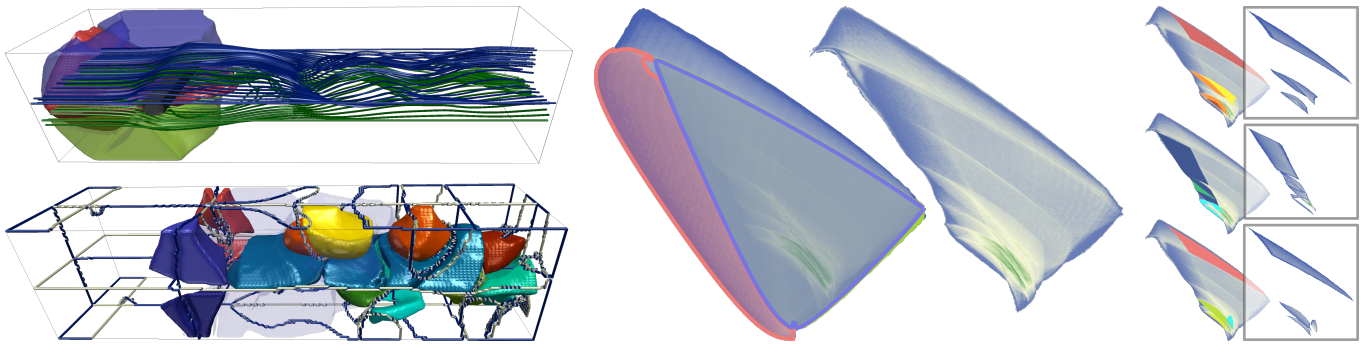
# Jacobi Fiber Surfaces for Bivariate Reeb Space Computation

Julien Tierny, Hamish Carr

# Jacobi Fiber Surfaces for Bivariate Reeb Space Computation

Julien Tierny and Hamish Carr



(a) (b) (c)

Fig. 1. The Reeb space $\mathscr{R}(f)$ of a bivariate function $f$ segments the domain $\mathscr{M}$ into regions where *fibers*, multivariate analogs of level-sets, are made of a single connected component. This allows the automatic separation of volumetric regions that project to the same areas in the continuous scatterplot (CSP). In this flow example (streamlines are shown in green and blue in (a), top), the Reeb space of the velocity and curl magnitudes is used to segment the data into its main features (a). The largest regions of $\mathscr{R}(f)$ are located before the obstacle (in blue, red and green (a), top). While these features are not important for the understanding of the structure of the turbulent flow, their projections cover most of the CSP (red, blue and green polygons in (b), left). In particular, due to the symmetry in the data, the blue and green regions nearly coincide in the CSP. Removing these regions from the projection results in a less cluttered CSP, better revealing the projections of the turbulent features of the flow ((b), right). The user can further inspect these features with localized CSPs (c). These visualizations are enabled by our new Reeb space computation algorithm, which computes this segmentation in a minute and a half, while previous techniques either take days to compute or hours to approximate the result.

**Abstract**— This paper presents an efficient algorithm for the computation of the Reeb space of an input bivariate piecewise linear scalar function $f$ defined on a tetrahedral mesh. By extending and generalizing algorithmic concepts from the univariate case to the bivariate one, we report the first practical, output-sensitive algorithm for the exact computation of such a Reeb space. The algorithm starts by identifying the Jacobi set of $f$, the bivariate analogs of critical points in the univariate case. Next, the Reeb space is computed by segmenting the input mesh along the new notion of *Jacobi Fiber Surfaces*, the bivariate analog of critical contours in the univariate case. We additionally present a simplification heuristic that enables the progressive coarsening of the Reeb space. Our algorithm is simple to implement and most of its computations can be trivially parallelized. We report performance numbers demonstrating orders of magnitude speedups over previous approaches, enabling for the first time the tractable computation of bivariate Reeb spaces in practice. Moreover, unlike range-based quantization approaches (such as the Joint Contour Net), our algorithm is parameter-free. We demonstrate the utility of our approach by using the Reeb space as a semi-automatic segmentation tool for bivariate data. In particular, we introduce *continuous scatterplot peeling*, a technique which enables the reduction of the cluttering in the continuous scatterplot, by interactively selecting the features of the Reeb space to project. We provide a VTK-based C++ implementation of our algorithm that can be used for reproduction purposes or for the development of new Reeb space based visualization techniques.

**Index Terms**—Topological data analysis, multivariate data, data segmentation

## 1 Introduction

As scientific data-sets become more intricate and larger in size, advanced data analysis algorithms are needed for their efficient visualization and exploration. For scalar field visualization, topological analysis techniques have shown to be practical solutions in various contexts by enabling the concise and complete capture of the structure of the input data into high-level *topological abstractions* such as contour trees [8], Reeb graphs [35, 4, 44], or Morse-Smale complexes [20, 47, 11]. Such topological abstractions are fundamental data-structures that enable the development of advanced data analysis, exploration and visualization techniques, including for instance: small seed set extraction for fast isosurface traversal [45, 9], feature tracking [42], transfer function design for volume rendering [46], similarity estimation [43], or application-driven segmentation and analysis tasks [29, 22, 19, 21].

However, with the ongoing development of computational resources on the one hand and of sensing devices on the other, multivariate scalar data-sets become more and more common. Such data-sets represent functions that no longer map points of the domain to the real line as it is the case for univariate scalar data, but to Euclidean spaces of higher dimension, each dimension typically representing a variable under investigation (temperature, pressure, velocity magnitude, etc.). To enable the extension and the generalization of the topology based visualization techniques mentioned above to this new type of data, the core concepts and algorithms of topological data analysis first have to be generalized to ranges of higher dimension than one, from the univariate to the multivariate case. This paper addresses this problem by presenting an efficient algorithm for the computation of the Reeb space [15], a generalization of the notion of Reeb graph [37], from the univariate to the bivariate case. Although the bivariate case is a very specific case of multivariate data, we believe it constitutes an appeal-


- *Julien Tierny is with Sorbonne Universites, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, France. E-mail: julien.tierny@lip6.fr.*
- *Hamish Carr is with the University of Leeds. E-mail: h.carr@leeds.ac.uk.*


*Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx*

ing first step, as the variables of multivariate data are often co-analyzed and co-visualized in pairs in the form of two-dimensional scatterplots.

The notion of Reeb space and its properties were first investigated by Edelsbrunner et al. [15], who introduce a dimension-independent algorithm that admits in the bivariate case a quadratic complexity $O(n_e \times n_T)$ (where $n_e$ and $n_T$ stand for the number of edges and of tetrahedra in the domain respectively). In comparison, our algorithm has also a quadratic worst-case complexity but is output-sensitive, which results in orders of magnitude speedups in practice. Carr and Duke introduce a range-based quantization approach to approximate the Reeb space with the notion of Joint Contour Net [6]. In contrast, our algorithm is exact and parameter-free.

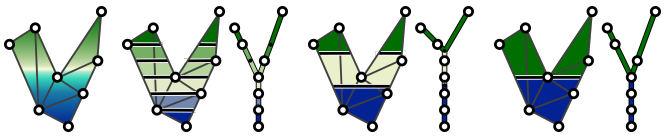
In this paper, we extend and generalize algorithmic concepts of the univariate case to the bivariate one, which results in the definition of a simple, efficient and practical algorithm for Reeb space computation on bivariate data. In doing so, we present an extensive algorithmic analogy between the univariate and the bivariate cases, which identifies their similarities and highlights their core differences. Our algorithm is simple to implement and most of its computations can be trivially parallelized. Extensive experiments report orders of magnitude speedups in comparison to previous approaches. Despite the simplicity of our algorithm, we believe it constitutes an important practical result as it brings computation times from hours or days to a few dozens of seconds, hence making bivariate Reeb space computation tractable for the first time. We additionally present a simplification heuristic that enables the progressive coarsening of the Reeb space. To demonstrate the versatility and the utility of our approach, we introduce *continuous scatterplot peeling*, a technique exploiting the Reeb space to reduce cluttering in continuous scatterplots, by separating regions in the domain that overlap in the range. We also provide a lightweight VTK-based C++ implementation of our algorithm that can be used for reproduction purposes or for the development of new Reeb-space based visualization techniques.

### 1.1 Related work

The related work can be classified into two main categories: topological data analysis of univariate scalar fields (which we partially extend in the current paper to bivariate fields) and visualization techniques for bivariate and multivariate data.

Topological data analysis of univariate scalar fields Over the last two decades, topological data analysis has played a key role for the development of advanced analysis, exploration and visualization algorithms for univariate scalar fields. Topological techniques focus on the efficient computation of high-level *topological abstractions* that capture in a concise and complete manner the structure of the data. Such abstractions include the Reeb graph [37, 35, 4, 44] (and its loop free variant the contour tree [8]) or the Morse-Smale complex [20, 47, 11]. These structural abstractions constitute fundamental data-structures that facilitate, accelerate or automate various geometric algorithms in visualization, including for instance small seed set extraction for fast isosurface traversal [45, 9], feature tracking [42], transfer function design [46], similarity estimation [43], and application-driven segmentation and analysis tasks [29, 22, 19, 21]. We refer the reader to survey articles [4, 11] for an extended discussion on the applications of topological data analysis to visualization. Among the existing topological abstractions, the Reeb graph is a popular data-structure that represents the evolution of the connectivity of the level sets of the input data, by contracting connected components of level sets to points. We review in the following some of the existing algorithms for its computation and relate them to our extension to the bivariate case.

**Vertex-based contouring:** The first combinatorial algorithm for Reeb graph computation on piecewise-linear 2-manifolds was introduced by Shinagawa and Kunii [40]. It constructs the Reeb graph by first segmenting the domain along the level sets of each vertex. This step is equivalent to considering the pre-image of all maximal range intervals where no vertex projects. Next, the arcs of the Reeb graph are constructed by connecting adjacent components in the domain (Fig. 2(a)). This results in a $O(n_v \times n_t)$ time complexity where $n_v$ and $n_t$ stand for the number of vertices and triangles of the input domain respectively. We call such an approach a *vertex-based contouring* strategy.



(a) [40], [15] (b) [3], [6] (c) [36], this paper

Fig. 2. Algorithmic analogy to the univariate case (left: input univariate scalar field). The algorithms by Edelsbrunner et al. [15], by Carr and Duke [6] and our approach are bivariate generalizations of the *vertex-based contouring* [40], the *quantized range contouring* [3, 23, 48] and the *critical point contouring* [36, 12] strategies, respectively.

**Quantized range contouring:** To accelerate the computation time in practice, approximate algorithms based on range quantization have been proposed [3, 23, 48]. These algorithms extract the connected components of the pre-images of regularly sampled intervals of the range, whose width is given by a user-defined range quantization parameter. The Reeb graph is then constructed by connecting adjacent components (Fig. 2(b)), resulting in a $O(n_i \times n_t)$ time complexity (where $n_i$ stands for the number of range intervals). This strategy, which we call *quantized range contouring*, speeds up Reeb graph computation since fewer level sets are extracted ($n_i$ is typically orders of magnitude smaller than $n_v$) at the expense of an inaccurate extraction, possibly discarding arcs of the Reeb graph whose image is shorter than the user-defined range quantization threshold.

**Critical point contouring:** To combine the advantages of the two previous strategies (accuracy and speed), Patane et al. [36] and Doraiswamy and Natarajan [12] independently introduced a similar strategy, that we call *critical point contouring*. It is based on a central idea in Morse theory [31] which states that the topology of the level sets only changes in the vicinity of the critical points of the function (points where its gradient vanishes). Thus, these algorithms compute the Reeb graph by computing, for each critical point, the connected component of level set passing through it. Next, the arcs of the Reeb graphs are simply obtained by considering the partitions of the induced domain segmentation (Fig. 2(c)). This strategy also results in a quadratic time complexity $O(n_c \times n_t)$ for 2D domains and $O(n_c \times n_T)$ for 3D domains, where $n_c$ stands for the number of critical points. However, for most practical data-sets, $n_c$ is typically one to two orders of magnitude smaller than $n_v$, making this strategy much more efficient than vertex-based contouring, while still guaranteeing an exact output (unlike quantized range contouring). Other Reeb graph computation techniques include algorithms with improved practical performance [35, 44] or optimal time complexity [10, 34].

Multivariate data analysis and visualization The analysis and visualization of multivariate data, functions which map each point of the domain to a Euclidean space of dimension higher than one, has gained a lot of attention lately, partly due to the prominence of this type of data in numerical simulations. Several automatic and interactive techniques have been developed to investigate the correlation between each represented variable (i.e. each dimension of the range), including scatterplot based visualizations [1, 30], gradient-based correlation analysis [39, 33] or interactive brushing techniques [25, 7]. A detailed review of all the techniques addressing multivariate data is out of the scope of this paper and we refer the reader to survey articles [18, 27] for comprehensive descriptions.

A first topological insight in the correlation of the components of a multivariate function is given by the notion of Jacobi sets [13]. Hüttenberger et al. [24] use the notion of Pareto optimality to extract extremal structures in multivariate data. In particular, an extremal point is identified if it locally dominates its neighbors for each component of the multivariate function. Although these structures share similarities with the Jacobi sets, it is not clear how they relate to the topology of *fibers* (multivariate analogs of level sets). The notion of Reeb space, the generalization of the concept of Reeb graph [37] to the multivariate setting, was first investigated by Edelsbrunner et

al. [15], who describe a dimension-independent algorithm which can be specialized to the bivariate case with a quadratic time complexity ($O(n_e \times n_T)$, where $n_e$ stands for the number of edges in the domain). This algorithm constructs the Reeb space by gluing domain-incident connected components of pre-images of *chambers*, which can be interpreted in the bivariate case as maximal convex polygons in the range, delimited by the image of the set of all edges of the domain. This strategy can be seen as the bivariate generalization of the vertex-based contouring approach in the univariate case (Fig. 2(a)). To the best of our knowledge, no implementation of this algorithm has ever been documented so far. In this paper, we specialize and improve this algorithm for the bivariate case. Carr and Duke [6] introduced an algorithm that approximates the Reeb space by considering the connected components of the pre-images of a pixel-based quantization of the range. This corresponds to the bivariate generalization of the quantized range contouring strategy (Fig. 2(b)). In particular, this algorithm approximates exact chambers [15] with pixels, whose size is defined by a user-defined range quantization threshold. Thus, as with the quantized range contouring strategy in the univariate setting, this algorithm may miss features in the Reeb space which project to sub-pixel regions in the range. We refer the reader to [32] for a detailed theoretical discussion regarding the convergence of this quantization approach. In this paper, we present an algorithm that generalizes the critical point contouring strategy (Fig. 2(c)), which yields, similarly to the univariate case, exact, parameter-free, output-sensitive and fast computations.

### 1.2 Contributions

This paper makes the following new contributions:

1. **Approach:** We generalize algorithmic concepts from the univariate case to the bivariate one and provide a comprehensive analogy between the two settings, identifying their similarities and highlighting their core differences. We believe this yields a simple and intuitive description of bivariate Reeb spaces, which may facilitate their adoption in visualization applications.

2. **Algorithm:** We present an algorithm for bivariate Reeb space computation that is simple, parameter-free, output-sensitive and orders of magnitude faster than previous algorithms in practice. Most of its computations can be trivially parallelized for improved performances. We present a heuristical simplification strategy that enables the interactive coarsening of the Reeb space.

3. **Application:** We present an application of the bivariate Reeb space to *Continuous Scatterplot Peeling*, a technique that enables the interactive clutter reduction in continuous scatterplots by selecting the features of the Reeb space to project.

4. **Implementation:** We provide a lightweight VTK-based C++ implementation of our algorithm that can be used for reproduction purposes or for the development of new visualization techniques based on Reeb spaces.

## 2 Preliminaries

This section describes our formal setting (Sec. 2.1) and presents preliminary results (Sec. 2.2) that will guide the definition of our algorithm (next section). Introductions to fiber topology are also available [38, 5]. To provide an intuitive description of the Reeb space, we will develop in the following a running analogy between the univariate and bivariate cases, discussing their similarities and differences.

### 2.1 Background

Let $f : \mathscr{M} \to \mathbb{R}^2$ be a bivariate piecewise-linear (PL) scalar field defined on a PL 3-manifold $\mathscr{M}$ embedded in $\mathbb{R}^3$. It has values at the vertices of $\mathscr{M}$ and is linearly interpolated on the simplices of higher dimension. In the following, $\mathscr{M}$ and $\mathbb{R}^2$ will be called the *domain* and the *range* respectively. $f$ can be decomposed into two independent, univariate PL scalar fields $u : \mathscr{M} \to \mathbb{R}$ and $v : \mathscr{M} \to \mathbb{R}$:

$$\forall p \in \mathscr{M}, \;\; f(p) = \{u(p), v(p)\} \tag{1}$$

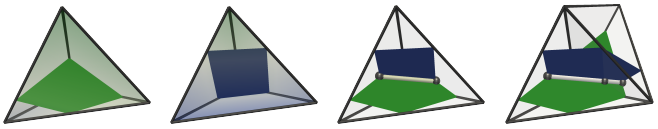


Fig. 3. Given a point $q = \{q_u, q_v\} \in \mathbb{R}^2$, its *fiber* $f^{-1}(q)$ can be obtained by considering the intersection of the level sets of its coordinates $u^{-1}(q_u)$ (a) and $v^{-1}(q_v)$ (b). $f^{-1}(q)$ is shown with a white cylinder in (c) and its restriction to the triangles of $\mathscr{M}$ are shown with black spheres (c,d).

**Analogy:** As it is often the case in the univariate case, we will assume that $f$ is *generic* in order to easily handle degeneracies. This implies that the restriction of $f$ to the vertices of $\mathscr{M}$ is injective (i.e. no two vertices in $\mathscr{M}$ are equally valued by $u$ and by $v$). In practice, this guarantees that no edge of $\mathscr{M}$ collapses to a point in $\mathbb{R}^2$. However, in contrast to the univariate case, additional requirements are needed. In particular, the following, stronger condition has to be satisfied as well:

**Definition 1 (Non-collinearity condition)** *For any pair of edges $e_1$ and $e_2$ of $\mathscr{M}$, if the images $f(e_1)$ and $f(e_2)$ are not collinear, $f$ is said to satisfy the* non-collinearity condition.

Such a condition guarantees for instance that no triangle and no tetrahedron of $\mathscr{M}$ map to a line segment in $\mathbb{R}^2$. We will detail other useful guarantees in the following that derive from this condition. The non-collinearity and genericity conditions can easily be satisfied in practice through a two-dimensional instance of simulation of simplicity [17]. Such a process can be achieved symbolically, by re-implementing the necessary predicates, or numerically. For ease of implementation, we opted for the numerical alternative and the scalar fields $u$ and $v$ are slightly perturbed with the following $\varepsilon$*-expansions* [17], for each vertex $v_p$ of $\mathscr{M}$, where $\mathscr{O}(v_p) \in \mathbb{N}$ is the offset position of $v_p$ in memory and $\varepsilon$ is an arbitrarily small constant:

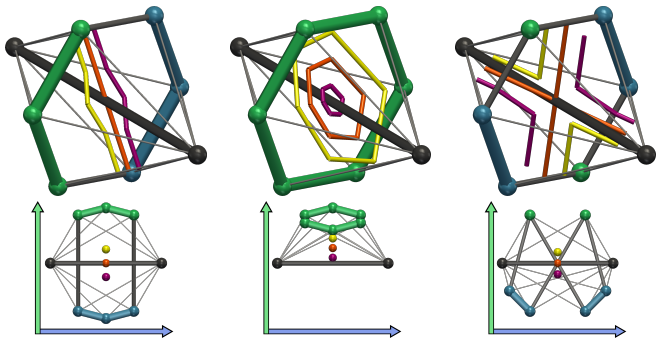
$$\begin{aligned} u(v_p) &\leftarrow u(v_p) + \varepsilon\mathscr{O}(v_p) \\ v(v_p) &\leftarrow v(v_p) + \varepsilon\mathscr{O}(v_p) \times \varepsilon\mathscr{O}(v_p) \end{aligned} \tag{2}$$

Since $\mathscr{O}(v_p)$ is by construction injective, this $\varepsilon$-expansion introduces for each vertex a unique perturbation, given by the vector $\{\varepsilon\mathscr{O}(v_p), \varepsilon\mathscr{O}(v_p) \times \varepsilon\mathscr{O}(v_p)\}$. Moreover, since the slope of this perturbation vector is also unique for each vertex ($\varepsilon\mathscr{O}(v_p)$), this guarantees that no vertex pair can be perturbed in a collinear manner, which is sufficient to enforce the non-collinearity condition.

**Definition 2 (Fiber)** *Let $q = \{q_u, q_v\} \in \mathbb{R}^2$ be a point in the range. The* fiber *of $q$, noted $f^{-1}(q)$, is the pre-image of $q$ onto $\mathscr{M}$ through $f$:*

$$\begin{aligned} f^{-1}(q) &= \{p \in \mathscr{M} \mid f(p) = q\} \\ &= \{p \in \mathscr{M} \mid u(p) = q_u\} \cap \{p \in \mathscr{M} \mid v(p) = q_v\} \end{aligned} \tag{3}$$

Fibers are multivariate analogs of level-sets. They can be made of several connected components, called *fiber components*. By construction (Eq. 3), $f^{-1}(q)$ is given by the intersection of the level sets $u^{-1}(q_u)$ and $v^{-1}(q_v)$. Since both $u$ and $v$ are assumed to be generic, the restrictions of $u^{-1}(q_u)$ and $v^{-1}(q_v)$ to the interior of a tetrahedron of $\mathscr{M}$ are either empty or planar polygons (Figs. 3(a) and 3(b)). Moreover, since $f$ is assumed to satisfy the non-collinearity condition (Def. 1), these restrictions cannot be coplanar. It follows that the restriction of a fiber to the interior of a tetrahedron of $\mathscr{M}$ is either empty or a single line segment (Fig. 3(c)). Then the restriction of a fiber to the interior of a triangle of $\mathscr{M}$ is either empty or a point. In the latter case, the existence of such a point implies that the restrictions of the corresponding fiber to the cofaces of the triangle (its at most two adjacent tetrahedra) are non empty due to the continuity of the PL interpolant. Hence such restrictions are line segments (Fig. 3(d)). In other words, fibers cannot disconnect or disappear along the interior of the triangles of $\mathscr{M}$. Then, the only simplices of $\mathscr{M}$ across which fibers can change their topology (appear, connect, disconnect or disappear) are its vertices and edges. In particular, such configurations belong to the *Jacobi set* of $f$ [13].



(a) (b) (c)

Fig. 4. Jacobi edge classification. The input edge $e$, $Lk^-(e)$ and $Lk^+(e)$ are represented with black, blue and green cylinders respectively in the domain (top) and the range (bottom). Fibers crossing the edge $e$ are represented in yellow, orange and purple in each column. (a) Fibers do not change their topology when crossing a regular edge. (b) Fibers shrink to a point and disappear when crossing an extremum edge. (c) Fibers merge (yellow to orange) and immediately split (orange to purple) when crossing a saddle edge.

**Analogy:** In the univariate case, the topology of the level sets of a PL scalar field can only change at certain vertices, called critical points [2]. These can be extracted by considering the lower and upper links of each vertex. A similar strategy can be derived in the bivariate case, by considering the lower and upper links of each edge however.

The *star* $St(e)$ of a simplex $e$ is the set of simplices $\sigma$ that contain $e$ as a face. The *link* $Lk(e)$ of a simplex $e$ is the set of faces of the simplices of the star $St(e)$, which do not intersect $e$.

Let $d_e : Lk(e) \rightarrow \mathbb{R}$ be the following univariate PL scalar field:

$$d_e(v) = \langle \overrightarrow{f(v)f(v')}, \overrightarrow{n_{f(e)}} \rangle \tag{4}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product, $\overrightarrow{n_{f(e)}}$ the vector normal to the image $f(e)$ of the edge $e$, and $v'$ one of the vertices of $e$. This scalar field can be interpreted as a range-based signed distance field from the vertices of $Lk(e)$ to $f(e)$. Both $\overrightarrow{f(v)f(v')}$ and $\overrightarrow{n_{f(e)}}$ are guaranteed to be different from the null vector since no edge can collapse to a point through $f$ (genericity condition).

The *lower link* $Lk^-(e)$ of an edge $e$ is the subset of $Lk(e)$ containing only vertices with negative $d_e$ values: $Lk^-(e) = \{\sigma \in Lk(e) \mid \forall v \in \sigma : d_e(v) < 0\}$. The *upper link* $Lk^+(e)$ is defined symmetrically: $Lk^+(e) = \{\sigma \in Lk(e) \mid \forall v \in \sigma : d_e(v) > 0\}$. Given the non-collinearity condition (Def. 1), there is no $v \in Lk(e)$ such that $d_e(v) = 0$. Hence, $Lk^-(e)$ and $Lk^+(e)$ cannot be both empty and a vertex $v \in Lk(e)$ can always be classified as either belonging to $Lk^-(e)$ or $Lk^+(e)$. This definition of lower and upper links generalizes the univariate one.

**Definition 3 (Jacobi edge)** *An edge $e$ of $\mathcal{M}$ is regular if and only if both $Lk^-(e)$ and $Lk^+(e)$ are simply connected. Otherwise, $e$ is a* Jacobi edge *of $f$.*

If $Lk^-(e)$ or $Lk^+(e)$ is empty, $e$ is called an *extremum edge*. Otherwise, if $e$ is neither regular nor an extremum edge, it is called a *saddle edge*. As shown in Fig. 4(b), the number of fiber components in $St(e)$ goes from 0 to 1 (or 1 to 0) when crossing an extremum edge in the range. Thus, fiber components either appear or disappear when crossing an extremum edge. In contrast, this number does not evolve when crossing a regular edge (Fig. 4(a)). Finally, similar to saddles in 2D for the univariate case, the number of fiber components suddenly decreases to one at a saddle edge and increases again after crossing it (Fig. 4(c)). In conclusion, the topology of fibers only changes when crossing Jacobi edges: components appear or disappear on extremum edges and components merge or disconnect on saddle edges. The set of all Jacobi edges is called the *Jacobi set* [13].

**Analogy:** Given the generalization to the bivariate case of the notions of lower and upper links, note that the definition of a Jacobi edge is a direct extension of that of a critical point in the univariate case [14]. Therefore, we will consider that Jacobi edges are analogs of critical points in the univariate case. However, Jacobi edges lose many of the nice properties of their univariate analogs. In particular, they are not *isolated* (two Jacobi edges can share a vertex) and their images are not necessarily disjoint (the images of two Jacobi edges can intersect). As discussed later on, this complicates the design of topological algorithms in the bivariate case.

As illustrated in Fig. 4, the topology of the fibers can only change in the vicinity of Jacobi edges. The notion of the Reeb space enables a more global understanding of these topological events [15]:

**Definition 4 (Reeb space)** *Given a point $p \in \mathcal{M}$ and its image $f(p) = q \in \mathbb{R}^2$, let $f^{-1}(q)_p$ be the connected component of the fiber $f^{-1}(q)$ which contains $p$. The Reeb space $\mathcal{R}(f)$ is a two-dimensional complex defined as the quotient space $\mathcal{R}(f) = \mathcal{M} / \sim$ by the equivalence relation $p_1 \sim p_2$, which holds if:*

$$\begin{cases} f(p_1) = f(p_2) = q \\ p_2 \in f^{-1}(q)_{p_1} \end{cases}$$

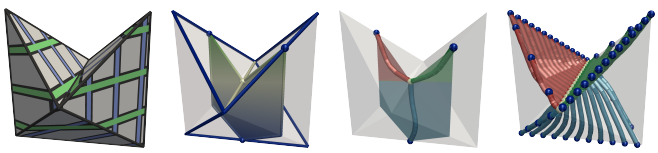
**Analogy:** The notion of Reeb space is obtained with a direct extension to the bivariate case of the Reeb graph definition [37]. Intuitively, fiber components are retracted to points and such points will be adjacent in $\mathcal{R}(f)$ if the corresponding fiber components are also adjacent in $\mathcal{M}$. Note that, unlike the univariate case, the Reeb space is no longer a simplicial complex. In particular, it is locally homeomorphic to a subset of $\mathbb{R}^2$ but it is not manifold in the vicinity of the equivalence classes of Jacobi edges. Given an isovalue $q_u$, if one restricts Def. 4 to the subset $u^{-1}(q_u) \subset \mathcal{M}$, note that the corresponding Reeb space becomes equal by definition to the Reeb graph of the restriction to the subset $u^{-1}(q_u)$ of the function $v : \mathcal{M} \rightarrow \mathbb{R}$. In other words, cutting a bivariate Reeb space along an isovalue of one function yields the Reeb graph of the other function on the corresponding level-set (Fig. 5).

### 2.2 Reeb space characterization and computation

In this subsection, we introduce novel notions that further characterize the Reeb space and which are central to our algorithm. As discussed above, the Reeb space is made of two-dimensional cells that locally connect in a non-manifold way. Each of these cells represents a region of the domain where fibers are made of a single connected component, and which can be further characterized as follows.

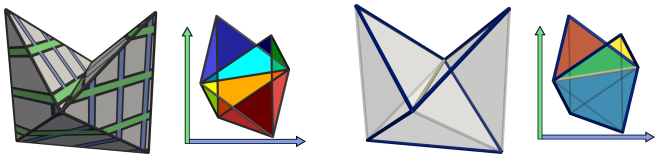
**Definition 5 (3-sheet)** *Let $S$ be a connected, three-dimensional subset of $\mathcal{M}$. $S$ is said to satisfy the* retraction condition *if for each point $p \in S$ with $f(p) = q \in \mathbb{R}^2$, the restriction of $f^{-1}(q)$ to $S$ is equal to $f^{-1}(q)_p$. If so, $S$ is called a* 3-sheet.

Intuitively, a 3-sheet can be interpreted as a connected region of $\mathcal{M}$ which, given the equivalence relation that defines the Reeb space (Def. 4), completely captures the equivalence class of each of its points. Note that the union of adjacent 3-sheets in $\mathcal{M}$ forms a valid 3-sheet as well, as long as it satisfies the retraction condition.



(a) (b) (c) (d)

Fig. 5. Relation between the Reeb space and the Reeb graph on a simple bivariate example (the X and Y 3D coordinates serve as input bivariate function). (a) Input bivariate function (left: green and blue perpendicular lines denote $u$ and $v$ level-sets). (b) Given an isovalue $q_u$, the restriction to $u^{-1}(q_u)$ of $v : \mathcal{M} \rightarrow \mathbb{R}$ is shown by the colored surface (from blue to green). The critical points [2] of this restriction (blue spheres: extrema, white sphere: saddle) are located on the Jacobi set of $f$ (extremum and saddle edges are shown in blue and white respectively). (c) Reeb graph of the restriction. (d) The Reeb space can be interpreted as a continuous stacking of such Reeb graphs, as $q_u$ continuously evolves.



(a) (b)

Fig. 6. Generalization of the critical point contouring strategy (toy example from Fig. 5). (a) Input bivariate function in the domain and the range (right, each colored polygon is a *chamber*). (b) The image (right) of the Jacobi set of $f$ (left: extremum and saddle edges are shown in blue and white respectively) also segments $f(\mathscr{M}) \subset \mathbb{R}^2$ into colored polygons within which the topology of the fibers is guaranteed not to change.

The vertex-based contouring strategy [40, 15] can be specialized to the bivariate setting by considering the pre-image of *chambers* and gluing the connected components of such pre-images if they are incident in $\mathscr{M}$. Such chambers would then be defined as maximal convex polygons in the range where no edge of $\mathscr{M}$ projects (Fig. 6(a)). Since the topology of the fibers can only change across the edges of $\mathscr{M}$, the connected components of the pre-images of chambers are then valid 3-sheets. To construct such 3-sheets, one first needs to construct the boundaries in $\mathscr{M}$ of the chamber pre-images. These are given by considering the union of the pre-image of the image of all edges of $\mathscr{M}$.

**Definition 6 (2-sheet)** *Let $e$ be an edge of $\mathscr{M}$. Its image $f(e)$ is a line segment in $\mathbb{R}^2$ and the pre-image of $f(e)$ is called the* 2-sheet *of $e$.*

Since $f$ is assumed to satisfy both the genericity and the non-collinearity condition, $f(e)$ is indeed a line segment and the corresponding 2-sheet is indeed a 2-dimensional object in $\mathscr{M}$. The computation of such 2-sheets can be achieved with the notion of *fiber surface* [7, 28], which is defined as the pre-image of a curve through a bivariate function. This computation requires $O(n_T)$ steps, for each of the $n_e$ edges of $\mathscr{M}$, yielding an overall time complexity of $O(n_e \times n_T)$ steps (where $n_T$ is the number of tetrahedra in $\mathscr{M}$). The above strategy is a bivariate specialization of the vertex-based contouring [40, 15].

However, the Reeb space can be computed much more efficiently, by generalizing the critical point contouring strategy [36, 12], which we do with our approach. In particular, as described in the previous subsection, the topology of the fibers can only change when crossing a certain sub-set of the edges of $\mathscr{M}$: the Jacobi set. Therefore, to construct the Reeb space, one can consider larger 3-sheets, being the connected components of the pre-images of multiple adjacent chambers. In particular, one can consider the pre-image of maximal convex polygons in the range where no *Jacobi edge* projects (Fig. 6(b)). To construct such extended 3-sheets, one needs to construct their boundaries in $\mathscr{M}$, which is given by the notion of Jacobi fiber surface:

**Definition 7 (Jacobi fiber surface)** *The 2-sheet of an extremum edge and of a saddle edge are called* extremum *and* saddle Jacobi fiber surfaces *respectively.*

The restriction of the Jacobi fiber surface of an extremum edge $e$ to its star $St(e)$ coincides with $e$ (Fig. 4(b)). Otherwise, like its constituting fibers (in orange, Fig. 4), it is manifold for regular edges (Fig. 4(a)) and non-manifold for saddle edges, precisely along $e$ (Fig. 4(c)).

As discussed in the experiment section, the number of Jacobi edges (noted $n_j$) is orders of magnitude smaller than $n_e$ for practical datasets, making this strategy appealing for performance improvement, resulting in a time complexity of $O(n_j \times n_T)$ steps.

In practice, this strategy can be further improved. In particular, for a given saddle edge $s$, only the connected component of the Jacobi fiber surface that passes through it needs to be considered. Indeed, as illustrated in Fig. 4(c), when crossing $f(s)$ in the range, only the fiber components in the vicinity of $s$ will disconnect or re-connect, precisely at $s$. In contrast, fiber components which do not touch $s$ in $\mathscr{M}$ when crossing $f(s)$ in the range will then have their connectivity unchanged. This means that such components belong to a *unique* 3-sheet before *and* after crossing $f(s)$.

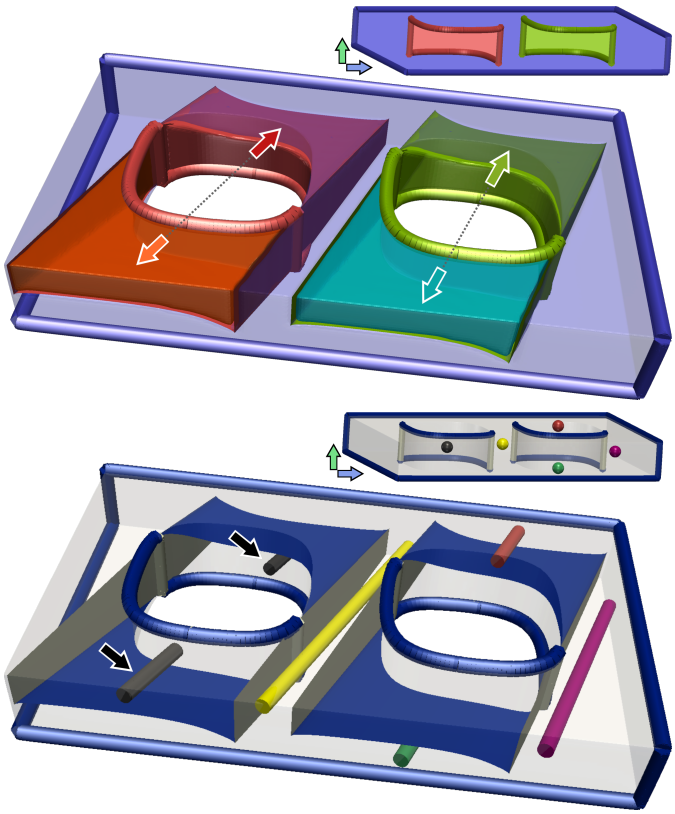


Fig. 7. Reeb space of a bivariate example (top, inset: $f(\mathscr{M}) \subset \mathbb{R}^2$, the X and Y 3D coordinates serve as input bivariate function). In this dataset, each topological handle yields a pair of 3-sheets (highlighted by colored, aligned arrow pairs, top) as illustrated by the black fiber which is made of two connected components (black arrows, bottom). Extrema Jacobi fiber surfaces (blue, bottom) contribute to the separation of these regions from the rest of the object, where fibers are made of a single connected component (other colors, bottom).

The decomposition of $\mathscr{M}$ into 3-sheets along Jacobi fiber surfaces then produces a direct representation of the Reeb space $\mathscr{R}(f)$, where each 3-sheet in $\mathscr{M}$ denotes a two-dimensional cell in $\mathscr{R}(f)$, and where adjacent 3-sheets in $\mathscr{M}$ denote adjacent two-dimensional cells in $\mathscr{R}(f)$. At this point, note that the union of two adjacent 3-sheets can still respect the retraction condition and therefore constitute a valid, larger 3-sheet if *no* saddle Jacobi fiber surface separates them. Indeed, such a configuration implies that a fiber component can travel from one 3-sheet to the other, without disconnecting or reconnecting. To complete our characterization of the Reeb space, we introduce the following two notions:

**Definition 8 (1-sheet)** *A connected component of the Jacobi set in $\mathscr{M}$ is called a 1-sheet.*

Note that in general, 1-sheets are not necessarily 1-manifolds. Thus, for completeness, we finally introduce the following notion:

**Definition 9 (0-sheet)** *A vertex of the Jacobi set of $f$ which is not exactly adjacent to two Jacobi edges is called a* 0-sheet.

**Analogy:** Jacobi edges and Jacobi fiber surfaces are bivariate analogs of the notions of critical points and critical contours in the univariate case respectively. However, unlike the univariate case where only saddle critical contours need to be considered [36, 12], the Jacobi fiber surfaces of extrema edges also need to be considered in the bivariate case. Fig. 7 illustrates this for a bivariate function defined on a bitorus. In particular, each topological handle yields a pair of non-mergeable 3-sheets: an orange-red pair and a cyan-green pair (highlighted by aligned arrow pairs of matching color, top). As illustrated by the black fiber (bottom), fibers in these areas are made of two connected components, while they are made of only one component anywhere else

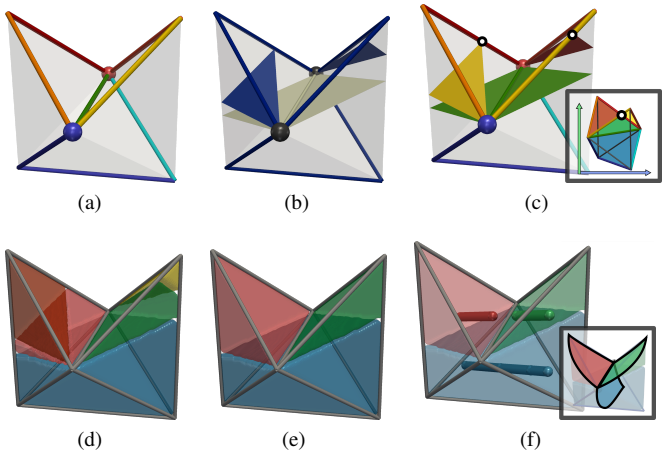


Fig. 8. Algorithm overview on the toy example from Fig. 5. Our algorithm computes the Reeb space by constructing sheets in order of increasing dimension. First, 1-sheets are constructed as connected components of the Jacobi set (a). Second, the corresponding 2-sheets are constructed. Extrema and saddle Jacobi fiber surfaces are shown in blue and white respectively (b). Here only the red and yellow extrema edges yields non-empty 2-sheets (c). Third, the 3-sheets are extracted as the connected components delimited by the 2-sheets (d). Finally, an expansion process merges adjacent 3-sheets that are not separated by any saddle Jacobi fiber surface (e). Within each 3-sheet of the output Reeb space (f), each fiber is made of a single connected component, as shown with cylinders of matching color (inset: schematic view of the Reeb space).

(yellow, purple, red and green fibers). In that configuration, both extremum and saddle Jacobi fiber surfaces are indeed needed to properly delimit the 3-sheets on the handles from the rest of the object, which constitutes a distinct 3-sheet (in blue, top) where the yellow, purple, red and green fibers belong (bottom). Finally, since the images of multiple Jacobi edges are not necessarily disjoint, note that multiple Jacobi fiber surfaces can intersect in $\mathscr{M}$.

## 3 Algorithm

Our algorithm naturally and concisely follows from the properties discussed in the previous section and an overview of it is given in Fig. 8. This section additionally describes the parallelization of our approach and presents a heuristic approach to Reeb space simplification.

As discussed in our characterization of the Reeb space (Sec. 2.2), the topology of the fibers can only change when crossing the Jacobi edges of $f$. Therefore, as illustrated in Fig. 8, our algorithm constructs the Reeb space by extracting the connected components of the pre-image of maximal convex polygons in the range where no Jacobi edge projects (Fig. 6(b)). The computation of such components (called 3-sheets in the previous section) first requires the extraction of their boundaries (2-sheets), which itself requires the computation of the Jacobi set (1-sheets). Thus, our algorithm computes the Reeb space by constructing such sheets in order of increasing dimension, as detailed in the following subsections.

### 3.1 1-sheet computation

1-sheets are constructed as connected components of the Jacobi set of $f$. Jacobi edges are first extracted according to the classification presented in Fig. 4. Such a classification is a local operation, that requires for each edge $e \in \mathscr{M}$ the construction of its link $Lk(e)$ and the evaluation of its range distance field $d_e$ (Eq. 4). Once the Jacobi edges have been identified, the 0-sheets are extracted as the vertices of the Jacobi set which are not adjacent to exactly two Jacobi edges. Finally, each connected component of the Jacobi set is extracted with a standard breadth-first search pass.

### 3.2 2-sheet computation

2-sheets, or Jacobi fiber surfaces, are constructed, for each Jacobi edge $e$, as the pre-image of $f(e)$. Such a pre-image can be obtained with

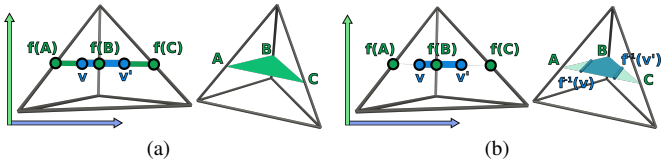



Fig. 9. Fiber surface computation within a tetrahedron given a line segment $\{v, v'\} \in \mathbb{R}^2$ (the orientation axes on the left denote a range view, as opposed to a domain view). First (a), the level set $d_e^{-1}(0)$ is computed (green triangle, right). Second (b), this level set is clipped (blue polygon, right) by linear interpolation to the points that strictly projects to $\{v, v'\}$.

the algorithm described in [28] which presents an approach for the computation of pre-images of curves through bivariate functions. For completeness, we sketch its main steps here. Given a line segment in $\mathbb{R}^2$ (in our current case, $f(e)$), the algorithm visits each tetrahedron of $T \in \mathscr{M}$ and applies the following procedure. The level set $d_e^{-1}(0)$ is first computed (Eq. 4). This corresponds to the set of points of $T$ that project in $\mathbb{R}^2$ to the line that carries $f(e)$ (Fig. 9(a)). Second, this surface is further clipped to select only the points of $T$ that strictly project within $f(e)$. To achieve this, a linear parameterization $t : d_e^{-1}(0) \to \mathbb{R}$ is first evaluated. This parameterization is set to 0 and 1 at the two extremities of $f(e)$ ($v$ and $v'$ in Fig. 9(b)) and its values for the projected vertices of $d_e^{-1}(0)$ ($f(A)$, $f(B)$ and $f(C)$ in Fig. 9(b)) are given by linear interpolation in the range. Finally, the clipped fiber surface is obtained by considering the pre-image of the interval $[0,1]$ onto $d_e^{-1}(0)$ through $t$. We refer the reader to [28] for further details.

As discussed in the previous section, for a given saddle edge $s$, only the connected component of its Jacobi fiber surface that passes through it needs to be considered for Reeb Space computation. Therefore, for such a Jacobi edge $s$, we construct its Jacobi fiber surface with the above algorithm by visiting $\mathscr{M}$ in a breadth-first search fashion, starting in $s$ and propagating to adjacent tetrahedra only if the current tetrahedron $T$ is indeed intersected by the fiber surface. This strategy effectively builds the connected component of Jacobi fiber surface that passes through $s$, while avoiding an exhaustive visit of all the tetrahedra of $\mathscr{M}$. This improvement cannot be applied to the extremum Jacobi fiber surfaces however since fibers disappear when crossing an extremum edge (Fig. 4). Then, as a consequence, given an extremum edge $e$, the restriction of the pre-image of $f(e)$ to $St(e)$ is only $e$ itself.

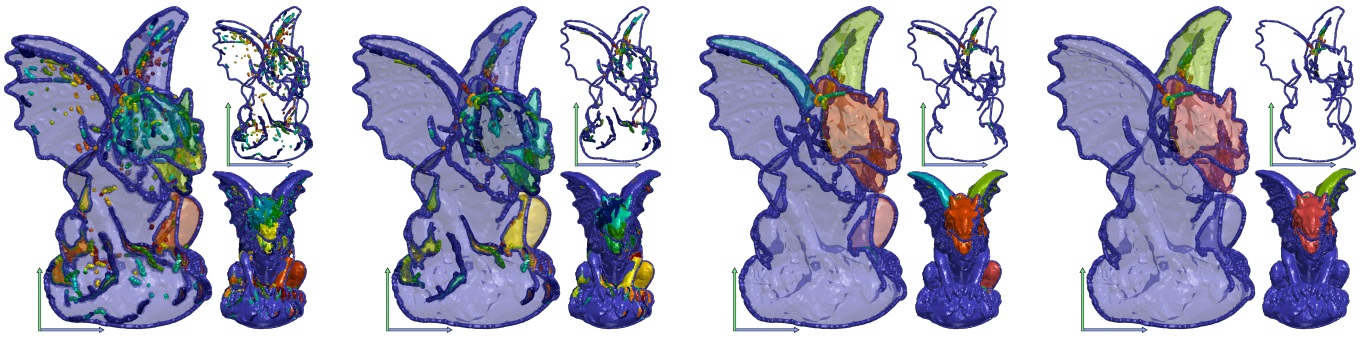
### 3.3 3-sheet computation

Once the 2-sheets have been extracted in $\mathscr{M}$, the 3-sheets can be constructed as connected components of $\mathscr{M}$ delimited by the Jacobi fiber surfaces. This is obtained through a breadth-first search traversal of the tetrahedra of $\mathscr{M}$, which stops when hitting a 2-sheet. In order to capture the geometry of the boundaries of the 3-sheets accurately, the tetrahedra crossed by Jacobi fiber surfaces should be re-meshed along them, which can be achieved through constrained triangulation [41].

As described in the previous section, at this point, the union of two adjacent 3-sheets can form itself a valid 3-sheet if no saddle Jacobi fiber surface separates them. Therefore, we merge adjacent 3-sheets that satisfy this criteria in a last step, as illustrated in Fig. 8(e).

### 3.4 Parallelism

Most of the steps described so far are local operations which can be trivially parallelized as follows. First, the classification of the edges of $\mathscr{M}$ as Jacobi or regular can be achieved in parallel (i.e. the $n_e$ edges of $\mathscr{M}$ are approximately equally distributed among the $n$ threads). The subsequent breadth-first search that identifies the 1-sheets is intrinsically sequential. However, as illustrated in the experimental section, the computation time of this last step is negligible in practice with regard to that of the Jacobi edge classification. Next, the computation of the Jacobi fiber surfaces is also achieved in parallel on a per Jacobi edge basis (i.e. the $n_j$ Jacobi edges of $f$ are approximately equally distributed among the $n$ threads). Again, the subsequent breadth-first search that constructs the 3-sheets is intrinsically sequential. However,



(a) (b) (c) (d)

Fig. 10. Simplification sequence for the Reeb space of a silhouette example. Each orientation axis denotes a range view, as opposed to a domain view (bottom right). The X and Y 3D coordinates serve as input bivariate function. The 1-sheets get progressively simplified (top right), yielding simpler and simpler silhouettes. Along the simplification (from left to right), the Reeb space contains 6152, 83, 5 and 3 3-sheets respectively.

Table 1. Running time of the different steps of the algorithm (in seconds). $n_T$ and $n_j$ are the numbers of tetrahedra in $\mathscr{M}$ and Jacobi edges of $f$. The columns $JE$, $1S$, $2S$, $3S$ and $P$ represent the computation time for the Jacobi edge classification, the extraction of the 1-, 2- and 3-sheets respectively and the parallel version of our algorithm (12 cores).

| Data-set | $n_T$ | $n_j$ | JE | 1S | 2S | 3S | Total | P | Speedup |
|---|---|---|---|---|---|---|---|---|---|
| Water dimer | 302,715 | 9,742 | 0.6 | 0.0 | 164.7 | 18.5 | 183.8 | **27.4** | 6.7 |
| Gargoyle | 1,098,602 | 21,689 | 3.0 | 0.0 | 754.3 | 5.7 | 762.9 | **71.7** | 10.6 |
| Mechanical | 1,482,764 | 11,486 | 3.3 | 0.0 | 345.7 | 4.8 | 353.9 | **32.6** | 10.9 |
| Bitorus | 2,555,904 | 976 | 5.7 | 0.0 | 65.4 | 0.7 | 71.8 | **6.9** | 10.4 |
| Vortex street | 5,060,475 | 8,654 | 10.1 | 0.0 | 904.4 | 15.3 | 929.8 | **92.0** | 10.1 |

Table 2. Running time comparison with the Joint Contour Net [6] (1 thread, 2 quantizations) and the chamber approach [15] (parallel). Running times are provided in seconds. The bold numbers indicate the speedups obtained by our algorithm.

| Data-set | $n_T$ | JCN-1 [6] | | JCN-100 [6] | | Chambers-P [15] | |
|---|---|---|---|---|---|---|---|
| Water dimer | 302,715 | 20.3 | **0.1** | 290.4 | **1.6** | 1,388.4 | **50.7** |
| Gargoyle | 1,098,602 | 226.5 | **0.3** | 2,011.6 | **2.6** | 8,654.6 | **120.7** |
| Mechanical | 1,482,764 | 490.0 | **1.4** | 5,649.2 | **16.0** | 8,116.6 | **249.0** |
| Bitorus | 2,555,904 | 521.5 | **7.3** | 4,903.7 | **68.3** | 23,361.2 | **3,385.7** |
| Vortex street | 5,060,475 | 587.7 | **0.6** | 15,660.0 | **16.8** | 103,862.0 | **1,128.9** |

the computation time of this last step is negligible in practice with regard to that of the Jacobi fiber surface computation (Sec. 4).

### 3.5 Simplification

While the theoretical extension of topological persistence [16] to the bivariate case is out of the scope of this paper, we observed in practice, as it is the case in the univariate setting, that the simplification of the Reeb space is often beneficial. We therefore introduce the following heuristic for the progressive coarsening of the Reeb space, which is inspired by the univariate case. We first evaluate the area of the projection of each 3-sheet in $\mathbb{R}^2$ by summing the area of the projection of each of its tetrahedra. This measure, noted $a(S)$, will serve as an importance score for each 3-sheet $S$. Next, given a user defined simplification score $\alpha$, the 3-sheet $S^*$ that minimizes $a$ is merged with the maximizer of $a$ among its adjacent neighbors if $a(S^*) < \alpha$. This merging process involves the concatenation of the sheets' list of tetrahedra and the update of the importance score. The merging procedure is iteratively repeated until $a(S^*)$ becomes greater than $\alpha$. Along this simplification sequence, we also prune 1-sheets (as well as the corresponding 2-sheets) if they contain saddle edges and if they become adjacent to only one 3-sheet (which implies that they no longer separate 3-sheets). The resulting sequence is illustrated in Fig. 10.

## 4 Results and Discussion

In this section, we present practical results obtained with a VTK-based C++ implementation of the algorithm (provided as additional material). Experiments were performed on a desktop computer with two Xeon CPUs (2.6 GHz, 6 cores each) with 64GB of RAM.

### 4.1 Time requirement

Given an input tetrahedral mesh $\mathscr{M}$ with $n_T$ tetrahedra and $n_e$ edges, our algorithm first starts by classifying edges as Jacobi or regular in $O(n_e)$ steps. Next, the 1-sheets are constructed with a breadth first search traversal of the $n_j$ Jacobi edges in $O(n_j)$ steps. Then, the Jacobi fiber surface of each Jacobi edge is computed, requiring $O(n_j \times n_T)$ steps overall. Finally, 3-sheets are constructed and expanded in linear time. Therefore, the overall complexity of our approach is dominated by the quadratic term induced by the Jacobi fiber surface computation. This observation is confirmed by the numerical experiments reported in Tab. 1, where the column $JE$, $1S$, $2S$ and $3S$ report the computation time, with one thread, for the Jacobi edge classification and the extraction of the 1-, 2- and 3-sheets respectively. In particular, the Jacobi fiber surface extraction clearly dominates the process, with 95% of the computation time on average.

In practice, as illustrated in Tab. 1, the computation time for the Jacobi fiber surfaces is indeed related to the number of Jacobi edges $n_j$. In particular, the minimum overall computation time is observed for the data-set which minimizes its number of Jacobi edges (the Bitorus data-set) although it is one of the largest in terms of number of tetrahedra $n_T$. Reciprocally, the second longest computation occurs for the data-set that maximizes $n_j$ although it is one of the smallest in terms of $n_T$. This shows that the practical running time of our algorithm is more dependent on the topological complexity of the input bivariate function $f$ than on the size of $\mathscr{M}$. This demonstrates the output-sensitive nature of our algorithm since the number of 3-sheets in the Reeb space is dependent on the topological complexity of $f$ (expressed as the number of Jacobi edges).

As discussed in Sec. 3.4, most of the computations of our algorithm can be trivially parallelized. However, two sub-routines (the extraction of the 1- and 3-sheets) involve breadth-first search traversals, which are intrinsically sequential procedures. As reported in Tab. 1, the sum of these two steps only represent 3% of the computation time on average. Therefore, these two sequential sub-routines should not penalize the overall parallel performance. In practice, we implemented shared-memory parallelism with OpenMP and performance numbers are reported in the column P in Tab. 1 (using 12 cores). The corresponding speedup with regard to the sequential version of our algorithm (next column) is equal to 9.7 on average (out of 12 ideally), which indicates an efficient parallelization. In particular, the practical effect of this parallelization is to bring the computation time down to an interval between a few seconds and a minute and a half for all of our data-sets.

### 4.2 Comparison

Tab. 2 provides a computation time comparison with prior approaches. We first compare to the Joint Contour Net (JCN) [6] with a sequential C++ implementation provided by the authors of the technique. As discussed in Sec. 1.1, this algorithm approximates the Reeb space by considering the connected components of the pre-image of a pixel-based quantization of the range. The pixel-size in this quantization approach is a user-defined parameter. We report experiments for two

values of this parameter, chosen such that the range gets quantized by a 1x1 and a 100x100 rasterization respectively. The former quantization (1x1) will obviously approximate the Reeb space with only one pixel and it will therefore miss all of its features. We believe the second quantization (100x100) provides a modest yet reasonable rasterization resolution. The columns JCN-1 and JCN-100 of Tab. 2 report the corresponding computation times. While it takes minutes of computation for the 1x1 rasterization, it reaches hours of computation for the 100x100 rasterization, for most of our data-sets. The bold numbers in these columns indicate the speedup obtained with the *sequential* version of our algorithm. These indicate an average speedup of 21 over the 100x100 rasterized JCN, which is further increased to 218 if we compare to the parallel version of our algorithm. In addition to this two order of magnitude speedup, in contrast to the JCN, our approach is parameter-free and directly computes the exact Reeb space.

Next we compare to a bivariate specialization of the dimension-independent algorithm presented by Edelsbrunner et al. [15], which we discussed in Sec. 2.2. We recall that no implementation of this algorithm has been documented so far to the best of our knowledge. We therefore implemented this approach ourselves in C++ by considering in particular the 2-sheet of each edge of the input mesh. Due to the extremely long computation times obtained by this approach, we parallelized it on a per edge basis and the computation times reported in the column Chambers-P of Tab. 2 have been obtained with 12 cores. Even when parallelized, this approach typically requires several hours of computation (more than 24 hours for the largest data-set, implying days of computation in sequential mode). This illustrates that this algorithm is not tractable in practice. The bold numbers in this column indicate the speedup obtained by the *parallel* version of our algorithm, reporting from one to three orders of magnitude speedups.

In conclusion, in contrast to approximation techniques based on range quantization, our approach computes the exact Reeb space while still providing orders of magnitude speedups with regard to existing algorithms. We believe this is an important result for the applicability of the Reeb space to visualization as it brings its computation time from hours or days down to less than two minutes (column P, Tab. 1).

### 4.3 Limitations

In the univariate setting, in the vicinity of critical points, level sets change their topology, but not necessarily their connectivity. For instance, an isosurface can change its genus in the vicinity of a saddle while maintaining its number of connected components. Thus, the critical point contouring strategy will tend in practice to compute unneeded critical contours in these configurations, for which no branching actually occurs in the Reeb graph. Our approach suffers from the same limitation: fiber components change their topology when crossing a saddle Jacobi edge but not necessarily their connectivity. For instance, a fiber can evolve from a closed to an open curve in the vicinity of a Jacobi edge. Such an edge will trigger the computation of a Jacobi fiber surface in our approach although fibers do not necessarily disconnect there. This implies unnecessary computations.

Moreover, in the univariate setting, the time performance of the critical point contouring strategy deteriorates and converges towards that of the vertex-based contouring strategy for pathological cases of scalar fields that resemble random fields since for such fields, the number of critical points is no longer negligible with regard to the number of vertices in the mesh. The same limitation applies to our work. However, as illustrated in our experiments, we found that the number of Jacobi edges was sufficiently low for practical data-sets to guarantee good performances, even for noisy bivariate fields (Fig. 10(a)).

Finally, unlike the univariate case, extrema edges also need to be considered in the generalization of the critical point contouring to the bivariate case (Fig. 7). However in practice, the corresponding Jacobi fiber surface may be empty or may not contribute to the structure of the Reeb space (Fig. 8), which implies unnecessary computations.

## 5 Application: Continuous Scatterplot Peeling

To demonstrate its versatility and utility, we apply our algorithm to bivariate data segmentation tasks. As described in Sec. 2.2, for each

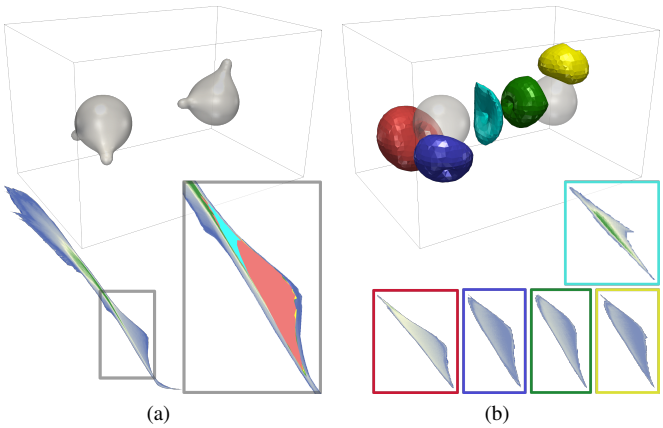



Fig. 11. Reeb space driven visualization of a water dimer (a, top). Although the continuous scatterplot (CSP) of the electron density and its gradient magnitude does not exhibit clear discontinuities above the diagonal (a, bottom), some of the largest 3-sheets of the Reeb space overlap in this area (inset zoom). Each of these layers can be visualized in an independent CSP (b, bottom). Four of them have a very similar shape (red, blue, green and yellow). These correspond to covalent bonds between oxygen and hydrogen atoms, which are captured by the corresponding 3-sheets (b, top). The remaining 3-sheet (cyan) denotes the hydrogen bond linking the two molecules.

point $q \in \mathbb{R}^2$, each of the connected components of $f^{-1}(q)$ will be located in a distinct 3-sheet. In other words, the segmentation of $\mathscr{M}$ into the 3-sheets of $\mathscr{R}(f)$ enables the automatic separation of the regions of the domain which overlap in the range.

Based on this observation, we introduce *continuous scatterplot peeling*, a technique which enables the segmentation of the data into overlapping layers in the continuous scatterplot (CSP) [1]. Given the segmentation of $\mathscr{M}$ into the 3-sheets of $\mathscr{R}(f)$, we let users iterate through them in decreasing order of projection area (which also corresponds to the importance score $a(S)$ used in our simplification algorithm, Sec. 3.5). This process enables the review of the different 3-sheets of the segmentation in decreasing order of importance, both in the domain and the range. Next, users can select whether or not they wish to include the corresponding *layer* (the corresponding 3-sheet) in the continuous scatterplot. We finally compute the final continuous scatterplot(s) for the layers selected by the user with an implementation provided by the original authors of the technique. Figs. 1, 11 and 12 present several visualizations obtained with this technique.

Fig. 1 presents a classical vortex street flow simulation (flow turbulence behind an obstacle). The Reeb space of the flow velocity and curl magnitudes enables an automatic segmentation of the domain. In particular, the 3 largest 3-sheets of $\mathscr{R}(f)$ correspond to regions before the obstacle (Fig. 1(a), top). These regions have little interest for the application, however they overlap most of the CSP (Fig. 1(b), left). Note that due to the symmetry of the vortex street, the blue and green 3-sheets overlap nearly perfectly in the CSP. Thanks to the segmentation of $\mathscr{M}$ into the 3-sheets of the Reeb space, these regions can be easily removed from the projection, which results in a less cluttered CSP (Fig. 1(b), right), better revealing the projections of the turbulent features of the flow. At this point, some of the largest remaining 3-sheets are shown (Fig. 1(a), bottom), capturing the geometry of the turbulent flow in the domain. These 3-sheets can be independently visualized in the CSP. In particular, the top, center and bottom CSPs (Fig. 1(c)) represent the regions with matching colors in the domain (top, center and bottom regions of the vortex street respectively).

Fig. 11 shows a Reeb space driven visualization of a water dimer. In quantum chemistry, molecular simulations are often analyzed in the light of two scalar fields: the electron density and a function of its gradient magnitude [26]. Such a bivariate function is represented in a CSP (Fig. 11(a), bottom), where points on the diagonal are known to correspond to regions of space where no chemical interaction occurs. The automatic segmentation of this data into the 3-sheets of the Reeb

space indicates 5 large 3-sheets that project off this diagonal (zoom). Although the CSP does not exhibit clear discontinuities in these areas, these 5 3-sheets all overlap. Our Reeb space based segmentation enables the independent CSP visualization for each of these regions (Fig. 11(b), bottom). Note that four of these CSPs are very similar in shape. They correspond to 3-sheets that capture the covalent bonds between the hydrogen and oxygen atoms (matching colors, Fig. 11(b), top). The remaining 3-sheet (cyan) exhibits a CSP of different shape: the corresponding region in the domain denotes the hydrogen bond linking the two molecules, as expected by the chemists. Note that this subtle, yet important feature projects to a small cyan area in the range. This means that a JCN approximation of the Reeb space may miss this feature for insufficient quantization thresholds.

Fig. 12 presents the Reeb space driven visualization of a flow simulation within a mechanical piece (Fig. 12(a)). The considered input bivariate function is given by the flow velocity and curl magnitudes. Originally, the Reeb space counts 4863 3-sheets. Our Reeb space simplification algorithm (Sec. 3.5) progressively merges adjacent 3-sheets in increasing order of range area, until a user threshold of 5% of the overall area of $f(\mathcal{M})$ is reached. At this threshold, this simplification results in the automatic identification of 7 remaining 3-sheets (Fig. 12(b)). These correspond to the main features of the flow: its two inlets (dark and light blue), its two outlets (orange and red) and three interleaved regions in the turbulence area (inset zoom). Once projected, this simplified Reeb space (Fig. 12(c), right) enables an easier understanding of the structure of the CSP, as compared to the original Reeb space (center). The overlap of the projected 3-sheets can be addressed by CSP peeling, to inspect layers independently (Fig. 12(d)).

## 6 Conclusion

In this paper, we presented an efficient algorithm for the computation of Reeb spaces of bivariate functions defined on tetrahedral meshes. By developing a comprehensive analogy with the univariate case, we believe we have given a simple and intuitive presentation of bivariate Reeb spaces. We detailed and discussed the core algorithmic similarities and differences between the univariate and the bivariate settings. As a result, we presented an algorithm that extends to the bivariate case the critical point contouring strategy used for fast Reeb graph computation in the univariate setting. This results in an efficient, output-sensitive and parallel technique for Reeb space computation. While our algorithm is very simple, it yields several orders of magnitude speedups with regard to previous work, as detailed through our experimental results. We believe this is an important practical result for the applicability of the Reeb space to scientific visualization as it brings its computation time from hours or days down to a few dozens of seconds, hence making bivariate Reeb space computation tractable for the first time.

We demonstrated the utility of our algorithm by using the Reeb space as a semi-automatic segmentation tool for bivariate data. In particular, we showed that it could be used to separate overlapping features in the continuous scatterplot, hence reducing its clutter and enabling further localized inspections. Beyond this application, we believe this work opens several exciting avenues for the generalization of topology based techniques to bivariate data, including for instance feature similarity estimation (as suggested in Fig. 11), automatic feature segmentation for quantitative analysis, silhouette and Jacobi set simplification (as suggested in Fig. 10), 2-dimensional skeletal structure extraction for shape modeling (as suggested in Fig. 5), etc. We hope that our companion C++ implementation will help the community in the development of such Reeb space based visualization techniques.


## Acknowledgments

This work is partially supported by the Bpifrance grant *"AVIDO"* (Programme d'Investissements d'Avenir, reference P112017-2661376/DOS0021427) and the EPSRC grant EP/J013072/1. The author would like to thank the anonymous reviewers for their thoughtful remarks and suggestions. Special thanks go to Joshua A. Levine for his careful proofreading and precious feedback. The vortex street data-set (Fig. 1) has been kindly provided by Robert Geist and Joshua A. Levine.

This paper is dedicated to my daughter Amy.

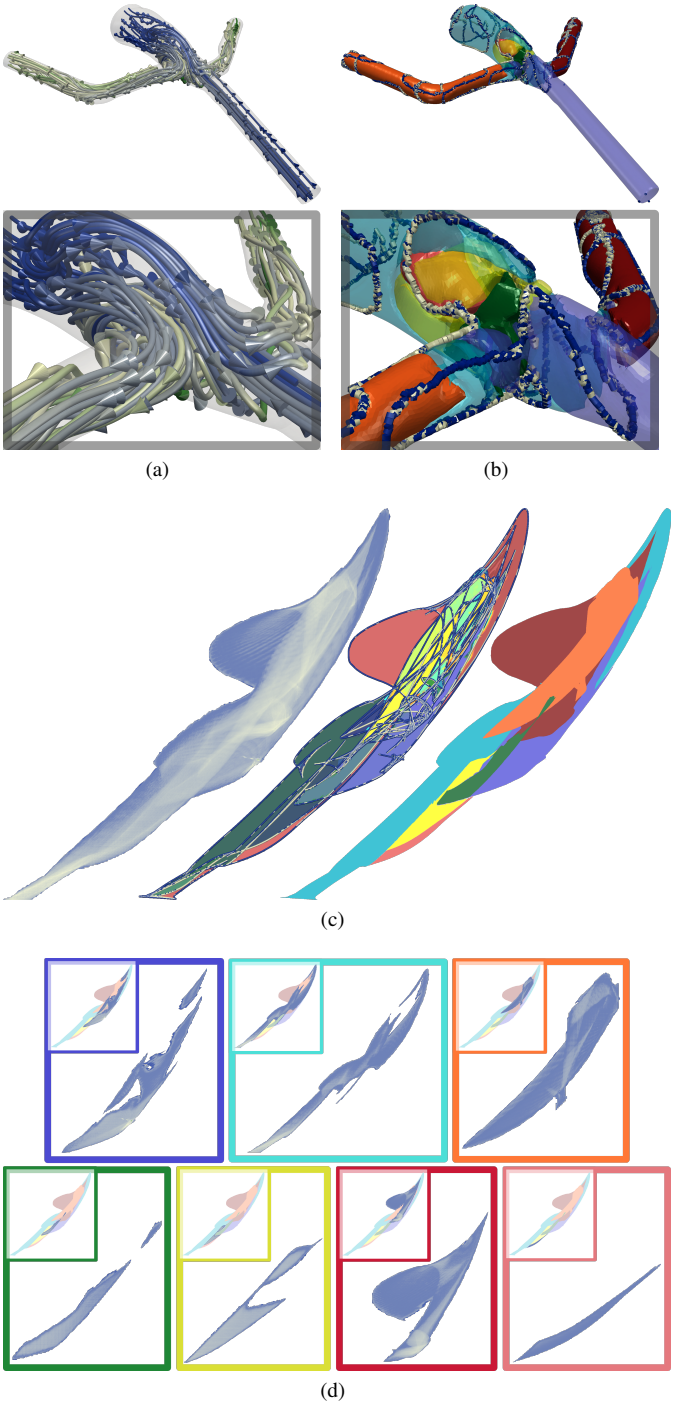




Fig. 12. Continuous scatterplot peeling for the flow velocity and curl magnitudes within a mechanical piece. The color shade from blue to green indicates integration time (a). While the Reeb space initially counts 4863 3-sheets, after simplification, only 7 remain (b). These 3-sheets correspond to two inlet and two outlet regions, as well as 3 regions in the area of turbulence (inset zoom). Flow velocity and curl magnitudes correspond to the X and Y axes in the continuous scatterplot (c). The projection of the simplified Reeb space (c, right) yields less clutter than the original one (c, center, blue and white cylinders denote the projections of extrema and saddle edges). However, several projected 3-sheets still overlap. Continuous scatterplot peeling enables the visualization of each of these layers independently (d, matching colors).

## REFERENCES


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

[2] T. F. Banchoff. Critical points and curvature for embedded polyhedral surfaces. *The American Mathematical Monthly*, 1970.

[3] S. Biasotti, B. Falcidieno, and M. Spagnuolo. Extended Reeb graphs for surface understanding and description. In *Discrete Geometry for Computer Imagery*, 2000.

[4] S. Biasotti, D. Giorgio, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 2008.

[5] O. Burlet and G. De Rham. Sur certaines applications génériques d'une variété close à 3 dimensions dans le plan. *L'Enseignement Mathématique*, 1974.

[6] H. Carr and D. Duke. Joint contour nets. *IEEE Transactions on Visualization and Computer Graphics*, 2014.

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

[8] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. In *Proc. of Symposium on Discrete Algorithms*, pages 918–926, 2000.

[9] H. Carr, J. Snoeyink, and M. van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *Proc. of IEEE VIS*, pages 497–504, 2004.

[10] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in Reeb graphs of 2-manifolds. In *Proc. of ACM Symposium on Computational Geometry*, pages 344–350, 2003.

[11] L. De Floriani, U. Fugacci, F. Iuricich, and P. Magillo. Morse complexes for shape segmentation and homological analysis: discrete models and algorithms. *Computer Graphics Forum*, 2015.

[12] H. Doraiswamy and V. Natarajan. Output-sensitive construction of reeb graphs. *IEEE Transactions on Visualization and Computer Graphics*, 2012.

[13] H. Edelsbrunner and J. Harer. *Jacobi Sets of Multiple Morse Functions*. Cambridge Books Online, 2004.

[14] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2009.

[15] H. Edelsbrunner, J. Harer, and A. K. Patel. Reeb spaces of piecewise linear mappings. In *Proc. of ACM Symposium on Computational Geometry*, 2008.

[16] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28:511–533, 2002.

[17] H. Edelsbrunner and E. P. Mucke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9:66–104, 1990.

[18] R. Fuchs and H. Hauser. Visualization of multi-variate scientific data. *Computer Graphics Forum*, 2009.

[19] A. Gyulassy, P. Bremer, R. Grout, H. Kolla, J. Chen, and V. Pascucci. Stability of dissipation elements: A case study in combustion. *Computer Graphics Forum (Proc. of EuroVis)*, 2014.

[20] A. Gyulassy, P.-T. Bremer, B. Hamann, and V. Pascucci. A practical approach to Morse-Smale complex computation: scalabity and generality. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, pages 1619–1626, 2008.

[21] A. Gyulassy, A. Knoll, K. Lau, B. Wang, P. Bremer, M. Papka, L. A. Curtiss, and V. Pascucci. Interstitial and interlayer ion diffusion geometry extraction in graphitic nanosphere battery materials. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2015.

[22] A. Gyulassy, V. Natarajan, M. Duchaineau, V. Pascucci, E. Bringa, A. Higginbotham, and B. Hamann. Topologically Clean Distance Fields. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 13:1432–1439, 2007.

[23] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proc. of ACM SIGGRAPH*, 2001.

[24] L. Hüttenberger, C. Heine, H. Carr, G. Scheuermann, and C. Garth. Towards multifield scalar topology based on pareto optimality. *Computer Graphics Forum (Proc. of EuroVis)*, 32(3):341–350, 2013.

[25] H. Janicke, M. Bottinger, and G. Scheuermann. Brushing of attribute clouds for the visualization of multivariate data. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2008.

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

[27] J. Kehrer and H. Hauser. Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 2013.

[28] P. Klacansky, J. Tierny, H. Carr, and Z. Geng. Fast and exact fiber surfaces for tetrahedral meshes. *IEEE Transactions on Visualization and Computer Graphics*, 2016.

[29] D. E. Laney, P. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci. Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2006.

[30] D. Lehmann and H. Theisel. Discontinuities in continuous scatterplots. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2010.

[31] J. Milnor. *Morse Theory*. Princeton U. Press, 1963.

[32] E. Munch and B. Wang. Convergence between categorical representations of reeb space and mapper. In *Proc. of ACM Symposium on Computational Geometry*, 2016.

[33] S. Nagaraj, V. Natarajan, and R. Nanjundiah. A gradient-based comparison measure for visual analysis of multifield data. *Computer Graphics Forum (Proc. of EuroVis)*, 2011.

[34] S. Parsa. A deterministic o(m log m) time algorithm for the reeb graph. In *Proc. of ACM Symposium on Computational Geometry*, 2012.

[35] V. Pascucci, G. Scorzelli, P. T. Bremer, and A. Mascarenhas. Robust on-line computation of Reeb graphs: simplicity and speed. *ACM Transactions on Graphics (Proc. of ACM SIGGRAPH)*, 26:58.1–58.9, 2007.

[36] G. Patanè, M. Spagnuolo, and B. Falcidieno. A minimal contouring approach to the computation of the reeb graph. *IEEE Transactions on Visualization and Computer Graphics*, 15:583–595, 2009.

[37] G. Reeb. Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique. *Comptes-rendus de l'Académie des Sciences*, 222:847–849, 1946.

[38] O. Saeki. *Topology of Singular Fibers of Differentiable Maps*. Springer, 2004.

[39] N. Sauber, H. Theisel, and H. Seidel. Multifield-Graphs: An approach to visualizing correlations in multifield scalar data. *IEEE Transactions on Visualization and Computer Graphics*, 12:917–924, 2006.

[40] Y. Shinagawa, T. Kunii, and Y. L. Kergosien. Surface coding based on morse theory. *IEEE Computer Graphics and Applications*, 1991.

[41] H. Si. Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software*, 2015.

[42] B. S. Sohn and C. L. Bajaj. Time varying contour topology. *IEEE Transactions on Visualization and Computer Graphics*, 2006.

[43] D. M. Thomas and V. Natarajan. Multiscale symmetry detection in scalar fields by clustering contours. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2014.

[44] J. Tierny, A. Gyulassy, E. Simon, and V. Pascucci. Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 15:1177–1184, 2009.

[45] M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pasucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *Proc. of ACM Symposium on Computational Geometry*, 1997.

[46] G. Weber, S. E. Dillard, H. Carr, V. Pascucci, and B. Hamann. Topology-controlled volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 2007.

[47] K. Weiss, F. Iuricich, R. Fellegara, and L. De Floriani. A primal/dual representation for discrete Morse complexes on tetrahedral meshes. *Computer Graphics Forum (Proc. of EuroVis)*, 2013.

[48] Z. Wood, H. Hoppe, M. Desbrun, and P. Schroder. Removing excess topology from isosurfaces. *ACM Transactions on Graphics*, 2004.