Distance between Extremum Graphs

Vidya Narayanan* Dilip Mathew Thomas† Vijay Natarajan‡

Indian Institute of Science, Bangalore

ABSTRACT

Scientific phenomena are often studied through collections of related scalar fields generated from different observations of the same phenomenon. Exploration of such data requires a robust distance measure to compare scalar fields for tasks such as identifying key events and establishing correspondence between features in the data. Towards this goal, we propose a topological data structure called the complete extremum graph and define a distance measure on it for comparing scalar fields in a feature-aware manner. We design an algorithm for computing the distance and show its applications in analysing time varying data.

Keywords: scalar field topology, extremum graphs, distance measure.

Index Terms: I.3.5 [COMPUTER GRAPHICS]: Computational Geometry and Object Modeling

1 INTRODUCTION

Many scientific experiments and simulations measure physical quantities over a domain of interest. The data thus generated is commonly represented as scalar fields. In the past, there has been a lot of research focus to develop methods to explore scalar field data and to identify important features in the data. Collections of closely related scalar field datasets like time varying data and ensemble data are often generated by scientists to better understand the underlying scientific phenomenon of interest. Newer paradigms are needed to analyse such datasets. An important step in this direction is to develop robust methods to compare similar scalar fields. In this paper, we study the problem of quantifying the similarity between scalar fields and show its applications in exploring time varying data.

The features present in a scalar field dataset play a key role in understanding the properties of the scientific phenomenon being studied. Therefore, it is pertinent to design similarity measures for comparing scalar fields in a feature-aware manner. Topological methods using abstract representations of the scalar field have been quite successful in the exploration and visualization of scalar fields. Hence, it is natural to examine similarity between scalar fields based on the similarity between their topological abstractions. This has led to the development of distance measures like the barcode metric, the bottleneck distance, the interleaving distance, and the functional distortion distance.

The extremum graph is a topological data structure introduced by Correa et. al. to design a visual representation of scalar fields called topological spines and captures the proximity between the extrema in the scalar field [10]. We construct a complete extremum graph that captures the proximity between every pair of extrema. To compare two scalar fields, we design a distance measure based on the maximum weight common subgraph between their complete extremum graphs. Since computing the maximum common subgraph is costly, we leverage the hierarchical structure of the complete extremum graph to compute the distance measure. Before we describe the proposed method, we briefly review existing methods for comparing scalar fields.

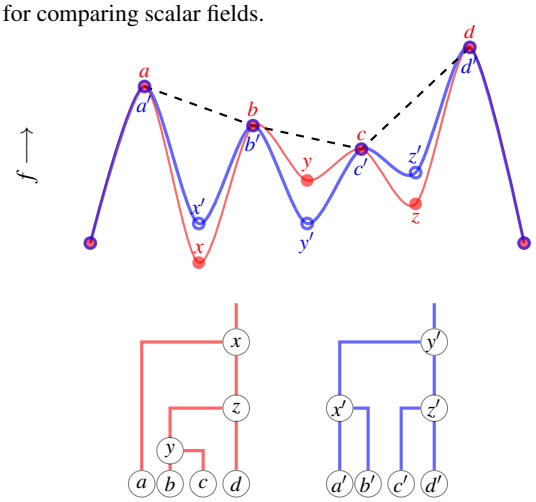

Figure 1: A merge tree, and its superset, the contour tree, captures the nesting structure of level sets. For the two functions shown above (blue and red), no branch decomposition of the merge tree (below) reflects the correspondence between both maxima b to b' and c to c'. The extremum graph (dashed) captures proximity between the extrema and can provide a more intuitive correspondence between them.

1.1 Related Work

The size and complexity of scalar field datasets make it impractical to compare them directly in a feature-aware manner. Several distance measures defined on topological abstractions have been proposed in the literature to compare them indirectly. Topological methods associate a notion of importance, called persistence, with topological features in scalar fields [13]. Persistence diagrams encode the persistence of features as points in a plane [9]. The persistence diagrams of two scalar fields may be compared and the similarity between them can be measured using a distance function like the bottleneck distance. For this, the maximum of the supremum distance between a point and its image under L_∞ norm is considered for a given mapping between the points. The bottleneck distance considers distances measured by all possible mappings and is the infimum among them. It is known that the persistence diagram is stable under perturbations of the underlying function. Therefore, a large bottleneck distance implies that the underlying functions are also dissimilar. Closely related to the persistence diagrams is the barcode descriptor which represents persistence of features as intervals in the real line and has been used as a shape descriptor for point clouds [8]. The barcode metric to compare them is computed using a maximum weight bipartite matching that maximizes the intersection between the interval representation of the features. The persistence diagram representation of a scalar field does not capture the neighbourhood relationship between the features since no

*e-mail:sercvidya@ssl.serc.iisc.in
†e-mail:dilip@csa.iisc.ernet.in
‡e-mail:vijayn@csa.iisc.ernet.in

adjacency constraints are imposed on the points in the persistence diagram. This limits their use for comparing scalar fields.

The Reeb graph of a scalar field is a topological data structure that tracks the changes in the number of connected components of its level sets. Recent years have seen distance measures defined on Reeb graphs and its simpler variant called the merge tree. Morozov et al. define a distance between two merge trees called the interleaving distance. They consider two continuous maps that shifts the points in each merge tree onto the other under certain constraints and define the interleaving distance as the smallest possible shift under which such maps exist [18]. Beketayev et al. propose another distance measure on merge trees by examining the branch decomposition representations of a merge tree with the aim of identifying isomorphic subtrees through cost functions for matching and removing vertices in the branch decomposition representation [2]. The distance is defined as the minimum cost for generating isomorphic subtrees among all possible branch decompositions of the two merge trees. Similar to the interleaving distance, Bauer et al. define the functional distortion distance between two Reeb graphs by considering two continuous functions that map points in each Reeb graph onto the other and minimizing, among all maps, the distortion in the scalar values of the points under the map and their composition [1].

While the above methods define distances between topological structures in a rigorous manner, several methods use simpler notions to identify similarity between scalar fields. One such approach is to define a similarity score between the Reeb graphs and its loop-free variant, namely, the contour tree and has been used for matching shapes [17, 34] and identifying repeating features [29]. Saikia et al. recently introduced the extended branch decomposition graphs as an efficient data structure to encode branch decompositions of all subtrees of the merge tree. This is used to identify repeating features and periodicity in time varying data through a matching procedure based on dynamic programming [23]. The ability of the extremum graph to capture proximity relationship of the features has been used to identify similar features within a dataset by comparing the geodesic distance between pairs of extrema computed using an augmented version of the extremum graph [30]. Other approaches compare level sets using their mutual information and shape descriptors for identifying important isovalues [5, 16] and repeating features [28] respectively. The correlation between different functions which are not necessarily similar but defined on the same domain have also been studied for exploring multifield data [19, 24, 25].

The graph structure of the Reeb graph and merge tree naturally enforces adjacency constraints on the features in a scalar field. However, these constraints relate to the merging and splitting of level sets and an edge may connect features that do not have close proximity. Figure 1 shows two simple 1D functions with different nesting structures of the sublevel sets in the merge tree. When these functions are compared using merge tree based measures [2, 18], the extrema b, c and b', c' will be simplified rather than matched. On the other hand, extremum graph based measures can associate these extrema and their descending manifolds intuitively because they are based on proximity. The extremum graph is designed to capture the proximity of features in the field and our comparison measure therefore enforces proximity based constraints. Further, while most of these structures are graphs, graph theoretic distance measures [6, 7] have not been adapted for the purpose of comparing them. The complete extremum graph structure we introduce allows us to design maximum common subgraph based distances for comparing extremum graphs. Our distance measure is feature-aware and inherently handles topological noise that appear as insignificant features.

1.2 Contributions

We introduce a new distance measure between extremum graphs of scalar fields. The following are the main contributions of this paper:

- We introduce the notion of a complete extremum graph that associates proximity information for all pairs of extrema in the extremum graph. The construction helps define a distance between extremum graphs even when they differ in terms of the number of maxima. We describe a simple algorithm to construct the complete extremum graph.
- We introduce a feature-aware distance measure between extremum graphs. Our distance measure is based on the maximum common subgraph of the complete extremum graphs. We discuss graph pruning and partitioning strategies to effectively compute the distance measure.
- We demonstrate the effectiveness of the distance measure by applying it to understand periodicity and to track features in time-varying data sets.

2 Complete Extremum Graphs

In this section we discuss the complete extremum graph and describe an algorithm to compute it.

2.1 Extremum Graphs

Let $f : \mathrm{D} \to \mathbb{R}$ be a scalar function defined on a smooth n-dimensional manifold, D. For $x \in \mathrm{D}$, if the gradient $\nabla f(x) = 0$, x is called a *critical point* of f on D. All other points are called *regular*. The function f is called a *Morse function*, if all critical points are non-degenerate i.e., the Hessian matrix of second order partial derivatives is non-singular. Critical points can be classified using an index which equals the number of independent directions along which f decreases. The index of a *minimum* is 0 and of a *maximum* is n. *Saddles* have indices from $n-1$ to 1. Gradients are well defined for regular points. Morse functions allow for a decomposition of the domain D based on the integral curves of ∇f. The union of all integral curves that terminate at a maximum define the *descending manifold* of that maximum. An analogous segmentation can be considered based on minima and their *ascending manifolds* [12].

Correa et. al [10] introduced the extremum graph structure to develop a planar visual representation of a scalar field called topological spines. An extremum graph is a representation of the Morse decomposition. The graph is called a maximum (minimum) graph, when the decomposition is based on the descending (ascending) manifolds. For many datasets, prominent features are expressed by either its descending or ascending manifolds. For such data, the extremum graph provides a suitable abstraction by representing its features and their proximity. As the maximum graph of f is equivalent to the minimum graph of $-f$, we restrict our discussion to maximum graphs and refer to them as extremum graphs.

We combinatorially represent the extremum graph of a scalar field $f : D \to \mathbb{R}$ by $EG_f(V,E)$. The vertex set V consists of the maxima of f. A pair of vertices v_i and v_j share a saddle s_{ij} if paths of steepest ascent from s_{ij} terminate at v_i and v_j. The edge (v_i, v_j) is used to denote this adjacency between the descending manifolds of v_i and v_j. $\mathcal{S}(v_i, v_j)$ denotes the saddle associated with the edge (v_i, v_j).

To identify similarity between scalar fields, we define a distance measure over extremum graphs that takes into account the edge structure while trying to minimize the perturbation required to match its vertices. Our aim is to compare the underlying fields based on the distortion that needs to be introduced in them such that their extremum graphs become identical — given a threshold, δ, can a correspondence be achieved between the vertices such that the underlying function needs to be modified by at most δ, under a

suitable norm? Under this correspondence, is the edge structure of the extremum graph maintained?

2.2 Complete Extremum Graphs

A pair of extremum graphs can differ in the number of maxima they represent and their adjacency relationships. In order to compare them, we introduce the notion of a *complete extremum graph*. The complete extremum graphs allows edges between all pairs of vertices in the graph. It associates with each vertex, a cost that depends on the perturbation necessary in f to simplify the extremum. It associates with each edge, the cost of introducing it. This cost represents the extent of perturbation required in f to introduce a shared saddle between the descending manifolds of the two end point maxima of the edge. This allows for comparing proximity relationships between all pairs of maxima. An importance measure that is often

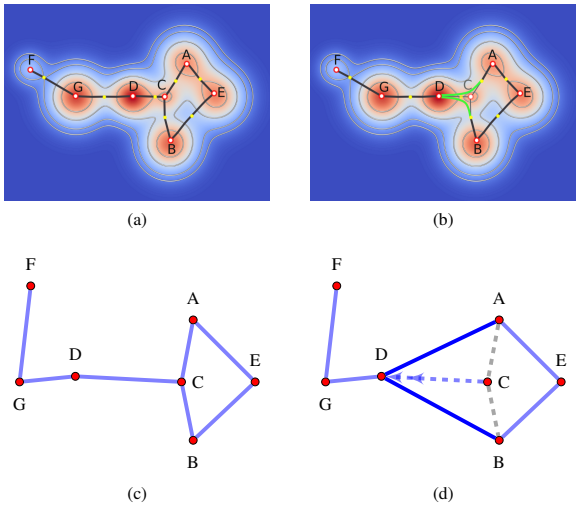

Figure 2: Extremum graphs provide an abstraction of scalar fields and encodes adjacency relationships between its extrema. (a) shows a 2D scalar function overlayed with its extremum graph. (b) The extremum C can be simplified by a merger into extremum D, following which D becomes adjacent to A and B, after this simplification A and B appear much farther away in the graph. (c) shows a combinatorial representation of the extremum graph. (d) Simplification of vertex C into vertex D involves contraction of edge (C,D).

associated with critical points is its *persistence* [14]. Consider the extremum graph in Figure 2. The maximum C can be *simplified* by merging it with an adjacent maximum D. In terms of the underlying function f, the maximum C and the saddle $\mathcal{S}(\mathtt{C},\mathtt{D}) = \mathtt{S}$ are *cancelled* by reversing the gradient flow direction along that path. To realize this modification, the function f has to be modified by at most $f(\mathtt{C}) - f(\mathtt{S})$. This introduces adjacencies between the surviving maximum D and the other neighbours of C. The maximum C and its associated edges are removed. In terms of the combinatorial graph, this refers to contracting the edge $(\mathtt{C},\mathtt{D})$. To optimally simplify a function f, with respect to the L_∞ norm, edges can be ordered according to the function difference their removal demands and simplified in increasing order. The persistence of a critical point refers to the difference associated with it at the time of its removal.

Hierarchical structures have been introduced for Morse-Smale complexes [3], contour trees [21, 27], and extremum graphs [10]. They simplify extrema in increasing order of persistence and update the edge structure based on the simplification. Such approaches allow for addition of edges between surviving maxima. A natural cost that can be associated with the inserted edge is the persistence of the merged maximum that introduces it. However, there are two issues with this approach. First, simplification strictly follows the order of persistence. If persistence values are not well separated, a small perturbation in function values can lead to different graph representations. These choices further restrict subsequent simplifications and make comparisons difficult. Second, persistence is a measure designed to identify the minimum perturbation necessary to simplify extrema. The edge structure that follows the simplification is a consequence of this choice. Therefore, the simplification threshold at which an edge appears can overestimate the perturbation necessary to introduce adjacencies. In Figure 2, the maxima A and B can be made adjacent by merging the intermediate maximum C with A. However, persistence directed simplification will merge C with D. Following this simplification, A and B appear to be further away in the graph. Although persistence provides an intuitive measure of importance for vertices, the ensuing simplification can provide an unintuitive measure for the edges. This motivates the design of a new measure that captures the cost of introducing an edge in the extremum graph.

Let $< v_1, p_1, p_2, \ldots, p_n, v_2 >$ be a path between maxima v_1 and v_2 in the extremum graph. If all the intermediate maxima p_i can be simplified and merged into the end points, v_1 and v_2 become adjacent. The perturbation required for this simplification, estimates the cost for introducing the edge via that path. The minimum cost over all paths between a pair of maxima provides an accurate cost to introduce the edge between them. Edges cannot be independently eliminated during simplification. The cost of eliminating an edge is implicitly equivalent to eliminating one of its vertices. We therefore only consider edges, whose costs do not exceed the persistence of its end points and bound edge costs by the minimum of the persistence of its end points.

We represent the complete extremum graph as an attributed graph $G_f(V,E)$. The vertices of the graph are identical to the vertices of the extremum graph. We associate with each vertex v_i, its persistence denoted by $\mathcal{P}(v_i)$. The edges of the complete extremum graph extends the edge set of the extremum graph by including edges between all pairs of maxima.The cost of an edge is denoted by $\mathcal{C}((v_i,v_j))$ such that $\mathcal{C}((v_i,v_j)) \leq min(\mathcal{P}(v_i),\mathcal{P}(v_j))$. We normalize all scalar functions to have the range $[0,1]$ ensuring $0 \leq \mathcal{P}(v_i) \leq 1$ and $0 \leq \mathcal{C}((v_i,v_j)) \leq 1$. The persistence of the global maximum is set to 1. We derive a distance measure between extremum graphs based on these vertex and edge attributes.

2.3 Computation

We compute the extremum graph based on an approximate Morse decomposition [31]. The approximate decomposition is fast and dimension independent but may introduce saddles between pairs of maxima whose descending manifolds do not share a saddle in the true decomposition. Since we generate a complete graph, these additional edges have little consequence. Computing all paths between every pair of maxima is computationally infeasible. We only require the minimum cost of a path between a pair of maxima. We compute this by modifying the simplification algorithm to consider all relevant paths between maxima.

Algorithm 1 generates the complete extremum graph $\mathrm{G}_f(V,E')$ for an input extremum graph $\mathrm{EG}_f(V,E)$ by computing persistence $\mathcal{P}$ for all maxima and edge costs $\mathcal{C}$ for all pairs of maxima. Edges of the extremum graph are inserted in a priority queue, $\mathcal{Q}$. The *priority* of an edge (v_i,v_j) with saddle $\mathcal{S}(v_i,v_j) = s$ is the cost of simplifying the edge, i.e., $f(v_i) - f(s)$. We assume that $f(s) < f(v_i) < f(v_j)$ and simplifying an edge implies cancelling the maximum v_i and saddle s to merge them into v_j . Note that the priority of an edge indicates the function modification required to simplify the edge and the cost associated with each edge indicates the function modification required to introduce the edge. When an edge is popped from the priority queue, unlike persistence simplification, where all the other edges associated with that maximum are deleted, we retain all other edges for subsequent simplification along other paths. We introduce new edges between the neighbours of the simplified max-

imum v_i and the surviving maximum v_j. The set of neighbours of a vertex v is denoted by $N(v)$.

Note that while the other edges associated with the simplified maximum are retained, each edge is simplified only once. The value associated with new edges introduced due to simplification is not less than that of the simplified edge. During simplification, an edge may appear between an already adjacent pair of maxima. Suppose, an edge exists between a pair of maxima with cost c. After further simplification, if another edge is introduced between the same pair at a cost $c' > c$, that introduces a higher saddle, we only update the saddle. Since edge costs are computed in a monotonically increasing fashion, the higher saddle is only used for pairs that have cost greater than c'.

Figure 3 shows a few steps in the construction of the complete extremum graph for the graph shown in Figure 2. First, edge (F,G) is popped, and the persistence of vertex F is recorded. As the vertex F has no neighbours, no further edges are added. The next edge with lowest function difference is (C,D) and is popped from $\mathcal{Q}$. The persistence of the vertex C is recorded. By simplifying vertex C, the maximum D now neighbours maxima A and B. The simplified edge (C,D) is considered processed and the vertex C and D are no longer not considered neighbours in subsequent steps. As edges associated with C are still maintained in $\mathcal{Q}$, (C,A) is popped. The persistence of C is not changed and this simplification is used to identify and add the edge (A,B). Next (C,B) is simplified followed by (A,E) introducing (E,D) and the persistence of A is recorded. The remaining edges are processed similarly.

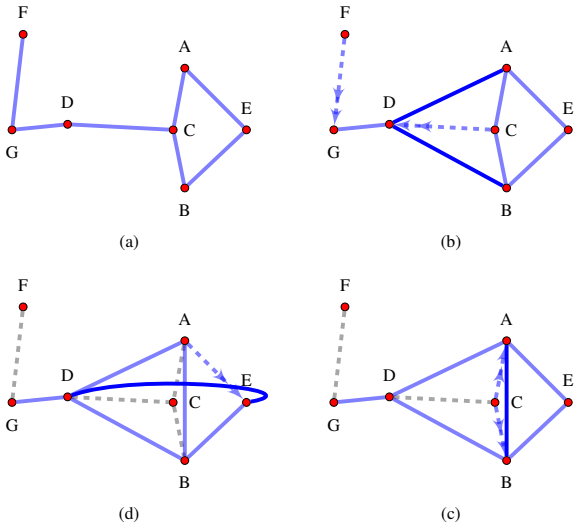

Figure 3: Construction of the complete extremum graph. (a) Input extremum graph EG. (b) (F,G) is processed. (C,D) is processed introducing (A,D) and (B,D). Other edges associated with C are retained.(c) (C,A) is processed and (A,B) is introduced. (d)(A,E) is processed introducing (E,D).

3 Distance between Extremum Graphs

To compute the distance between a pair of extremum graphs, we adapt the maximum common subgraph based measure [7] to the attributed complete extremum graphs. We find a correspondence between the vertices of the complete graphs with pairwise constraints enforced by the edge costs. The quality of correspondence between a pair of vertices is measured by the difference in the persistence of the maxima they represent and is refered to as *vertex distortion*. As the graph is complete, a pair of vertex correspondences determines their edge correspondence. The difference in the cost associated with the corresponding edges indicates the quality of the edge correspondence and we refer to this difference as *edge distortion*. Low

Algorithm 1: CEG-SIMP Complete Extremum Graph

Input: Extremum Graph $EG_f(V,E)$
Output: Complete Extremum Graph $G_f(V,E')$, $\mathcal{P}$ and $\mathcal{C}$
for each $(v_i,v_j) \in E$ **do**
  $\mathcal{C}((v_i,v_j)) = 0$
  $\mathcal{Q}.push((v_i,v_j))$ /* $priority((v_i,v_j)) = f(v_i) - f(s)$*/
  $\mathcal{P}(v_i) = \infty\ \mathcal{P}(v_j) = \infty$
while $!pq.empty()$ **do**
  $(v_i,v_j) = \mathcal{Q}.pop()$
  $ProcessedEdges.insert((v_i,v_j))$
  $E' = E' \cup (v_i,v_j)$
  $s = \mathcal{S}(v_i,v_j)$
  $c = priority((v_i,v_j))$
  if $\mathcal{P}(v_i) = \infty$ **then**
    $\mathcal{P}(v_i) = c$
  $N(v_i) = N(v_i) \setminus v_j$
  $N(v_j) = N(v_j) \setminus v_i$
  for $v' \in N(v_i)$ ***and*** $(v',v_j) \notin ProcessedEdges$ **do**
    $s' = \mathcal{S}(v',v_i)$
    if $\mathcal{C}((v',v_j)) > c$ **then**
      $\mathcal{C}((v',v_j)) = c$ /* Update cost*/
      $\mathcal{S}(v',v_j) = s'$
      $\mathcal{Q}.push((v',v_j))$
    else if $f(s') > f(s)$ **then**
      $\mathcal{S}(v',v_j) = s'$ /* Update saddle*/
      $\mathcal{Q}.push((v',v_j))$
  for $(v_i,v_j) \in E'$ **do**
    $\mathcal{C}((v_i,v_j)) = min(\mathcal{C}((v_i,v_j)), min(\mathcal{P}(v_i),\mathcal{P}(v_j))$

edge distortion between the edges of the complete extremum graphs indicates high structural similarity between the extremum graphs being compared. We introduce a parameter that controls edge distortion and compute vertex distortions introduced by mappings that satisfy the edge constraints introduced under this parameter.

3.1 Maps between extrema

Let $G_f(V,E_v)$ and $G_g(U,E_u)$ be the complete extremum graphs of $f : D \to [0,1]$ and $g : D \to [0,1]$. We extend the vertex set of G_f by a set of dummy vertices to obtain $V \cup \{\phi_{|V|+1}, \phi_{|V|+2}, \dots \phi_{|V|+|U|}\}$. We extend the edge set of G_f to include edges (v_i,ϕ_k) between all pairs of dummy vertices ϕ_k and vertices v_i. We similarly extend the vertex and edge set of G_g and represent its dummy vertices by ψ. The attributes of the dummy vertices and edges are set to 0. We denote a mapping between vertex v_i and u_j as $v_i \mapsto u_j$. As the vertex mapping induces a map on the edges, corresponding edges are denoted as $(v_i,v_j) \mapsto (u_k,u_l)$.

A map $F : V \to U$, is called ρ-valid for $\rho \in [0,1]$ and denoted by F_ρ if it is bijective and the edge distortion of corresponding edges is bounded by ρ. As we extend both vertex sets to contain $|V|+|U|$ vertices, bijective maps can be found for all values of ρ. Given a valid map F_ρ, we measure the vertex distortion between individual maximum that have been mapped. The vertex distortion under a map F_ρ is denoted by $d^V_{F_\rho}$ with $d^V_{F_\rho}(v_i \mapsto u_j) = |\mathcal{P}(v_i) - \mathcal{P}(u_j)|$.

As the attribute associated with a dummy vertex is 0, correspondences involving a dummy vertex, i.e., $v_i \mapsto \psi_j$, implies simplification of the vertex v_i and its vertex distortion is equal to its persistence. Similarly, we denote the edge distortion by $d^E_{F_\rho}$ and it is bounded by ρ, by definition of a valid map, $d^E_{F_\rho}((v_i,v_j) \mapsto (u_k,u_l)) = |\mathcal{C}((v_i,v_j)) - \mathcal{C}((u_k,u_l))| \leq \rho$. We assume the edge distortion involving a dummy edge is 0. Figure 4 describes a valid map with $\rho = 0.25$ between two complete extremum graphs.

3.2 Distance between extremum graphs

The individual distortions of the vertices and edges is used to compute the maximum distortion of the vertex set and edge set, which is in turn used to compute a distance between the two graphs. We define a distance $D^V_F(V,U)$ between the vertex sets and $D^E_F(E_u,E_v)$

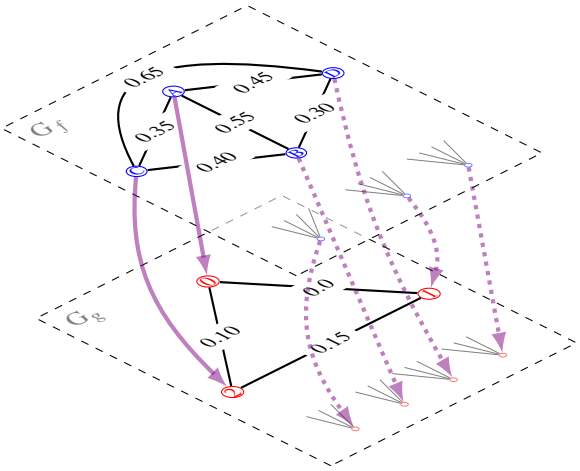

Figure 4: For $\rho = 0.25$, the figure shows a possible map between the two complete extremum graphs. Dummy vertices are inserted into each graph to obtain a bijective map. Edges are labelled with their corresponding edge cost. The correspondence $A \mapsto 0$ and $C \mapsto 2$ satisfies the ρ criterion since $|\mathcal{C}((A,C)) - \mathcal{C}((0,2))| = |0.35 - 0.10| \leq 0.25$. The correspondence $B \mapsto 1$ cannot be included since (A,B) and $(0,1)$ cannot be mapped within a distortion of 0.25 though (B,C) and $(1,2)$ satisfy the condition. Correspondences that involve a dummy vertex are indicated with dotted edges. The vertex distortion for this map can now be computed based on the difference in the vertex attributes of the corresponding vertices.

between the edge sets of the complete graphs based on the maximum distortion introduced by the map F_ρ. The distance between the vertex sets is indicative of the quality of the correspondence between the features of the functions being compared in terms of their persistence. The distance between the edge sets indicates how well the proximity relationships between pairs of extrema are preserved.

$$D^V_{F_\rho}(G_f, G_g) = max_{v \in V}\ d^V_{F_\rho}(v \mapsto F(v))$$

$$D^E_{F_\rho}(G_f, G_g) = max_{(v_i,v_j)\in E}\ d^E_{F_\rho}((v_i,v_j) \mapsto (F(v_i), F(v_j))$$

The distance between the graphs G_f and G_g is defined as the sum of the distance between their vertices and edges. Each map gives an estimate of the distance between the two graphs that measures the distortion in the graph induced by that map. For a fixed ρ, the minimum over all possible maps gives the true distance between extremum graphs.

$$D^G_{F_\rho}(G_f, G_g) = D^V_{F_\rho}(G_f, G_g) + D^E_{F_\rho}(G_f, G_g)$$

$$D_\rho(EG_f, EG_g) = D_\rho(G_f, G_g) = min\{D^G_F(G_f, G_g) | F \text{ is } \rho\text{-valid }\}$$

3.3 Composition of Maps

Let F_ρ be a ρ-valid map between complete extremum graphs $G_f(V, E_v)$ and $G_g(U, E_u)$ due to functions f and g respectively. Let G_ξ be a ξ-valid map between complete extremum graph $G_g(U, E_u)$ and $G_h(W, E_w)$ due to functions g and h respectively. Consider the composition map $H = G \circ F$ between $G_f(V, E_v)$ and $G_h(W, E_w)$. First, H is a $\rho + \xi$ valid map. Let $v_i \neq v_j$ and $v_i \mapsto w_m$ and $v_j \mapsto w_n$ under H due to maps $v_i \mapsto u_k$, $v_j \mapsto u_l$ under F_ρ and $u_k \mapsto w_m$ and $u_l \mapsto w_n$ under G_ξ. As F_ρ and G_ξ are bijective, $w_m \neq w_n$. The edge distortion introduced by H on $(v_i, v_j) \mapsto (w_m, w_n) = |\mathcal{C}((v_i, v_j)) - \mathcal{C}((w_m, w_n))|$

$$\begin{aligned}
|\mathcal{C}((v_i,v_j)) - \mathcal{C}((w_m,w_n))| &\leq |\mathcal{C}((v_i,v_j)) - \mathcal{C}((u_k,u_l)) \\
&\quad + \mathcal{C}((u_k,u_l)) - \mathcal{C}((w_m,w_n))| \\
&\leq |\mathcal{C}((v_i,v_j)) - \mathcal{C}((u_k,u_l))| \\
&\quad + |\mathcal{C}((u_k,u_l)) - \mathcal{C}((w_m,w_n))| \\
&\leq \rho + \xi
\end{aligned}$$

As edge distortions introduced by H is bounded by $\rho + \xi$, H is a $\rho + \xi$ valid map. The distortion induced by $H_{\rho+\xi}$ on the vertex v_i is bounded by the distortions induced by F_ρ on v_i and G_ξ on u_j.

$$\begin{aligned}
d^V_{H_{\rho+\xi}}(v_i \mapsto w_k) &= |\mathcal{P}(v_i) - \mathcal{P}(w_k)| \\
&= |\mathcal{P}(v_i) - \mathcal{P}(u_j) + \mathcal{P}(u_j) - \mathcal{P}(w_k)| \\
&\leq |\mathcal{P}(v_i) - \mathcal{P}(u_j)| + |\mathcal{P}(u_j) - \mathcal{P}(w_k)| \\
&= d^V_{F_\rho}(v_i \mapsto u_j) + d^V_{G_\xi}(u_j \mapsto w_k)
\end{aligned}$$

3.4 Metric Properties

We now verify that our distance measure $D_\rho(G_f, G_g)$ satisfies properties of a metric.

1. $D(G_f, G_g) \geq 0$. By definition, as all vertex and edge distortions are non-negative, this is true.

2. $D(G_f, G_g) = 0 \Leftrightarrow G_f = G_g$. We first prove the implication in the forward direction, Assuming all extrema have strictly positive persistence, to achieve a distance 0, all dummy vertices of G_f should map to dummy vertices in G_g. All non-dummy vertices should map to corresponding non-dummy vertices that have identical persistence giving zero distortion. If the edge costs are not identical between the graphs, all vertices can only be mapped under $\rho > 0$. Since the distance is 0, both the edges and vertices have identical attributes leading to $G_f = G_g$.
   The implication in the other direction follows from the fact that an identity map $F_0 : V \to V$ with $\rho = 0$ assigns a distance of 0, which is the minimum distance that can be attained. Hence, $D(G_f, G_g) = 0$.

3. $D(G_f, G_g) = D(G_g, G_f)$. By definition of the vertex and edge distances, D is a symmetric measure. For a map F that attains minimum distortion between G_f and G_g, F^{-1} attains the same distortion between G_g and G_f.

4. $D(G_f, G_g) + D(G_g, G_w) \geq D(G_f, G_w)$. From the composition of maps and bounded distortion of individual vertices and edges, the triangular inequality holds.

3.5 Computation

By definition, to compute the distance D_ρ between two complete extremum graphs, one requires a ρ-valid map between the extrema that introduces minimum distortion. One way to encode all valid maps is by considering a *product graph*. Let the two graphs to be compared be $G_f(V', E_v)$ and $G_g(U', E_u)$. We extend the vertex and edge sets as described in the map computation and denote the extended vertex sets as V and U respectively, with $|V| = |U| = |V'| + |U'|$. The product graph $P(V \times U, E)$ contains $|V| \times |U|$ vertices of the form $p\langle v_i, u_j\rangle$. Vertex p indicates a possible correspondence between its *factor* vertices v_i and u_j. The weight of a vertex is defined as $w(p\langle v_i, u_j\rangle) = 1 - d(v_i \mapsto u_j) = 1 - |\mathcal{P}(v_i) - \mathcal{P}(u_j)|$. In practice, we only require $|V'| + |U'|$ vertices in the product graph to identify vertices that will be simplified. So, inserting one dummy vertex to each extremum graph and manipulating weights and edges is sufficient for computation.

We use the validity constraints to introduce edges in the product graph. An edge exists between vertex $p_1\langle v_i, u_j\rangle$ and $p_2\langle v_k, u_l\rangle$ if $v_i \neq v_k$, $u_j \neq u_l$ and $|\mathcal{C}((v_i, v_k)) - \mathcal{C}((u_j, u_l))| \leq \rho$

Any $\rho-$valid map is now represented by a maximal clique in the product graph. The weight of a maximal clique is defined as the minimum weight of any vertex it contains. Note that since the cliques are maximal, trivial cliques are avoided. In practice, when we are interested in determining the actual correspondence as opposed to just the distance, the weight of a maximal clique can be defined as the sum of its vertex weights.

An optimum map that minimizes the distortion between G_f and G_g is now a maximum weight clique in the product graph. The vertices of the maximum clique provide the correspondences of the optimum map.

To compute the optimum correspondence map and thereby the distance $D_\rho(G_f, G_g)$, we can enumerate all maximal cliques in P and identify the maximum weight clique C_* [4]. While we use the Bron-Kerbosch algorithm to enumerate cliques, any weighted maximum clique enumeration algorithm can be used to compute the optimum map. Enumerating cliques has exponential time complexity and is feasible only for small graphs. We discuss some generic pruning strategies to sparsify product graphs as well as some approaches to partition extremum product graphs in order to reduce the search space.

3.5.1 Pruning

We prune the product graph to remove vertices and edges that cannot be a part of any maximum weight clique. The weight of any maximal clique is a lower bound on the weight of the maximum clique, W_*, in P. We compute a lower bound, W_{lb}, for W_* by computing a greedy clique. We order the vertices of P based on their weights and degrees i.e., $deg(p\langle v,u\rangle) \cdot w(p\langle v,u\rangle)$ and iteratively pick the next vertex that satisfies clique conditions.

P is a product graph and each maximal clique computes a correspondence that respects edge constraints defined by the parameter ρ. We can adapt the maximum weight bipartite matching to compute a bottleneck distance [11], that gives an optimum correspondence when no pairwise constraints are imposed by ρ. Hence, this cost, W_{ub}, on the set of factor vertices, is an upper bound for W_*.

Now, for each edge in the product graph, we compute an upper bound on the weight of a maximum clique that contains this edge. Let $N(x)$ represent the neighbours of a vertex $x\langle v,u\rangle$ in the product graph and $W(S)$ be the weight of the maximum clique restricted to a set of vertices $S \subset V \times U$. Let $W_{ub}(S)$ be the maximum bipartite matching cost between the factored vertices of S. Any maximal clique containing the edge between x and y is entirely contained in the intersection of their neighbourhoods by definition of a clique. Therefore, $W(N(x) \cap N(y)) \leq W_{ub}(N(x) \cap N(y))$.

$W(x) + W(y) + W_{ub}(N(x) \cap N(y)) < W_{lb}$ implies that the weight of any maximal clique containing the edge (x,y) does not exceed the lower bound. Therefore, pruning the edge (x,y) from P does not affect the maximum weight clique. Once edges have been pruned, vertices can also be pruned in a similar fashion. A vertex $x\langle v,u\rangle$ is pruned if $W(x) + W_{ub}(N(x)) < W_{lb}$.

Further, based on the application, vertices and edges can be additionally pruned using geometric information and constraints. An edge in the product graph refers to a pair of vertices in each complete extremum graph. If these vertices are expected to maintain similar geodesic or euclidean distance between them, the edges of the product graph may be pruned based on such properties.

3.5.2 Partitioning

While pruning helps in reducing the number of edges and vertices in the product graph, direct construction of the maximum clique remains impractical. We leverage the fact that the importance of a feature is captured by the persistence of maxima, to partition the product graph and process the partitions in an importance-aware manner. The maximum weight clique is constructed incrementally by restricting the computation to a growing subset of the product graph.

We order the factored vertices in the increasing order of their persistence values, $\mathcal{P}$. To identify threshold values that denote a relatively significant change in persistence between vertices, we plot the difference between the persistence of the ordered vertices, $\mathcal{P}(i+1) - \mathcal{P}(i)$. A peak in this difference plot indicates a significant change in the persistence values and we identify the persistence associated with these points as thresholds. We restrict thresholds to have a minimum separation of 1% of the function range.

After identifying a set of thresholds $T\{t_0 > t_1 > \cdots > t_n\}$, we partition the vertex set of the product graph based on it. We denote the partially computed maximum clique as a set of vertices C'. In each iteration i, we consider only those vertices $p\langle v_i, u_j\rangle$, whose minimum persistence value $min\{\mathcal{P}(v_i), \mathcal{P}(u_j)\}$ is at least t_i. We consider the graph induced by these vertices and enumerate the maximum clique for it. Vertices that indicate a correspondence between non-dummy vertices are added to C'. We then prune the product graph by eliminating vertices that are not adjacent to all the vertices in C'. Vertices that indicate a dummy correspondence are included in the next iteration along with vertices that satisfy the next threshold. In the last iteration, all dummy correspondences are also included in C'.

4 Applications

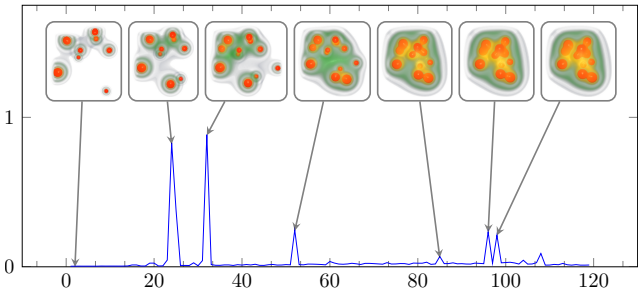

Figure 5: The plot shows distances computed between consecutive time steps for a synthetic time-varying dataset. The distance plot helps in summarizing the dataset, peaks in the plot indicate frames that have a large distance with respect to the previous time step, indicating an event of importance. The peak at time step 24 occurs due to the creation of a new feature in the bottom right.

Time varying data is often available for a large time period due to the nature of the physical phenomenon or a high time resolution simulation. Visualizing the data frame by frame is tedious. Distance measures can be used to provide an overview of the entire sequence of data. This can help in identifying frames of significant activity, series of time steps that are stable and other interesting patterns such as periodicity in the data. Further, correspondence based distance measures also facilitate tracking features across time steps. In time varying scalar fields, these patterns and variations are often captured by the changes in its topological abstractions across time steps. Figure 5 shows the distance plot computed between consecutive time steps of synthetic dataset. Peaks in the plot refer to a pair of time steps that show significantly higher distance, they indicate time steps that involve creation or merger of features. Peaks at time step 24, 32 and, 54 appear due to creation of new extrema. Over time, these features merge and this event can be seen as peaks at time step 96 and 98. We now discuss some applications of our distance measure to a few time varying datasets that validate and illustrate its usefulness.

4.1 Periodicity in Time-Varying Data

Identification of periodicity in a time-dependent data is critical to understanding the underlying phenomenon and validating simulations. Figure 6 shows a few time steps of the Bénard–von Kármán vortex street formed by a flow around a cylinder simulation[1] that exhibits periodic vortex shedding [33]. To verify that our distance function indeed identifies the known periodicity in this data [23], we compare each time step with all other time steps of the dataset.

[1]This data set has been simulated by Tino Weinkauf [32] using the Free Software *Gerris Flow Solver* [22] and is available from http://people.mpi-inf.mpg.de/ weinkauf/notes/cylinder2d.html.

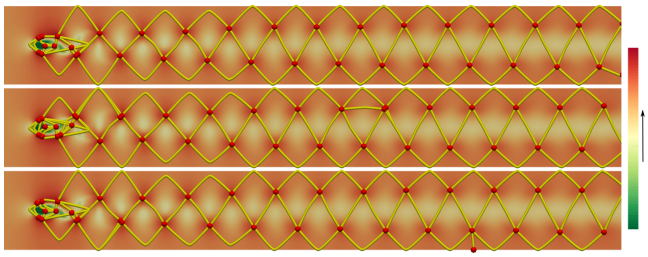

Figure 6: Time-step 0 (top), 38 (middle) and 75 (bottom) of the flow around a cylinder simulation. This simulation is time dependent with a time period of 75, these time steps appear similar. The vortex shedding alternates between the two sides of cylinder with a time period of 38. Time step 38 appears symmetric to time step 0 and 75. The extremum graph is overlayed in yellow.

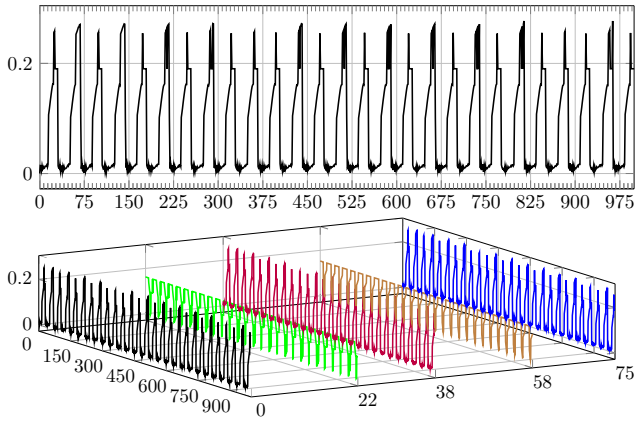

Figure 7: To identify periodicity, we compare the extremum graph of each time step with all 1000 time steps of the data. The line plot above shows distances computed between time step 0 and time steps 0-1000. The time steps are indicated on the x-axis and distance is indicated on the y-axis. The plot below shows distances computed with respect to time step 22, 38, 58 and 75. From both plots, a time period of 38 can be identified.

To ensure that edge constraints are strictly enforced, we use a low $\rho = 0.001$ across all comparisons. Figure 7 shows the vertex distortion plots for a few time steps. Each time step is compared against all time steps. We can observe from the distance plot that the simulation is periodic. The time period of this simulation is known to be 75, the plot however indicates a time period of 38. This is due to the alternating nature of vortex shedding from the two sides of the cylinder as shown in Figure 6. Since the distance measure is over the extremum graph, the symmetric nature of these oscillations are identified.

4.2 Correspondence and Tracking of Features

When data is time-varying, one is often interested in tracking features identified in one time-step, across all other time-steps. The ability to perform such tracking greatly aids in visualizing the features of a dataset. In this example, we show that the correspondences found based on the complete extremum graph are intuitive and use them to track features across time steps. We compute correspondences between adjacent time steps and propagate the correspondence by transitivity across all time steps to uniquely identify a feature. The complete graph structure allows us to compare features irrespective of instabilities in the merging order of the extrema or geometric overlap in the features. Figure 8 shows the first few time steps of the turbulent vortex flow data[2]. In order to show the correspondence in the turbulent vortex flow, we compute iso-surfaces

[2]http://vis.cs.ucdavis.edu/TVDR/Vortex/

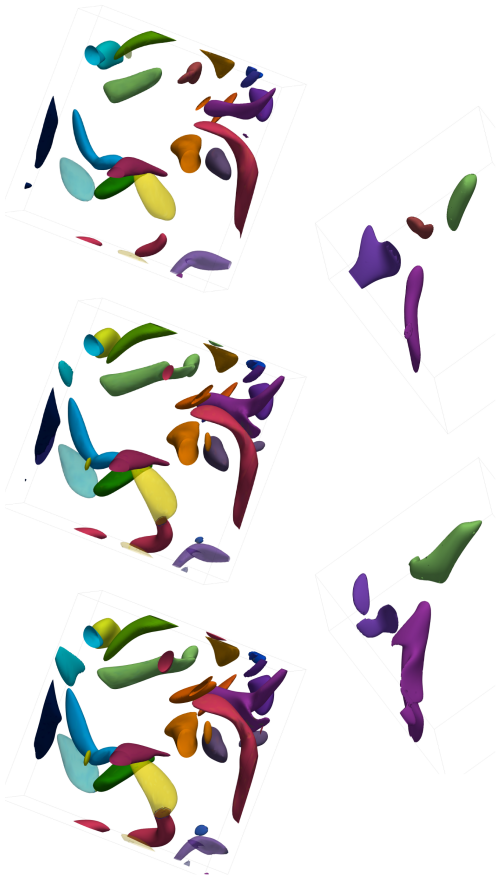

Figure 8: To track features in the turbulent vortex data, we compare the complete extremum graphs of consecutive time steps. Tracked features across time steps 2, 4 and 6 are shown on the left. Low opacity values indicate higher structural distortion in the complete extremum graph. Three features are shown in isolation on the right, the violet feature undergoes a split. The green and brown features merge, the purple feature grows.

of the volume at an iso-value that is 50% of the maximum vorticity magnitude in each dataset [26]. The correspondence established between the descending manifolds is mapped to the parts of the iso-surface that lie in that descending manifold and is indicated by its color. The complete extremum graph implicitly handles merging and splitting of the extrema features. Figure 8 (right) shows three features in isolation from time steps 2 and 7. The violet feature splits. The green and brown features merge and the purple feature grows. To label a feature that corresponded to a dummy vertex, as in the case when a feature splits, we adopt the label of its parent feature according to the persistence based simplification order. As the goal here is to compute correspondences and track as many features possible, we compute correspondences at various ρ levels. We compute a set of ρ values based on the peaks in the difference plot of the edge costs. We map the lowest ρ value at which a feature is mapped inversely to its opacity, indicating the extent to which the graph structure with respect to that feature is similar across time steps. Note, the violet feature in the bottom and the yellow feature in the center of the tracked frames in Figure 8, these features undergo merges and splits and their transparency indicates that the variation in their edge structure is relatively higher as compared to other features. To speed up clique computation, we consider only vertices with persistence $> 10\%$ of the function range and also prune them based on the moments of the descending manifold [15].

5 Conclusion

We presented a distance measure to compute the similarity between scalar fields and showcased its applicability in understanding time varying datasets. We introduced a complete extremum graph structure that captures proximity information between all pairs of extrema and computed maximum weight common subgraphs to determine the similarity between fields. Graph based distances are typically hard to compute and have exponential complexity in the worst case, we discussed pruning and partitioning strategies to effectively compute the distance measure. However, the distance computation is not real time. Other strategies and heuristics based on the extremum graph structure to improve computational aspects can be further worked upon.We used application specific schemes to identify ρ values, a general approach to efficiently compute optimal values needs further investigation. The complete extremum graph structure provides a way of relating all pairs of extrema based on function perturbation. It would be interesting to see if this information can be effectively employed in other similarity measures that are easier to compute, such as shape distribution [20], to understand scalar fields. Finally, while the complete extremum graph is designed to mitigate the effects of instabilities due to binary choices that are inherent in simplification based approaches, we intend to perform a thorough analysis of the stability of this distance function in the future.

Acknowledgements

This work was partially supported by the Department of Science and Technology, India, under Grant SR/S3/EECE/0086/2012. Vijay Natarajan was supported by a fellowship for experienced researchers from the Alexander von Humboldt Foundation and by the Robert Bosch Centre for Cyber Physical Systems, Indian Institute of Science.

References

[1] U. Bauer, X. Ge, and Y. Wang. Measuring distance between Reeb graphs. In *SOCG'14*, page 464, 2014.

[2] K. Beketayev, D. Yeliussizov, D. Morozov, G. . H. Weber, and B. Hamann. Measuring the distance between merge trees. *Topological Methods in Data Analysis and Visualization III, Mathematics and Visualization. Springer-Verlag*, 2013.

[3] P.-T. Bremer, B. Hamann, H. Edelsbrunner, and V. Pascucci. A topological hierarchy for functions on triangulated surfaces. *Visualization and Computer Graphics, IEEE Transactions on*, 10(4):385–396, 2004.

[4] C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, Sept. 1973.

[5] S. Bruckner and T. Möller. Isosurface similarity maps. *Comput. Graph. Forum*, 29(3):773–782, 2010.

[6] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.

[7] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern recognition letters*, 19(3):255–259, 1998.

[8] G. Carlsson, A. Zomorodian, A. Collins, and L. Guibas. Persistence barcodes for shapes. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 124–135. ACM, 2004.

[9] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

[10] C. Correa, P. Lindstrom, and P.-T. Bremer. Topological spines: a structure-preserving visual representation of scalar fields. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):1842–1851, 2011.

[11] H. Edelsbrunner and J. Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

[12] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical morse complexes for piecewise linear 2-manifolds. In *Proceedings of the seventeenth annual symposium on Computational geometry*, pages 70–79. ACM, 2001.

[13] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete and Computational Geometry*, 28(4):511–533, 2002.

[14] H. Edelsbrunner, D. Morozov, and V. Pascucci. Persistence-sensitive simplification functions on 2-manifolds. In *Proceedings of the Twenty-second Annual Symposium on Computational Geometry*, SCG '06, pages 127–134, New York, NY, USA, 2006. ACM.

[15] J. Flusser, B. Zitova, and T. Suk. *Moments and moment invariants in pattern recognition*. John Wiley & Sons, 2009.

[16] M. Haidacher, S. Bruckner, and M. E. Gröller. Volume analysis using multimodal surface similarity. *IEEE Trans. Vis. Comput. Graph.*, 17(12):1969–1978, Oct. 2011.

[17] M. Hilaga, Y. Shinagawa, T. Komura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *SIGGRAPH*, pages 203–212, 2001.

[18] D. Morozov, K. Beketayev, and G. Weber. Interleaving distance between merge trees. *Discrete and Computational Geometry*, 49:22–45, 2013.

[19] S. Nagaraj, V. Natarajan, and R. S. Nanjundiah. A gradient-based comparison measure for visual analysis of multifield data. *Comput. Graph. Forum*, 30(3):1101–1110, 2011.

[20] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *ACM Transactions on Graphics (TOG)*, 21(4):807–832, 2002.

[21] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. The toporrery: computation and presentation of multi-resolution topology. In *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, pages 19–40. Springer, 2009.

[22] S. Popinet. Free computational fluid dynamics. *ClusterWorld*, 2(6), 2004.

[23] H. Saikia, H.-P. Seidel, and T. Weinkauf. Extended branch decomposition graphs: Structural comparison of scalar data. *Computer Graphics Forum (Proc. EuroVis)*, 33(3):41–50, June 2014.

[24] D. Schneider, C. Heine, H. Carr, and G. Scheuermann. Interactive comparison of multifield scalar data based on largest contours. *Computer Aided Geometric Design*, 30(6):521–528, 2013.

[25] D. Schneider, A. Wiebel, H. Carr, M. Hlawitschka, and G. Scheuermann. Interactive comparison of scalar fields based on largest contours with applications to flow visualization. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1475–1482, 2008.

[26] D. Silver and X. Wang. Volume tracking. In *Visualization'96. Proceedings.*, pages 157–164. IEEE, 1996.

[27] S. Takahashi, Y. Takeshima, G. Nielson, and I. Fujishiro. Topological volume skeletonization using adaptive tetrahedralization. In *Geometric Modeling and Processing, 2004. Proceedings*, pages 227–236. IEEE, 2004.

[28] D. Thomas and V. Natarajan. Multiscale symmetry detection in scalar fields by clustering contours. *IEEE Transactions on Visualization and Computer Graphics*, 99(PrePrints):1, 2014.

[29] D. M. Thomas and V. Natarajan. Symmetry in scalar field topology. *IEEE Trans. Vis. Comput. Graph.*, 17(12):2035–2044, 2011.

[30] D. M. Thomas and V. Natarajan. Detecting symmetry in scalar fields using augmented extremum graphs. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2663–2672, 2013.

[31] D. M. Thomas and V. Natarajan. Detecting symmetry in scalar fields using augmented extremum graphs. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2663–2672, 2013.

[32] T. Weinkauf and H. Theisel. Streak lines as tangent curves of a derived vector field. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization 2010)*, 16(6):1225–1234, November - December 2010.

[33] H.-Q. Zhang, U. Fey, B. R. Noack, M. König, and H. Eckelmann. On the transition of the cylinder wake. *Physics of Fluids (1994-present)*, 7(4):779–794, 1995.

[34] X. Zhang, C. L. Bajaj, B. Kwon, T. J. Dolinsky, J. E. Nielsen, and N. A. Baker. Application of new multi-resolution methods for the comparison of biomolecular electrostatic properties in the absence of global structural similarity. *SIAM J. Multiscale Modeling and Simulation*, 5:1196–1213, 2006.