of direct isosurface ray tracing and volume rendering. Then in Section 3, we introduce the fiber surface, its relationship with isosurfaces, and the existing method for fiber surface extraction.

2.1 Isosurfaces and Marching Cubes

In rectilinear (structured) meshes, a trilinear interpolant is normally assumed, for which the correct isosurfaces are hyperbolic sheets [30]. The conventional method for isosurface rendering has been extraction via marching cubes [28] or some variant; paired with rasterization of the resulting mesh. Wilhelms and Van Gelder [40] proposed a min/max octree hierarchy that allowed the extraction process to only consider cells containing the surface. This concept has been extended with frustum and per-ray visibility culling [24, 25] and multiresolution volume data [39]. Livnat & Tricoche [26] effectively combined mesh extraction with point-based splatting for efficient isosurface rendering. Rosenthal & Linsen [33] partitioned the 3D domain using a kd-tree and computed points on the isosurface by linearly interpolating. Shi & Jaja [37] used a persistent octree indexing structure for accelerating isosurface extraction from volume data. Ljung and Ynnerman [27] explored visualizations of intersection curves from isosurfaces of multifields, a special type of fiber geometry.

2.2 Direct Isosurface Ray Casting

Hanrahan [10] first demonstrated a method for ray tracing algebraic implicit surfaces using the rule of signs to isolate roots. Parker et al. [31] used a ray tracer to directly render isosurfaces of volume data, using a hierarchical grid of macrocells as an acceleration structure. A single ray was tested for intersection inside a cell of eight voxel vertices, solving a cubic polynomial to find where the ray intersects the interpolant surface in that local cell. Hadwiger et al. [9] and Gobbetti et al. [7] demonstrated systems for real-time ray casting of isosurfaces on the GPU, using the rule of signs to locate roots and a secant method to solve for surface intersection. Similarly, Knoll et al. [18] proposed peak finding, a method for combining discrete isosurfacing and volume rendering. Direct isosurface rendering has also used to visualize isosurface of large dynamic particle datasets using density maps [20] and isosurfaces on face-centered cubic datasets [12].

2.3 Direct Volume Rendering

Direct volume rendering is a popular technique for classifying and visualizing multifield data. Laidlaw [22] first proposed multidimensional transfer functions for classification of MRI data. 2D transfer functions have been proposed for augmenting classification of univariate data with gradient magnitude [15] or curvature [13]. Early multifield classification approaches employed Gaussian kernels [16] or maximum-intensity projection [36]. Kruger & Westermann proposed a hardware accelerated volume ray casting method [21]. More recent work has focused on user interfaces for dimensionality reduction and transfer function generation, using scatterplots [6], kernel density estimation [23, 29], parallel coordinates [8, 41], and combinations of all three [42]. We refer interested readers to the comprehensive survey in [11].

Less work has addressed accurately reconstructing and rendering geometric features from multifield volume data. Kotava et al. [19] note that the sampling rate required for volume rendering with sharp feature reconstruction is the product of the frequencies of all component fields convolved via the transfer function. They propose sampling directly in transfer function space to efficiently reconstruct high-frequency features similar to our fiber surfaces.

3 Background: Fibers

To generalize isosurfaces to bivariate fields, instead of taking a single value $h \in R$, we take a single point $h \in R^2 (= Ranf)$, and find its inverse image $f^{-1}(h) = \{x \in Domf : f(x) = h\}$ to extract a *fiber* [34].

In the case of a bivariate volumetric field $f : R^3 \to R^2$, fibers are the intersection of the isosurfaces of each component of f, i.e. $f^{-1}(h) = f_1^{-1}(h_1) \bigcap f_2^{-1}(h_2)$. 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.

The inverse image $f^{-1}(h) = \{x \in Domf : f(x) = h\}$ of a single point $h \in R^2 (= Ranf)$ in the range space is a contour in the domain space. The inverse image $f^{-1}(h)$ of a line segment $l \in R^2$ in the range space is an open fiber surface in the domain space, the combination of multiple contours. The inverse image of closed polygon is one or multiple closed fiber surface. For computational purposes, therefore, it sufficed to deal with the case of a closed polygon, which we refer to as a *fiber surface control polygon* (*FSCP*). Since the computation is based on the marching cubes tables, we know that this is not an exact solution, and more exact solutions are desirable. Increasing the number of voxels covered by a FSCP enlarges the fiber surface in domain space.

Fiber surfaces [5] provide a surface extraction technique to get fiber surface based on marching cubes. In a tetrahedral mesh, the isosurfaces extracted in each cell are planar, and bivariate fibers are defined by intersection of two such isosurfaces. Barring degeneracies, these intersections will therefore be linear. For trilinear interpolant on a cubic mesh, however, the isosurfaces are hyperbolic sheets [30], and individual fibers will be found at the intersection of two such sheets, and may have multiple connected components. For example, if two copies of the most complex trilinear case (13.4.1) are rotated 90 degrees from each other, each edge of the cube will have an isosurface of each component intersecting it, and the intersection of the pair of isosurfaces can be forced to generate at least 12 fiber components. Moreover, given FSCPs that induce an arbitrary number of intersections of a fiber surface with a given cube, it is clear that exact fiber surfaces cannot be extracted as a polygonal mesh from trilinearly interpolated cubic cells, let alone higher-order interpolants. Recently, Klacansky et al. [14] implemented a highly-tuned implementation of fiber surface extraction based on marching tetrahedra. Sakurai et al. [35] also utilize fiber topology as analysis tool in mathmatics.

4 Ray Tracing Fiber Surfaces

If mesh geometry is not required, we can render fiber surfaces for these meshes using ray tracing methods, as with scalar fields. There are several advantages to this, chiefly performance, scalability, and the ability to render pixel-exact surfaces, irrespective of the underlying interpolant.

To ray trace fiber surfaces, as with isosurfaces, one may choose between analytical intersection methods for implicit surface patches defined within each voxel [31] or point-sampling techniques employing the rule of signs [10]. In this paper we employ the latter, assuming the multifield is piecewise-smooth, and that intersection points of the ray and fiber surface can be solved for arbitrary segments along the ray. This approach is more general but less robust, and sensitive to chosen segment size and the frequency of the underlying FSCP and multifield. However, as shown in similar work for fast intersection of isosurfaces [7, 9], the sacrifice in robustness is well-suited for OpenGL/GPU implementations with fast 3D texture interpolation.

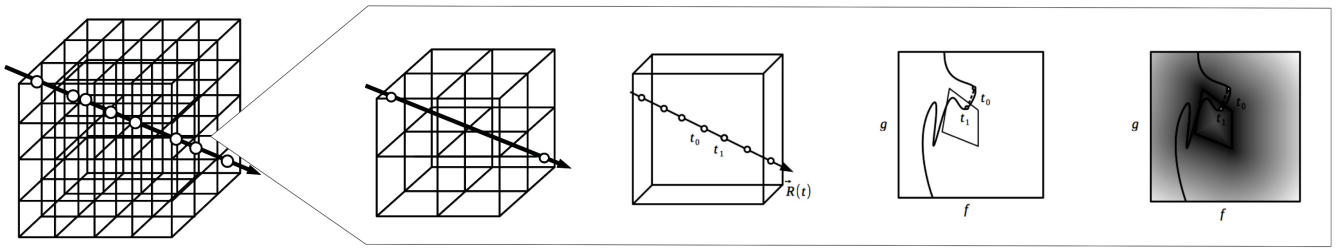

Fig. 2. From left to right: tracing a ray though grid of macrocells, tracing a ray in a 2^3 macrocell, sampling along the ray within the macrocell, finding the intersection point in range space, and using distance field to accelerate intersection test.

Explicit ray-FSCP intersection. The simplest fiber surfaces are defined by lines in the range, expressed as a general linear function $af_1 + bf_2 = C$. Given any point x on the fiber $f^{-1}(p)$ in the domain, $(f_1(x), f_2(x)) = p$. The problem of finding the point x on the ray, which also lies on the fiber $f^{-1}(p)$, in the domain can be seen as how to find the intersection point between ray segment and fiber control line segment in the range. In this work, domain-space segment size is given by distance along the normalized ray. One unit corresponds to one voxel width. Specifically, the segment size is $\delta t = t_1 - t_0$ in Fig. 4. Assuming the transfer function is piecewise-constant within the ray segment, a simple segment-segment intersection test can be used to find the fiber. Thus, ray casting from a FSCP is straightforward: we intersect the ray segment with all segments of polygon in the range to find the nearest intersecting point. If fiber surface is transparent, the ray segment will return the sum of the color of all intersected segments. Then, we move to the next ray segment, and search for a fiber surface.

Challenges with the naïve approach. Explicit ray-FSCP tests become cumbersome when the FSCP consists of numerous line segments. Moreover, if the segment size is too large, the actual path of ray segment in the range would become a complex curve which could intersect with fiber line segment multiple times. Thus, we must sample sufficiently in domain space so that piecewise-linear segments (chords) closely approximate the image of the ray in range space. Clearly, the smaller the segment size, the lower the performance of our method.

As with all ray tracing techniques, performance relies heavily on skipping large regions of empty space with no fiber surfaces, as well as efficiently intersecting fiber surfaces within each ray segment in between volume samples. To implement an efficient fiber surface ray caster, we require an acceleration structure and a conservative test for defining regions of space that definitively contain no fiber surfaces, allowing for occasional false positives but no false negatives. Guided by previous GPU and CPU isosurface ray casters [9, 18, 31], we developed a similar approach for fast ray casting of fiber surfaces.

5 Efficient Direct Ray Casting of Fiber Surfaces

For efficient direct ray casting of fiber surfaces, we use a two-stage precomputation step involving computation of a distance field and a uniform grid of macrocells; followed by a two-stage rendering step using uniform grid traversal and ray-FSCP intersection using a 2D distance field in range space. The overall method is sketched in Fig. 2 and described in Algorithm 1.

5.1 Distance Field Computation

Numerous 2D acceleration structures could be used to accelerate ray-FSCP intersection for multiple FSCPs consisting of complex polygons. We opted to use the distance field, the same approach used in the original fiber surfaces paper [5]. The distance field serves as both a rasterized approximation of arbitrarily complex FSCPs, an acceleration structure pointing to the nearest FSCP, and an "inside out" test indicating orientation of the (by necessity closed) polygon. Each element in the distance field specifies its minimum distance to the polygon. Positive and negative distances are used to distinguish outside and inside of the polygon, respectively. Ray-fiber surface intersection is then straightforward: by traversing elements in the distance field, we find the minimum distance from each element to all segments of FSCP and set the minimum distance as the value in the distance field.

One issue is the shape of FSCP, which can be either convex or concave. For the convex polygon, all segments are default as counter-clockwise. The cross product of two vectors, the direction vector of the polygon segment and the vector perpendicular to the segment, can be used to determine whether it is inside or outside. For concave polygons, the cross product alone cannot determine whether the element is inside or outside; this can be solved by using Jordan curve theorem. We shoot a semi-infinite ray from each points in the distance field and use many edges it crosses to determine whether the element is inside or outside.

5.2 Macrocell Grid Build and Traversal

We used a single-level uniform grid acceleration structure, traversed using 3D digital differential analyzer (*3DDDA*) method of Amantides and Wu [2]. We use cells of the grid, or *macrocells*, to store $n \times n \times n$ voxels and store the min/max values of both fields. In bivariate range space, this corresponds to a 2D box with vertices (f_{min}, g_{min}), (f_{min}, g_{max}), (f_{max}, g_{max}), and (f_{max}, g_{min}). Macrocells always include the minima and maxima of forward-adjacent voxels vertices, to prevent missing fibers. For example, a 1^3 macrocell computes intervals over a 2^3 volume. While other structures (octree, BVH, etc.) could prove faster, we chose the uniform grid for its efficient traversal on the GPU, and in particular in a GLSL shader.

Ray traversal of this grid occurs in the shader, using a cell-based 3DDDA traversal method. If the ray segment intersects with macrocell's min/max box, we sample along the ray within the cell to find intersection point. In effect, this provides a conservative test which ensures that macrocells are searched for fiber surfaces when the box *possibly* overlaps FSCPs in range space.

However, intersection test between ray segment and min/max box is expensive. Instead of doing that on the fly, we precompute it and save a boolean value to indicate whether there is fiber surface inside. Whenever FSCP is updated, we precompute the overlap between the range-space min/max of the macrocell and the FSCP. If there exists an overlap, which means fiber surface exists in that cell, we assign true to the corresponding cell the 3D uniform boolean grid, otherwise, we assign false.

5.3 Ray-FSCP intersection with the Distance Field

Assuming that the image of each ray segment in the domain space can be approximated as a straight segment in the range space, we can use linear interpolation to get the intersection point. Because of Bolzano's theorem, if distances of the segment endpoints are different signs, it is guaranteed that there must be at least a root in the interval. After fetching bivariate field data on the fly, we fetch its distance value. If distances are d_0 and d_1 and the segment endpoints are fs_0 and fs_1, the intersection point fs_t can be found using following equations.

$$d_t = |d_0/(d_0 - d_1)| \tag{1}$$

$$fs_t = fs_0 + d_t * (fs_1 - fs_0) \tag{2}$$

The main benefit of the distance field is accelerating the intersection test. Compared with a 2D segment-based intersection test, our method only needs to check signs of two values. Crucially, the distance field test requires the same fixed time regardless of how many segments each FSCP has. One limitation is that distance fields of multiple FSCPs cannot be merged – thus, each FSCP has its own distance field. In our implementation, we use 2D textures to store and upload distance fields to the GPU. Each channel of the texture can be used to store one FSCP's distance field, and we visualize up to 4 fiber surfaces at a time. One limitation is the distance field only works for closed polygons, because the sign test (indicating presence of a root) requires us to know whether a point is "inside" or "outside" the FSCP.

5.4 Gradient Computation and Shading

Once intersection point is detected, the shading color should be calculated based on its position in domain. Gradient is a crucial component for correct shading calculation. In our examples, we shade multiple data gradients ∇f_1 and ∇f_2 separately, using multiple central-differences neighbor stencils, because ∇f_1 and ∇f_2 tends to exhibit higher frequency than separate individual gradients. Thus, for fiber $g(x) = af_1(x) + bf_2(x)$, the gradient $\nabla g(x)$ is the linear combination of ∇f_1 and ∇f_2, which means $\nabla C = g(x) = a\nabla f_1(x) + b\nabla f_2(x)$ and (a,b) is coordinate of intersection point in the range.

Algorithm 1 Fiber surface ray tracing

```
1: // traverse macrocell grid using using 3DDDA method
2: for each macrocell do
3:     if there is fiber surface inside the macrocell then
4:         // do ray marching within the macrocell
5:         for each ray segment [p0, p1] do
6:             fs0 ← fetch_bivariate_volume_data(p0)
7:             fs1 ← fetch_bivariate_volume_data(p1)
8:             d0 ← fetch_distance(fs0)
9:             d1 ← fetch_distance(fs1)
10:            if d0 * d1 < 0 then
11:                find intersection point fst
12:                shade(fst)
13:            end if
14:        end for
15:    end if
16: end for
```

Algorithm 1 summarizes the our fiber surface ray tracing method. The outer loop (lines 2-16) traverses on macrocell grid using 3DDDA and the inner loop (lines 5 -14) performs ray marching within the macrocell. At each macrocell, we first fetch the precomputed boolean value as described in section 5.2. Distance value for endpoints of ray segment are also precomputed as described in section 5.1.

5.5 Fiber Surface Control Polygon Editor

The main benefit of fiber surface ray casting is exploring fiber surfaces interactively. We implement a FSCP editor, which allow users to drag FSCP vertices and drag the entire polygon. Once FSCP is updated, distance fields are updated and the boolean value of cells are updated based on new distance fields.

As shown in Fig. 3, the 2D (joint) histogram is colored by the number of voxels in each bins. Each FSCP has its corresponding fiber surfaces. By dragging the polygon center, users are able to move the entire polygon to any place they want. we also allow user to click and drag the polygon corner to explore fiber surface features. Note that once FSCP is updated, we have to precompute distance field and build boolean grid of macrocells again.

Using the editor to manipulate one FSCP at a time is intuitive. However, choosing interesting fiber surfaces can be challenging – significantly more difficult than choosing interesting isosurfaces of univariate data. Generally, we find that FSCPs yield interesting fiber surfaces where the gradient of values in range space is high, i.e. around "rooster tails" (Fig. 3), or in regions where the fields exhibit clear nonlinear relationships. It is also helpful to shrink and grow FSCPs, watching the corresponding surfaces expand and contract to correspond to a single point (fiber).

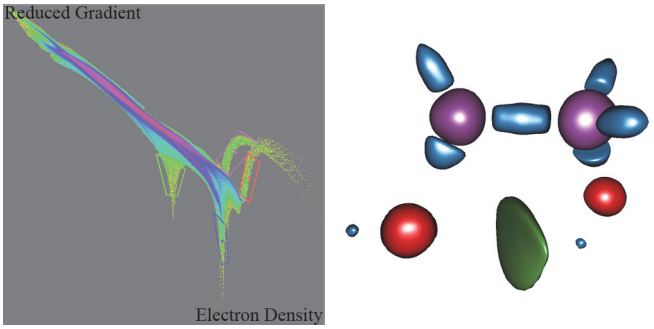

Fig. 3. Our FSCP editor classifying the Ethane-Diol dataset from [5]. Left: colored joint histogram and four FSCPs from Ethane-Diol. Right: fiber surface visualization based on the given FSCP.

5.6 Implementation

We implemented all preprocess code in C++ on the CPU, and used a GLSL shader-based ray caster for the actual fiber surface ray casting. However, our algorithm is intended to be generic, using as little of the fixed-function OpenGL pipeline as possible, and suitable for implementation in fast CPU frameworks, e.g., [1]. Besides sending bivariate volume data to the GPU, the distance field is sent to the GPU as a 2D texture. We utilize 3D texture to store four boolean values of each macrocell, indicating whether the corresponding fiber surface is contained. The 3D texture is updated whenever the FSCPs are modified as the preprocess step and uploaded on the GPU. We use multithreading in OpenMP to update the distance fields and boolean macrocell values dynamically.

6 Results

All benchmarks on CPU side were conducted on a Intel Core i7 3930K Processor 3.20GHz with 32GB memory. We test GPU side performance on NVIDIA GeForce GTX 780 and NVIDIA GeForce GT 650M 512 MB video card. We intentionally use a mobile GPU card to demonstrate our algorithm doesn't require any high end graphic card to get interactivity. Unless otherwise stated, we used a frame buffer of 1024×1024 for all frame rate numbers on GT 650M.

6.1 Overall Performance

In this section, we report performance statistics of our implementation. Generally, rendering performance is related less to data size, and more to the complexity of the fiber surfaces defined by the FSCPs. Preprocess time relates chiefly to the size of macrocells and complexity of FSCPs.

Preprocess time. Though fiber surface ray casting is a direct visualization method, some analyses require preprocess time. Table 1 gives the precomputation time for building our boolean grid of macrocells. Macrocell computation achieves reasonable parallelization with OpenMP: 6 threads yield a preprocess speedup of $2\times$-$4\times$ depending on the dataset. The total precomputation time is the sum of the time of updating the distance field and the time of updating the macrocell boolean value as long as FSCP is updated. Clearly, preprocess time depends on the total number of voxels linearly and more voxels per macrocell reduce the number of macrocells that need to be computed. We find that 2^3 macrocells generally give the best performance, but that larger data (i.e., BSF) do not suffer a high penalty from 4^3 or 8^3 macrocells. Generally, smoother data and FSCPs with more empty space to skip can achieve good performance with coarser macrocells, preserving interactivity even when exploring larger datasets.

The time distance fields updating on the fly is fixed and only depends on the size of distance field dimensions. In our tests, a 64×64 array is accurate enough for the distance field. According to experiments, it takes around 1 to 2 milliseconds to compute the distance field for

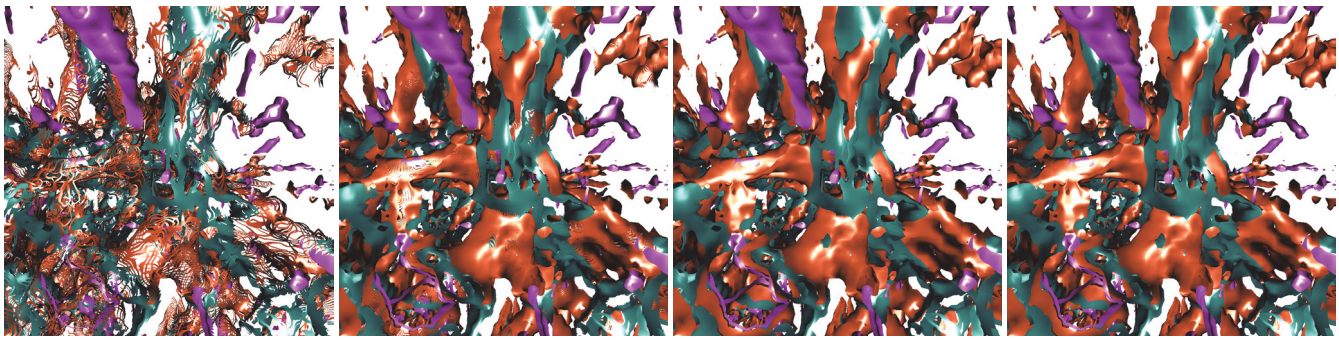

Fig. 4. Fiber surfaces with different segment sizes (in voxel units) along the ray on GTX 780 with 2^3 voxels per macrocell. From left to right: segment sizes of 0.4, 0.1, 0.025, and 0.00625, rendering at 58.73, 33.38, 11.47, and 3.21 fps respectively (see Table 2). The image converges with a segment size of 0.03.

Table 1. Distance field and macrocell precomputation time in seconds, which determines interactivity of the FSCP editor.

# Voxels/Macrocell		1^3	2^3	4^3	8^3
Dataset	# Voxels				
Ethane-Diol ($115\times116\times135$)	1.8M	0.321	0.036	0.007	0.003
OFC rt-time ($427\times152\times152$)	9.9M	1.481	0.182	0.036	0.005
Enzo (256^3)	16M	2.568	0.299	0.039	0.005
BSF ($665\times290\times170$)	32M	4.775	0.612	0.077	0.010
Enzo (512^3)	134M	–	2.613	0.303	0.040

Table 2. Frame rate performance for 256^3 Enzo with varying macrocell sizes on two different GPU architectures.

# Voxels/Macrocell	1^3	2^3	4^3	8^3	1^3	2^3	4^3	8^3
Segment size	GT 650M				GTX 780			
0.4	4.11	6.51	**7.25**	4.32	49.44	**58.73**	54.19	40.62
0.2	3.95	**5.48**	4.98	2.54	44.96	**47.42**	37.22	25.80
0.1	3.51	**3.98**	3.06	1.43	**36.77**	33.38	23.49	15.60
0.05	**2.75**	2.54	1.75	0.76	**25.74**	20.35	13.38	8.61
0.025					**16.10**	11.47	7.19	4.53
0.012					**9.28**	6.16	3.77	2.33
0.006					**5.06**	3.21	1.93	1.19
0.003					**2.65**	1.64	0.94	0.56

4 FSCP with 4 segments for each polygon. Generally, distance field computation is on the order of milliseconds and not a bottleneck; we included it in the overall preprocess time in Table 1.

Rather than use a continuous scatterplot [4] as in the original fiber surface paper [5], we precompute a 2D histogram and color each pixel based on the number of samples in the bin. We chose discrete histograms for ease of implementation, and because they better convey sparse regions of range space. Histogram computation is $O(N)$, generally on the order of seconds for our largest data, and could be computed simultaneously alongside macrocells. Since the histogram is a visual aid and not necessary for actual rendering, we omit this time from our numbers in Table 1.

Rendering performance. Table 2 gives the frame rate per second for 256^3 Enzo with varying macrocell size. Note that one unit of segment size corresponds to one voxel width in domain space. Generally, performance without any acceleration structure is $4\times$-$8\times$ slower than that with macrocells of 2^3-voxels. Compared with the mobile GPU, the discrete GPU (NVIDIA 780 GTX) accelerates performance by $7\times$-$12\times$ for 256^3 Enzo. As seen in Table 3, performance is generally interactive even on mobile GPU hardware. Exploring fiber surface by changing the FSCP depends on both preprocess time and rendering time. Therefore, even though 1^3 or 2^3 macrocell allows faster rendering time, we recommend using 4^3 macrocells for best overall interactive rate.

Our implementation allows users to change segment size at run time; full interactivity can be ensured by increasing the segment size when the user moves the camera, if necessary.

Rendering performance depends on the number of rays cast and the number of samples per ray, as well as the empty space in the dataset and the efficacy of the macrocells. Table 3 gives the frame rate per second for datasets with different resolution in segment size

Table 3. Frame rates with segment size 0.1 on our mobile GPU.

# Voxels/Macrocell	1^3	2^3	4^3	8^3
Dataset				
Homo-Lumo (41^3)	19.90	**23.25**	20.11	12.92
Ethane-Diol ($115\times116\times135$)	10.97	**17.23**	16.55	7.95
OFC rt-time ($427\times152\times152$)	23.05	**23.83**	19.40	14.67
Enzo (256^3)	3.51	**3.98**	3.06	1.43
BSF ($665\times290\times170$)	10.30	14.90	**16.64**	14.95
Enzo (512^3)	1.88	**2.76**	2.31	1.50

0.1, at 1024×1024 (1 megapixel). For small datasets, such as Homo-Lumo, our technique easily achieves 20 fps even on our mobile GPU. Frame rate depends somewhat on data size, but not linearly – with a uniform grid, complexity is best-case $O(1)$ and worst-case $(O(N^{3/2}))$. Comparing two Enzo datasets with different size, we note that fps for the 512^3 dataset is only $.77\times$ that of the 256^3 dataset, despite $8\times$ more voxels. This performance can be attributed to macrocells and early termination of the ray casting approach.

6.2 Robustness and Quality

Generally, the frequency of the fiber surface is governed by the product of the frequencies of both fields, and the frequency of the specified FSCP. However, since our method relies on the rule of signs [10], it is impossible to guarantee a sampling rate that would ensure robust root isolation. In other words, the main drawback of our method is that we cannot guarantee the intersection point will never be missed and assumes there is only one intersection point in the interval. When segment endpoints have distance field values with the same sign, this can result in false negatives and missed fiber surfaces. When they have different signs, false positives may occur. If the FSCP is very small, like a thin line, our intersection may lose parts of fiber surface as well. These issues are more prevalent for high-frequency data such as Enzo. As shown in Fig. 4, the impact of segment size on the fiber surface rendering, reducing segment size will alleviate this issue, but at the expense of performance. Though our method cannot guarantee never missing fibers, Fig. 4 demonstrates that the artifacts will become less and converge, when segment size is sufficiently small. However, the right segment size doesn't depends on data size, but only depends on the frequency of data and curvature of FSCP. In theory, if segment size is infinite small, then intersection would never be missed. For very small segment sizes, we are occasionally limited by floating point error and chosen discretization of the distance field.

Higher-order filtering A major benefit of our approach is that it operates independently of the choice of interpolation filter. As long as the underlying multifield is piecewise smooth, and the user has provided an interpolation kernel, the rule-of-signs method will suffice to extract fiber surface geometry, with the noted robustness limitations. In contrast, mesh extraction methods would require subdivision, and analytical ray-surface intersection would be potentially intractable for arbitrary higher-order interpolants. As demonstration, we implemented a 3D tricubic B-spline filter using the method of Sigg et al. [38]. As

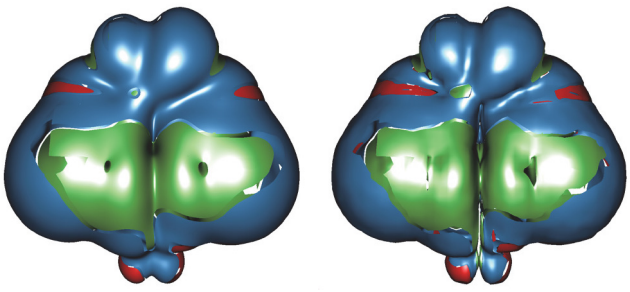

Fig. 5. Higher-order interpolation of the Homo-Lumo dataset. Left: rendering image with B-spline filter (8 fps). Right: without B-spline filter (23 fps).

shown in Fig. 5, higher order interpolation ensures smoother reconstruction of fiber surfaces, though at a cost of 3× time than base method. Our approach would extend to any interpolation filter for structured or unstructured data.

6.3 Comparison with Extraction and Volume Rendering

Fig. 3 and 6 show the results of fiber surface ray casting on Ethane-Diol and Enzo, two datasets from the original paper [5]. In comparing our techniques, we consider frame rate performance of ray casting vs rasterization, the time required to generate fiber surfaces via marching cubes extraction, and image quality, particularly near edges of fiber surface geometry. For Ethane-Diol, our directly ray-cast fiber surfaces deliver pixel-exact results and allow for correct rendering of sharp features along edges, for example the edge of the green fiber surface corresponding to the non-covalent bond interaction between oxygen atoms. This stands in contrast to the marching cubes results (see, e.g., Fig. 1 and 4 in [5], which exhibit noticeable facets. Even for this small dataset, they report an extraction time of 1.7s, implying significantly less interactive specification of fiber surfaces than our 16 fps. The Enzo dataset (Fig. 6), while not particularly large, is more entropic and yields complex fiber surface geometry. Our system renders at 3.98 fps on a mobile integrated GPU (Table 3); this compares favorably to the extraction time of serial marching cubes in VTK (17 seconds in [5]), and is in fact better than the rendering time of the extracted fiber surfaces (0.69 seconds). While the original paper did not consider the larger Enzo data, we note that performance for a 512^3 Enzo dataset is not significantly worse (2.76 fps in Table 3). Recently, Klacansky et al. [14] demonstrate a highly-tuned implementation of fiber surface extraction based on marching tetrahedra employing a BVH for improved performance. Though the underlying techniques are different, direct ray casting is generally faster for large or entropic data (33 fps for ray casting vs 3 fps for extraction on the 256^3 Enzo dataset). Generally, sublinear-time ray casting performs better with large data than linear-time extraction and rasterization approaches. Overall, both methods have advantages: extraction allows for analysis of the mesh as well as rendering. However, the per-pixel accuracy, inherent scalability and support for arbitrary interpolation filters make ray casting compelling.

In Fig. 7 we compare fiber surface ray casting to direct volume rendering. For the volume rendering transfer function, we convert the distance field to a color sample with a specified threshold; distance field values beneath that threshold have color and opacity of zero. This threshold serves as a fine to coarse "brush size" specifying the width of fiber surfaces (for actual fiber surfaces, that brush size is infinitesimally small, or zero). We find that for "fine" brush sizes (threshold .001), we find that direct fiber surface ray casting is an order of magnitude faster than equivalent volume rendering. For a "coarse" brush size (threshold .05), volume rendering performance is clearly competitive – however one loses the ability to precisely classify regions of range space, and the resulting renderings are more susceptible to occlusion and clutter. Although strategies for classifying and sampling vary widely with dataset, we have found our approach to be roughly an order of magnitude faster than the method of Kotava et al. [19] for comparable transfer functions.

6.4 Use Case: Combustion

To evaluate how fiber surface ray casting could be used in practice, we collaborated with mechnical engineers at the University of Utah Carbon Capture Multidisisciplinary Simulation Center (CCMSC) to investigate multifield volume data from coal boiler simulations. Specifically, we analyze data from two simulations: the large "BSF" boiler in Fig. 1 and a smaller oxy-fuel combustor simulation "OFC". These simulations were created in the Uintah framework, using the Arches computational code. These are fluid dynamic simulations with 130 fields pertaining to different physical attributes of the systems. Here, we explore several fields, such as the gas-phase residence time, O_2 mass fraction, devolatilization and char oxidation rates, particle temperatures, and radiative and convective energy transfer rates of various-sized coal particles. Fiber surfaces may provide a new tool for comparative analysis of these multifield data, and ultimately assist in the simulation, validation and design of coal combustion systems.

Devolatization vs char oxidation. During the combustion of coal particles, devolatilization is the process where heat causes the particles to partially decompose into fuel rich gases. Char oxidation is the heterogeneous reaction of the organic material in the particle with oxidizers that are able to diffuse to the surface of the particle. The simulation measures the rate at which these separate phenomena occur ($kg/m^3/s$) in two fields, "η_1" for devolatization and "η_2" for char oxidation. In Fig. 8, we use fiber surfaces to compare these two quantities. The fiber surfaces show that char oxidation dominates at later residence times, which is as expected. In the leftmost image, we compare the η_1 and η_2 fields directly. The fiber surfaces show that devolatization and char oxidation occur simultaneously in a narrow central column above the inlet. Devolatilization dominates close to the inlet, and char oxidation dominates further away from the inlet as O_2 is able to get to the particle surface. Then, in the two right images we compare them separately with respect to residence time (in seconds) over the course of the simulation.

Radiation vs convection. Another question we sought to answer is whether the heat transfer in the coal combustion system is predominately due to radiation or convection. In Fig. 9, we compare the radiation and convection heat transfer rates, both entities in W/m^3. We see from the 2D histograms that radiation dominates the energy output of combustion inside the boiler. The red and green fiber surfaces show the respective regions dominated by radiation and convection, respectively – we note that for the largest particle size, convective regions extend further into the center of the furnace. The blue fiber surfaces show that convection and radiation occur together only in a small region near the tip of the inlet, and the three cooling ports where the temperature differences are greatest.

Residence time vs particle temperature. Lastly, in Fig. 10, we compare residence time with the temperature of the three different sizes of coal particles. Curiously, although the 2D histograms exhibit some differences, classification of fiber surfaces along the horizontal (time) axis appear nearly identical. While at first we thought this was a mistake, further experimentation showed that indeed, the fiber surfaces are similar with respect to the time axis. Slight variations can be seen in the green fiber surface geometry (early residence time, very close to the inlet), but our overall finding is that most particles quickly exhibit the same temperature, regardless of size. This conclusion would be difficult to reach examining the 2D histogram or individual scalar fields alone.

Interactive exploration of fiber surfaces in the OFC combustion data sheds light on how to use them for analysis of scientific data. Conventionally, these multifield data would be visualized side-by-side via single-field volume rendering. Bivariate transfer functions would help us better correlate volume renderings against the joint histogram, but lack precision. Crucially, fiber surfaces let us explore not only whether fields of the simulation exhibit similar behavior, but where exactly that behavior occurs in domain space.

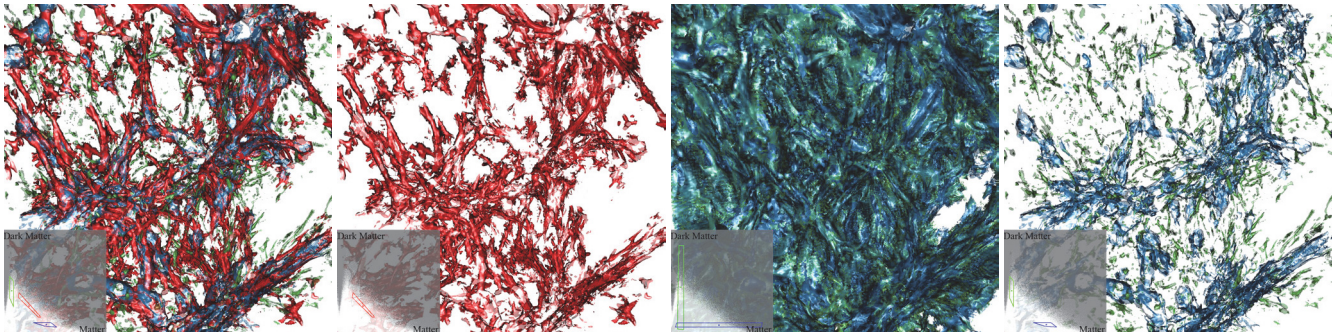

Fig. 6. Ray-cast fiber surfaces from the Enzo cosmology dataset, classifying matter (X axis) and dark matter density (Y axis). The red fiber surfaces show the filamentary backbone of the cosmology simulation (red), combining both fields. Isocontours of matter (green) and dark matter (blue), implemented as horizontal and vertical FSCPs, do not show this independently, even when restricted with respect to the other value. Direct ray casting at 3–5 fps on an NVIDIA Geforce GT 650M enables faster interaction than marching cubes-based fiber surface meshes (17 seconds extraction, 0.6 seconds rasterization).

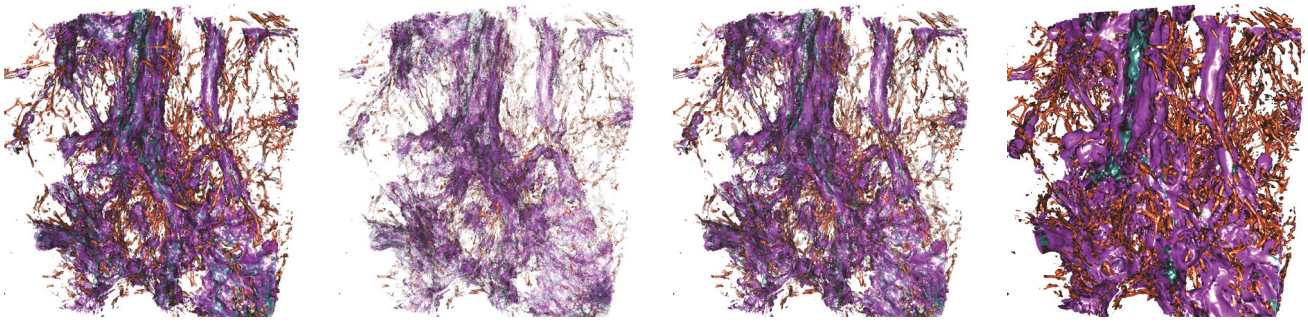

Fig. 7. From left to right: fiber surface ray casting with segment size 0.1 – 3.6 fps; volume rendering with segment size 0.1 ("fine" distance field threshold 0.001) – 3.6 fps; volume rendering with segment size 0.025, ("fine" distance field threshold 0.001) – 0.9 fps; and volume rendering to find the root ("coarse" distance field threshold = 0.05) – 4.3 fps. In these examples, rendering exact fiber geometry with volume rendering requires 4× the sampling rate, and thus lower performance. While volume rendering with a coarse threshold can render smooth features quickly, it introduces clutter and lacks the precise classification of of fiber surfaces.

7 Conclusion and Future Work

We have demonstrated a ray casting method for directly rendering fiber surfaces from bivariate data, defined by control polygons specified in range space. Our method requires negligible precomputation time and enables interactive classification of all datasets we considered. In particular, this enables us to explore larger volumes such as cosmology and combustion datasets, without having to generate fiber surfaces in offline precomputation. Moreover, ray casting ensures pixel-accurate rendering of fiber surface geometry and allows for the use of higher-order interpolation filters. As shown in our case studies, fiber surfaces are useful in discovering not only how scalar fields relate in quantity via a 2D (joint) histogram, but correlating that to *where* they relate in the domain. The ability to draw boundaries in range space, and use that to define regions of interest in the domain, is particularly helpful when scientists already use 2D histograms to understand their data.

There are several opportunities for improvement. The main limitation of our approach is that it relies on point sampling and the rule of signs, similar to GPU isosurface ray casting methods [9, 18]. This is fast, but not always accurate, as explored in Fig. 4. Because our multifield and fiber surfaces are piecewise-smooth, increasing segment size does ultimately result in accurate rendering of surface geometry. However, ray-fiber intersection within each cell [3, 31] or interval arithmetic [17] would be more robust alternatives. Moreover, fiber surface geometry is extremely sensitive to the dynamic range of multifield data. Solutions for dynamically zooming, panning and even clustering in range space would help better define FSCPs. Extracting fiber surfaces from double-precision data would be desirable. To this end, integration into a multifield data curation platform such as ViSUS [32], or using CPUs for access to larger memory, may be desirable. Lastly, fiber surfaces are defined for bivariate scalar fields; new research is required to extend the concept to higher-dimensional multifield data.

Acknowledgments

This work is supported in part by NSF: CGV: Award:1314896, NSF CISE ACI-0904631, DOE/Codesign P01180734, DOE/SciDAC DESC0007446, CCMSC DE-NA0002375, and PIPER: ER26142 DE-SC0010498. Additional support comes from the Intel Parallel Computing Centers program, the Argonne Leadership Computing Facility, and EPSRC Grant EP/J013072/1. Kui Wu was supported by NSF grant #1538593. This material is also based upon work supported by the Department of Energy, National Nuclear Security Administration, under Award Number(s) DE-NA0002375.We thank Joe Insley, Mike Papka and Julius Jellinek at ANL for the Enzo and HOMO-LUMO datasets, and Julien Tierny at the Sorbonne for the ethane diol data.

References

[1] Intel OSPRay Ray Tracing Visualization Engine. http://www.ospray.org.
[2] J. Amanatides and A. Woo. A Fast Voxel Traversal Algorithm for Ray Tracing. In *EG 1987-Technical Papers*. Eurographics Association, 1987. doi: 10.2312/egtp.19871000
[3] M. Ament, D. Weiskopf, and H. Carr. Direct Interval Volume Visualization. *IEEE transactions on visualization and computer graphics*, 16(6):1505–1514, 2010.
[4] S. Bachthaler and D. Weiskopf. Continuous Scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1428–1435, Nov 2008. doi: 10.1109/TVCG.2008.119
[5] H. Carr, Z. Geng, J. Tierny, A. Chattopadhyay, and A. Knoll. Fiber Surfaces: Generalizing Isosurfaces to Bivariate Data. *Computer Graphics Forum*, 34(3):241–250, 2015. doi: 10.1111/cgf.12636
[6] H. Doleisch. SimVis: Interactive Visual Analysis of Large and Time-Dependent 3D Simulation Data. In *2007 Winter Simulation Conference*, pp. 712–720, Dec 2007. doi: 10.1109/WSC.2007.4419665
[7] E. Gobbetti, F. Marton, and J. A. Iglesias Guitián. A Single-pass GPU Ray Casting Framework for Interactive Out-of-core Rendering of Massive

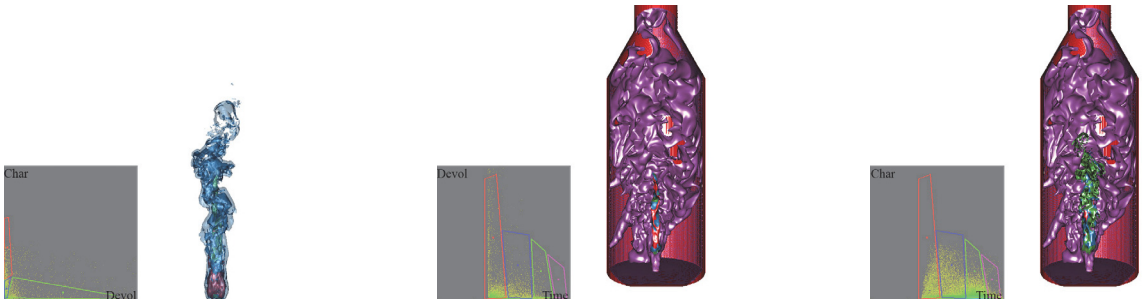

Fig. 8. Devolatilization (η_1) vs char oxidation (η_2) rates (kg/m^3/s). From left to right: η_1 vs η_2, residence time vs η_1, and residence time vs η_2. Fiber surfaces show that devolatilization dominates close to the inlet, whereas char oxidation spreads out towards the center of the furnace.

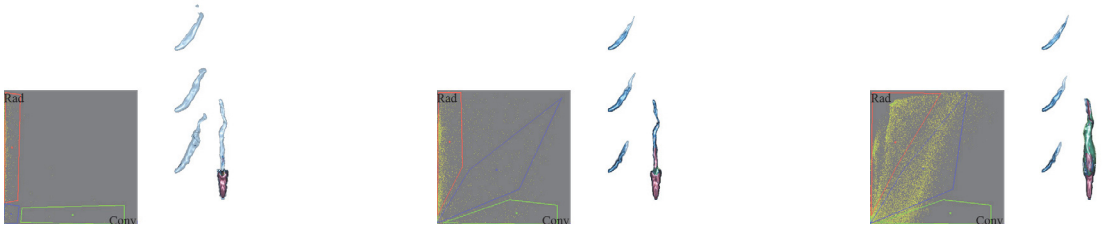

Fig. 9. Radiative and convective heat transfer rates in the OFC combustion dataset, for various sized coal particles (15 μm, 50 μm, 150 μm, from left to right). The 2D histogram shows the heat transfer mechanism for both small and large particles are dominated by radiation; for larger particles convection tends to become more significant.

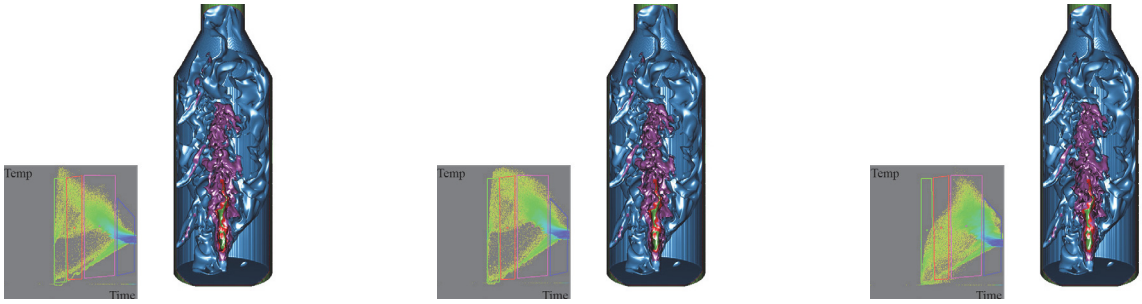

Fig. 10. Residence time vs. temperature of various sized coal particles, from left to right 15 μm, 50 μm, and 150 μm. While the 2D histogram shows that larger particles take a longer time to heat up, the fiber surfaces show that the overall geometry of the temperature field inside the chamber is similar for all particle sizes, except for a very small region close to the inlet.

Volumetric Datasets. *The Visual Computer*, 24(7):797–806, July 2008. doi: 10.1007/s00371-008-0261-9

[8] H. Guo, H. Xiao, and X. Yuan. Multi-dimensional Transfer Function Design Based on Flexible Dimension Projection Embedded in Parallel Coordinates. In *2011 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 19–26, March 2011. doi: 10.1109/PACIFICVIS.2011.5742368

[9] M. Hadwiger, C. Sigg, H. Scharsach, K. Bühler, and M. Gross. Real-Time Ray-Casting and Advanced Shading of Discrete Isosurfaces. *Computer Graphics Forum*, 24(3):303–312, 2005. doi: 10.1111/j.1467-8659.2005.00855.x

[10] P. Hanrahan. Ray Tracing Algebraic Surfaces. In *Proceedings of the 10th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '83, pp. 83–90. ACM, New York, NY, USA, 1983. doi: 10.1145/800059.801136

[11] 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, March 2013. doi: 10.1109/TVCG.2012.110

[12] M. Kim. GPU Isosurface Raycasting of FCC Datasets. *Graphical Models*, 75(2):90–101, Mar. 2013. doi: 10.1016/j.gmod.2012.11.001

[13] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Moller. Curvature-based Transfer Functions for Direct Volume Rendering: Methods and Applications. In *Visualization, 2003. IEEE*, pp. 513–520, Oct 2003. doi: 10.1109/VISUAL.2003.1250414

[14] P. Klacansky, J. Tierny, H. Carr, and Z. Geng. Fast and Exact Fiber Surfaces for Tetrahedral Meshes. *IEEE Transactions on Visualization and Computer Graphics*, PP(99):1–1, 2016. doi: 10.1109/TVCG.2016.2570215

[15] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional Transfer Functions for Interactive Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, Jul 2002. doi: 10.1109/TVCG.2002.1021579

[16] J. Kniss, S. Premŏze, M. Ikits, A. Lefohn, C. Hansen, and E. Praun. Gaussian Transfer Functions for Multi-field Volume Visualization. In *Visualization, 2003. IEEE*, pp. 497–504, Oct 2003. doi: 10.1109/VISUAL.2003.1250412

[17] A. Knoll, Y. Hijazi, A. Kensler, M. Schott, C. Hansen, and H. Hagen. Fast Ray Tracing of Arbitrary Implicit Surfaces with Interval and Affine Arithmetic. In *Computer Graphics Forum*, vol. 28, pp. 26–40. Wiley Online Library, 2009.

[18] A. Knoll, Y. Hijazi, R. Westerteiger, M. Schott, C. Hansen, and H. Hagen. Volume Ray Casting with Peak Finding and Differential Sampling. *IEEE*

Transactions on Visualization and Computer Graphics, 15(6):1571–1578, Nov 2009. doi: 10.1109/TVCG.2009.204

[19] N. Kotava, A. Knoll, M. Schott, C. Garth, X. Tricoche, C. Kessler, E. Cohen, C. D. Hansen, M. E. Papka, and H. Hagen. Volume Rendering with Multidimensional Peak Finding. In *2012 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 161–168, Feb 2012. doi: 10.1109/PacificVis.2012.6183587

[20] M. Krone, J. Stone, T. Ertl, and K. Schulten. Fast Visualization of Gaussian Density Surfaces for Molecular Dynamics and Particle System Trajectories. In *EuroVis - Short Papers*. The Eurographics Association, 2012. doi: EuroVisShort2012/067-071

[21] J. Kruger and R. Westermann. Acceleration Techniques for GPU-Based Volume Rendering. In *Visualization, 2003. IEEE*, pp. 287–292, Oct 2003. doi: 10.1109/VISUAL.2003.1250384

[22] D. H. Laidlaw. Geometric Model Extraction from Magnetic Resonance Volume Data. Technical report, PhD thesis , Caltech, 1995.

[23] L. Liu and W. H. Wong. Multivariate Density Estimation Based on Adaptive Partitioning: Convergence Rate, Variable Selection and Spatial Adaptation. *ArXiv e-prints*, Jan. 2014.

[24] Z. Liu, A. Finkelstein, and K. Li. Improving Progressive View-dependent Isosurface Propagation. *Computers & Graphics*, 26(2):209–218, April 2002.

[25] Y. Livnat and C. Hansen. View Dependent Isosurface Extraction. In *Visualization '98. Proceedings*, VIS '98, pp. 175–180. IEEE Computer Society Press, Los Alamitos, CA, USA, 1998.

[26] Y. Livnat and X. Tricoche. Interactive Point-Based Isosurface Extraction. In *Visualization, 2004. IEEE*, pp. 457–464, Oct 2004. doi: 10.1109/VISUAL.2004.52

[27] P. Ljung and A. Ynnerman. Extraction of Intersection Curves from Isosurfaces on Co-located 3D Grids. In *SIGRAD2003 Proceedings*, p. 23. Citeseer, 2003.

[28] W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pp. 163–169. ACM, New York, NY, USA, 1987. doi: 10.1145/37401.37422

[29] R. Maciejewski, I. Woo, W. Chen, and D. Ebert. Structuring Feature Space: A Non-Parametric Method for Volumetric Transfer Function Generation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1473–1480, Nov 2009. doi: 10.1109/TVCG.2009.185

[30] G. Nielson. On marching cubes. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):283–297, July 2003. doi: 10.1109/TVCG.2003.1207437

[31] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P. P. Sloan. Interactive Ray Tracing for Isosurface Rendering. In *Visualization '98. Proceedings*, pp. 233–238, Oct 1998. doi: 10.1109/VISUAL.1998.745713

[32] V. Pascucci, G. Scorzelli, B. Summa, P. Bremer, A. Gyulassy, C. Christensen, S. Philip, and S. Kumar. The ViSUS Visualization Framework. *EW Bethel, HC (LBNL), and CH (UofU), editors, High Performance Visualization: Enabling Extreme-Scale Scientific Insight, Chapman and Hall/CRC Computational Science*, 2012.

[33] P. Rosenthal and L. Linsen. Direct Isosurface Extraction from Scattered Volume Data. In *Proceedings of the Eighth Joint Eurographics / IEEE VGTC Conference on Visualization*, EUROVIS'06, pp. 99–106. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2006. doi: 099-106

[34] O. Saeki. *Topology of Singular Fibers of Differentiable Maps*. Lecture Notes in Mathematics. Springer-Verlag Berlin Heidelberg, 2004.

[35] D. Sakurai, O. Saeki, H. Carr, H. Y. Wu, T. Yamamoto, D. Duke, and S. Takahashi. Interactive Visualization for Singular Fibers of Functions f : R3 → R2. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):945–954, Jan 2016. doi: 10.1109/TVCG.2015.2467433

[36] J. P. Schulze and A. Rice. Real-Time Volume Rendering of Four Channel Data Sets. In *Visualization, 2004. IEEE*, pp. 34p–34p, Oct 2004. doi: 10.1109/VISUAL.2004.89

[37] Q. Shi and J. JaJa. Isosurface Extraction and Spatial Filtering using Persistent Octree (POT). *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1283–1290, Sept 2006. doi: 10.1109/TVCG.2006.157

[38] C. Sigg and M. Hadwiger. Fast Third-order Texture Filtering. *GPU gems*, 2:313–329, 2005.

[39] R. Westermann, L. Kobbelt, and T. Ertl. Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. *The Visual Computer*, 15(2):100–111, 1999. doi: 10.1007/s003710050165

[40] J. Wilhelms and A. Van Gelder. Octrees for Faster Isosurface Generation. *ACM Trans. Graph.*, 11(3):201–227, July 1992. doi: 10.1145/130881.130882

[41] X. Zhao and A. Kaufman. Multi-dimensional Reduction and Transfer Function Design Using Parallel Coordinates. In *Proceedings of the 8th IEEE/EG International Conference on Volume Graphics*, VG'10, pp. 69–76. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2010. doi: 069-076

[42] L. Zhou and C. Hansen. Transfer function design based on user selected samples for intuitive multivariate volume exploration. In *2013 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 73–80, Feb 2013. doi: 10.1109/PacificVis.2013.6596130