Similarity between scalar fields

A THESIS

SUBMITTED FOR THE DEGREE OF

Master of Science (Engineering)

IN FACULTY OF ENGINEERING

by

Vidya Narayanan

Supercomputer Education and Research Centre

Indian Institute of Science

BANGALORE – 560 012

FEBRUARY 2016

TO

My Family

Acknowledgements

I thank my advisors Prof. Vijay Natarajan and Prof. Matthew Jacob for their support and guidance. I have learnt immensely from Vijay, his patience, clarity and encouragement, even throughout his sabbatical, has been a great help. I thank Prof. Matthew for his help especially during Vijay's sabbatical. I had to take considerable time off academics during my first year at IISc for personal reasons, I am grateful to Vijay for his support. I also thank the chairman, Prof. R Govindarajan for his support and the excellent research environment at SERC. I enjoyed the coursework at IISc immensely. It was also a pleasure to audit a number of interesting courses at various departments. I thank Prof.Vijay, Prof. Sathish Govindrajan, Prof. Vittal Rao, Prof. Basudeb Datta, Prof. Siddharth Gadgil, Prof. Chiranjib Bhattacharya, Prof. Ramesh Hariharan and Prof. Ravi Kannan for their well designed courses at IISc. The staff at SERC have always been helpful and efficient and deserve special mention. Discussing and collaborating with my labmates - Aditya, Anurag, Dilip, Nitin, Preethi and Talha has always been interesting. I thank my family for all the love - Patti, Ammamma, Papaji, Amma, Sri, Ashu, and Siddharth kutty. I thank my friends - Lipika, Purvi, Sahana and Vaivaswatha for the good times. I thank Ravi, for being the person he is.

Publications based on this Thesis

1. Vidya Narayanan, Dilip Mathew Thomas and Vijay Natarajan; *Distance between Extremum Graphs.* In the proceedings of The IEEE Pacific Visualization Symposium (PacificVis 2015).

Abstract

Scientific phenomena are often studied through collections of related scalar fields such as data generated by simulation experiments that are parameter or time dependent . Exploration of such data requires robust measures to compare them in a feature aware and intuitive manner.

Topological data analysis is a growing area that has had success in analyzing and visualizing scalar fields in a feature aware manner based on the topological features. Various data structures such as contour and merge trees, Morse-Smale complexes and extremum graphs have been developed to study scalar fields. The extremum graph is a topological data structure based on either the maxima or the minima of a scalar field. It preserves local geometrical structure by maintaining relative locations of extrema and their neighborhoods. It provides a suitable abstraction to study a collection of datasets where features are expressed by descending or ascending manifolds and their proximity is of importance.

In this thesis, we design a measure to understand the similarity between scalar fields based on the extremum graph abstraction. We propose a topological structure called the complete extremum graph and define a distance measure on it that compares scalar fields in a feature-aware manner. We design an algorithm for computing the distance and show its applications in analyzing time varying data such as understanding periodicity, feature correspondence and tracking, and identifying key frames.

Contents

List of Figures

Keywords

Scalar fields, topological data analysis, extremum graphs, similarity, distance measures, maximum cliques

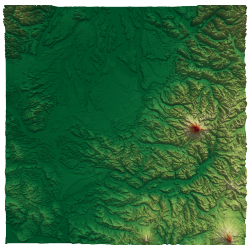
Chapter 1

Introduction

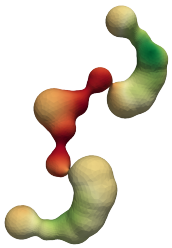
Many scientific experiments and simulations measure physical quantities over a domain of interest. The data thus generated is commonly represented as scalar fields. For example, medical images generated from X-ray CT or MRI imaging techniques are scalar fields, simulations performed for computational fluid dynamics or climate modeling can also be viewed as scalar fields. Some examples of scalar fields are shown in Figure 1.1.

Prior research focussed extensively in developing methods to explore scalar field data and to identify important features in the data. The features present in a scalar field play a key role in understanding the properties of the scientific phenomenon being studied. Topological data analysis attempts to understand scalar fields based on its critical points, level sets and their connectivity. Based on these topological features, a number of data structures such as contour and merge trees, Morse-Smale complexes and extremum graphs have been developed to abstract and analyze scalar fields. These representations of the scalar field have been quite successful in the exploration and visualization of scalar fields in a feature-aware manner.

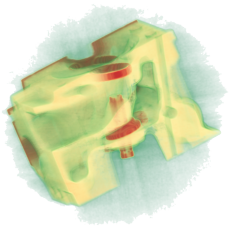
In many instances, data is generated as a collection of closely related scalar field datasets. Time varying data and ensemble data are often generated by scientists to better understand the underlying scientific phenomenon of interest. For example, scientists may generate multiple fields by altering simulation parameters to understand the nature of a phenomenon. Often, the phenomenon being studied is inherently time varying such as climate data generated over multiple hours and days. Newer paradigms are needed to analyze such datasets and an important



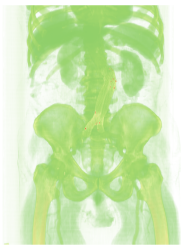
(a) A terrain represented as a 2D scalar field, the color map represents the height.



(b) The electrostatic potential field mapped to the surface of Gramicidin A molecule channel



(c) A 3D scalar field from an industrial CT scan.



(d) A 3D scalar field from an XRAY-CT scan of a human body.

(e) Color map.

Figure 1.1: A few examples of scalar fields (a-d) and the color map (e). Orange indicates high scalar values and green indicates low values. All scalar field images in this thesis use this color map.

step in this direction is to develop robust methods to compare similar scalar fields. Further, it is pertinent to design similarity measures for comparing scalar fields in a feature-aware manner. Hence, it is natural to examine similarity between scalar fields based on the similarity between their topological abstractions.

1.1 Distance between Extremum Graphs

In this thesis, we study the problem of quantifying the similarity between scalar fields by comparing their extremum graphs. More precisely, we compute a distance measure between the extremum graphs of scalar fields.

The extremum graph is a topological data structure that captures the proximity between the extrema in the scalar field [15]. It preserves local geometrical structure by maintaining relative locations of extrema and their neighborhoods. It provides a suitable abstraction to study a collection of datasets where features are expressed by descending or ascending manifolds and their proximity is of importance.

In order to compare extremum graphs, we propose a topological structure called the complete extremum graph and define a feature-aware distance measure on it. We design an algorithm for computing the distance and show its applications in analyzing time varying data such

as understanding periodicity, feature correspondence and tracking, and identifying key frames.

1.2 Contributions

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

- 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.

1.3 Organization

This thesis is organized as follows. We discuss concepts from topological data analysis necessary to understand related work in the area and the proposed distance measure, in Chapter 2. In Chapter 3, we discuss related work in the area. Various distance and similarity measures have been introduced in the literature, we discuss these measures briefly and motivate the need for a proximity-aware measure based on extremum graphs. We propose the topological structure called complete extremum graphs and its computation in Chapter 4. In Chapter 5, we formulate the proposed distance measure for complete extremum graphs and discuss its properties. Applications of our distance measure in analyzing time varying datasets is discussed

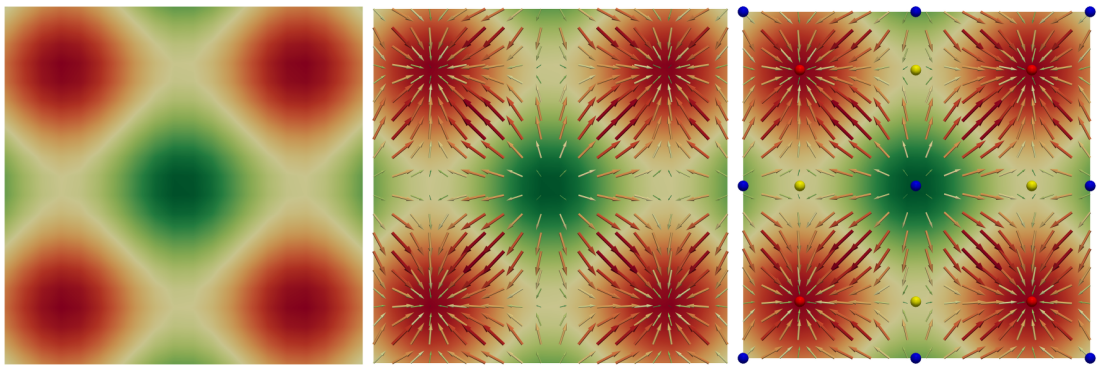
in Chapter 6. We conclude our discussion with promising directions and improvements in Chapter 7.

Chapter 2

Background

In this chapter we introduce the necessary background to define and explain distance measures on topological abstractions. Primarily, we define topological features such as critical points and level sets, and topological abstractions for scalar fields such as persistence diagrams, Reebgraphs, merge trees and extremum graphs.

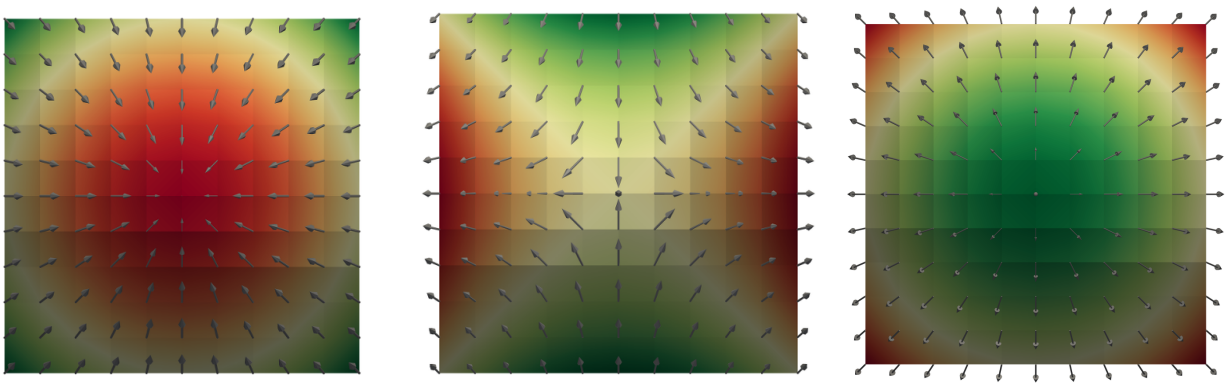
Let $f : \mathrm{D} \to \mathbb{R}$ be a scalar function defined on a smooth n-dimensional manifold, D. For $x \in \mathrm{D}$, if the gradient $\nabla f(x) = 0$, x is called a **critical point** of f on D. All other points are called **regular**.



(a) $f(x,y) = sin(x) + sin(y)$ (b) $\nabla f(x,y)$ (c) maxima(red), saddles(yellow) and minima(blue) of f

Figure 2.1: Critical points of $f(x,y) = sin(x) + sin(y)$

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



(a) maximum, index=2 (b) saddle, index=1 (c) minimum, index=0

Figure 2.2: Critical points and their indices.

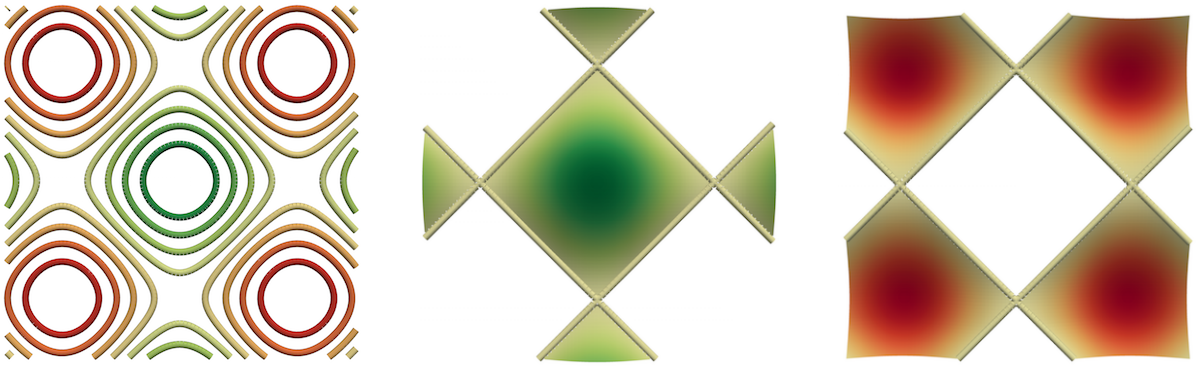
classified using an **index** which equals the number of independent directions along which f decreases. The index of a **minimum** is 0 and of a **maximum** is n. **Saddles** have indices from $n-1$ to 1.

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** [19]. Figure 2.6 shows the integral curves of the gradient field for a simple function and its descending manifolds.

The **level set** of a function $f : D \to \mathbb{R}$ takes the form $L(c) = \{x \in D \mid f(x) = c\}$. Connected components of a level set are called iso-contours. When D is a two dimensional manifold, these are curves. In three dimensions, connected components of the level sets are called isosurfaces. Sublevel sets take the form $L_{-}(c) = \{x \in D \mid f(x) \leq c\}$ and superlevel sets are defined as $L_{+}(c) = \{x \in D \mid f(x) \geq c\}$. Figure 2.3 illustrates level sets, sublevel sets and superlevel sets for the scalar function $f : [0,3\pi] \times [0,3\pi] \to \mathbb{R}$, $f(x,y) = sin(x) + sin(y)$.

Features in topological analysis typically refer to critical points, their descending or ascending manifolds and level sets. By associating a notion of importance with features, they can be analyzed in a hierarchical or multi-scale fashion.

Integral curves begin and terminate at critical points. Such critical points pairs, whose index differs by 1, can be **cancelled** and eliminated from the scalar field. Cancellations gives a mechanism to simplify the scalar field by reducing the number of features or critical points.



(a) $L(\{\ldots,-0.1,0.1,\ldots\})$ (b) $L_{-}(0.0)$ (c) $L_{+}(0.0)$

Figure 2.3: Level sets, sublevel sets and superlevel sets of $f(x,y) = sin(x) + sin(y)$

Figure 2.4 shows an example of a cancellation. Here, a maximum i.e., a critical point with index two, is cancelled with a saddle of index one. To realize the cancellation of a pair, the scalar function should be modified. The perturbation necessary is given by the difference in the scalar values of the pair of points. In terms of the integral curves of the gradient vector field, the flow is *reversed* along the critical point pair, giving a **simplified** field. Each simplification modifies the available pairs for surviving critical points. The maximum perturbation necessary to simplify a set of critical points depends on the order in which they are simplified. To optimally simplify a function f, with respect to the L_∞ norm, critical point pairs can be ordered according to the function difference their removal demands and simplified in order. The difference associated with a pair of critical points provides a measure of importance in terms of the local perturbation necessary to eliminate the pair and simplify the function. In two dimensions, these pairs are equivalent to the homological persistent pairs. Persistence gives a measure of importance to the topological features in the scalar field [20, 21]. In higher dimensions, it may not be possible to cancel all homological persistence pairs. However, we continue to refer to the function difference between a pair of critical points at the time of its cancellation as its **persistence**.

Cancellations can be done based on other criteria as well, as long as the cancellation is *valid* i.e., the gradient field after reversal remains acyclic.

Various topological abstractions have been derived based on these features and their relationships such as persistence diagrams, contour trees, Morse-Smale complexes and extremum

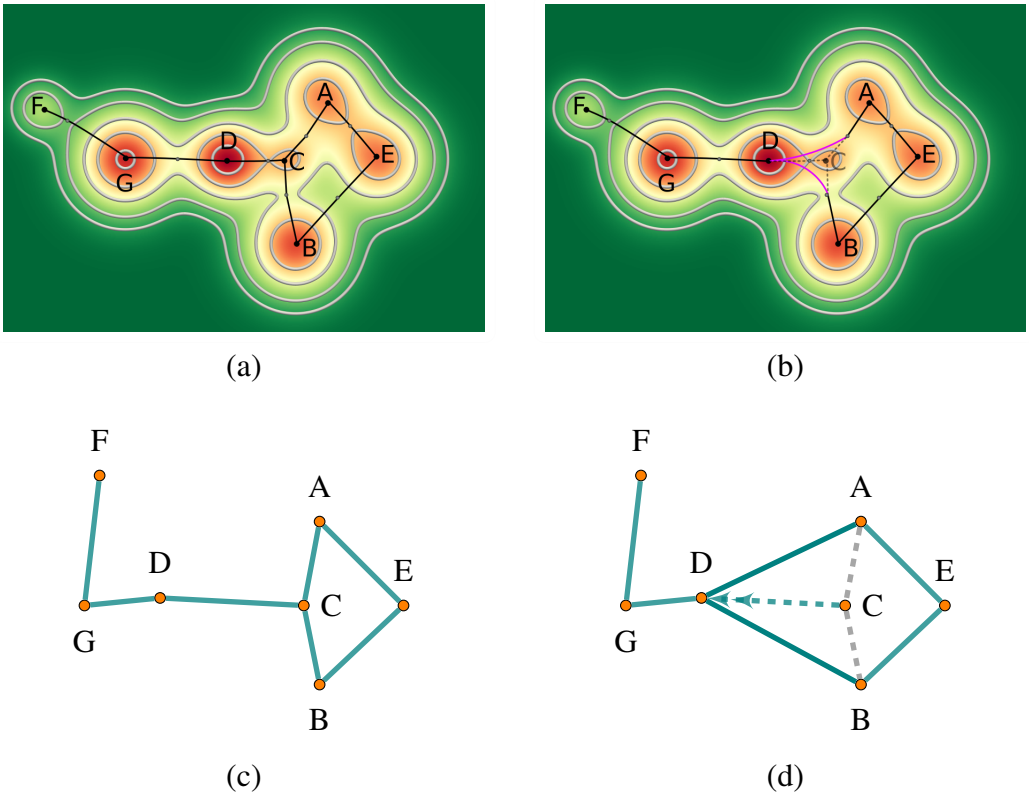
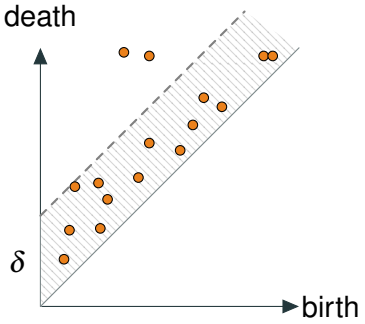
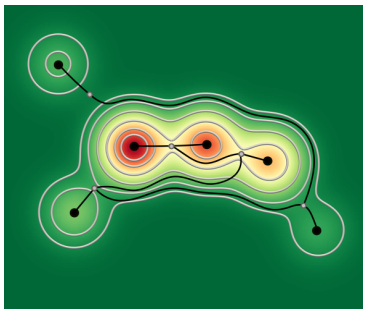


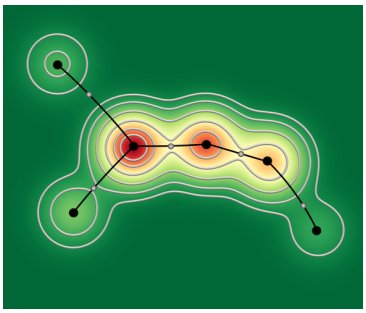
Figure 2.4: Extremum graphs provide an abstraction of scalar fields and encodes adjacency relationships between the extrema of the field. (a) shows a 2D scalar function overlayed with its extremum graph. (b) The extremum C can be simplified by a merger into extremum D. (c) shows a combinatorial representation of the extremum graph. (d) Simplification of vertex C into vertex D involves contraction of edge (C,D).



(a) Persistence Diagram



(b) A scalar field with its contour tree overlayed



(c) A scalar field with its extremum graph overlayed

Figure 2.5: Topological abstractions

graphs. We now discuss a few of them.

Persistence Diagram. A persistence diagram encodes the persistence of features as points in a plane [14]. It gives an overview of features-vs-noise in the scalar field, allowing the user to reason about the amount of noise and important features. Though it gives a high level overview of the critical points and their importance it does give any information about the relationships between the critical points such as their proximity. Figure 2.5(a) shows an example of a persistence diagram. The distance of a point from the diagonal gives a measure of its importance. All points in the diagram that fall in the shaded region can be simplified within a perturbation threshold of δ.

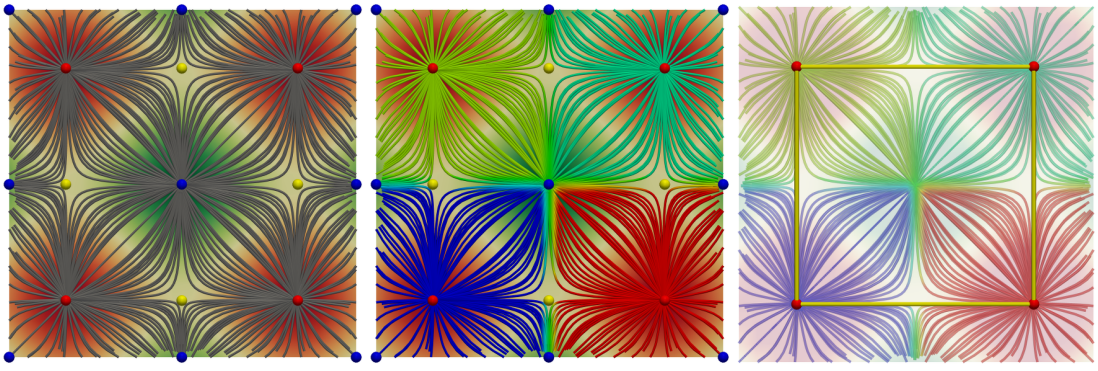
Contour tree / Reeb graph. The Reeb graph of a scalar field is a topological data structure that tracks the changes in the number of connected components of its level sets [17, 32]. If the domain is simply connected, the graph contains no loops and is called the contour tree. Contour trees and Reeb graphs track the nesting structure of sub level sets and super level sets in the scalar field. The merge tree and split tree are subsets of the contour tree that only tracking the merging or splitting of sublevel sets respectively [11]. These structures have been used for data analysis [1, 12] and designing transfer functions [45], in addition to other applications. As seen in Figure 2.5 (b), this structure can relate geometrically near by points in a non-intuitive manner.

Extremum graph. An extremum graph gives a proximity-aware description of the scalar

field. The maximum graph relates the maxima and $n-1$ saddles based on adjacency. Analogously, the minimum graph relates the minima and 1-saddles. For many datasets, often only one type of extrema and their ascending or descending manifolds represent the features. For such datasets, the extremum graph is a useful abstraction tool. Figure 2.5 (c) shows the extremum graph for the overlayed on the scalar field. It provides an intuitive and proximity aware abstraction.

Correa et al. [15] 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. 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 in this thesis. Figure 2.6 shows the maximum graph for a simple function.

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).



(a) integral curves of ∇f (b) descending manifolds (c) extremum graph

Figure 2.6: Morse decomposition of $f(x,y) = sin(x) + sin(y)$

In all these data structures, simplification based on persistence or other criteria is often employed to remove noise and generate hierarchical representations [4, 15, 30]. Figure 2.4

illustrates simplification of an edge in the extremum graph.

Chapter 3

Related Work

Similarity detection is a well studied area in pattern recognition, geometry, graphics and computer vision. Measures derived to compute (dis)similarity may satisfy metric properties, in which case they are called distance measures. Other measures that do not satisfy all metric properties may still be useful in discerning dissimilarity, such as the cosine similarity measure for vectors. In this section, we discuss related work that compare scalar fields.

3.1 Geometry and Statistics

Many approaches assume a dense correspondence between the domains of the fields that are to be compared. Once such a correspondence or alignment is established, the scalar field can be compared. The simplest approach assumes that the domains are identical. The root mean square distance, Chebyshev distance and other L_p metrics compare the scalar fields in a point to point fashion.

Many statistical approaches to image comparison in the area of computer vision consider the image as a gray-scale scalar field or multi valued color field. Histograms are often employed as a signature and various bin-to-bin or cross-bin measures have been used for comparing them [13, 33]. Another popular feature descriptor is based on the maximal stable extremal region [16, 26]. A component tree of the image is constructed and stable intervals are identified to extract important regions in an image. Image moments are then used to describe these

features and compare them [22]. Moment based approaches have also been used to compare vector fields [7].

Topological analysis attempts to understand the domain of the function based on the scalar field defined on it by analyzing level sets, segmentations based on gradients and so on. Manifold harmonics, a generalization of fourier analysis to manifolds, attempts to understand the scalar function based on the domain over which it is defined [41]. They decompose the defined function by using the eigen-functions of the laplace-beltrami operator as basis. The spectra of the laplace- beltrami operator and its eigenfunctions have been heavily used in shape analysis. Scalar fields defined over a 3D volume are common, especially in the form of simulation and imaging datasets. Techniques that have been well studied in geometry and graphics can be applied to iso-surfaces of these datasets. Thomas et al. [38] cluster isosurfaces based on properties of the isosurface shape. They use a shape signature based on the laplace-beltrami spectra and cluster contours in the signature space.

Information theoretic approaches have been proposed to identify important iso-values that provide non-redundant information [6, 23] within the same or multiple scalar fields defined on the same domain. Here isosurfaces are represented using their distance transform and mutual information is used to identify these values.

The correlation between different functions which are not necessarily similar but defined on the same domain have also been studied for exploring multi-field data [28, 35, 36].

3.2 Topological Data Analysis

We now discuss topology based distance measures and similarity measures that are closely related to our proposed measure. We motivate the need for feature-aware and proximity sensitive distance measures by identifying examples where existing measures may fail to provide an intuitive measure of similarity or correspondence.

3.2.1 Distance Measures

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 scalar fields indirectly.

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 supremum distance between a point and its image under L_∞ norm is considered for a given mapping between the points. The bottleneck distance considers all possible mappings and is the infimum among all such supremums. It is known that the persistence diagram is stable under perturbations of the underlying function and therefore a large bottleneck distance implies that the underlying functions are also dissimilar [14]. 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 [10]. The barcode metric is used to compare barcodes and 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 adjacency constraints are imposed on the points in the persistence diagram. This limits their use for comparing scalar fields.

Recent years have seen distance measures defined on Reeb graphs and its simpler variant called the merge tree. Morzov et al. defines 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 [27]. 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 [3]. 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 functional distortion distance between two Reeb graphs by considering two continuous functions that

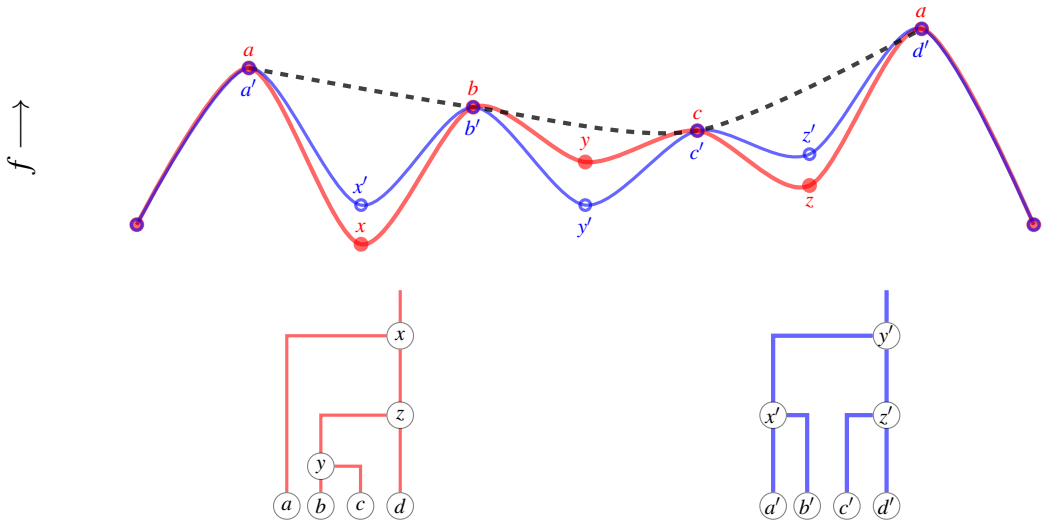


Figure 3.1: A merge tree, and its superset, the contour tree, captures the nesting structure of level sets. For the two functions shown above (blue and red), no branch decomposition of the merge tree (below) reflects the correspondence between nodes b to b' and c to c'. The extremum graph (dashed) captures proximity between the extrema and can provide a more intuitive correspondence between them.

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 [2]. Figure 3.1 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 [3, 27], the extrema b, c and b', c' will be simplified rather than matched. On the other hand, extremum graph based measures are able to associate these extrema and their descending manifolds intuitively because they are based on proximity.

3.2.2 Similarity Measures

While the above methods define distance between topological structures in a rigorous manner, several methods use simpler notions of similarity measures for identifying 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 have been used for matching shapes [24, 44] and identifying repeating features [39]. Saikia et al. recently introduced the extended branch decomposition graphs as an efficient data structure to encode branch decompositions

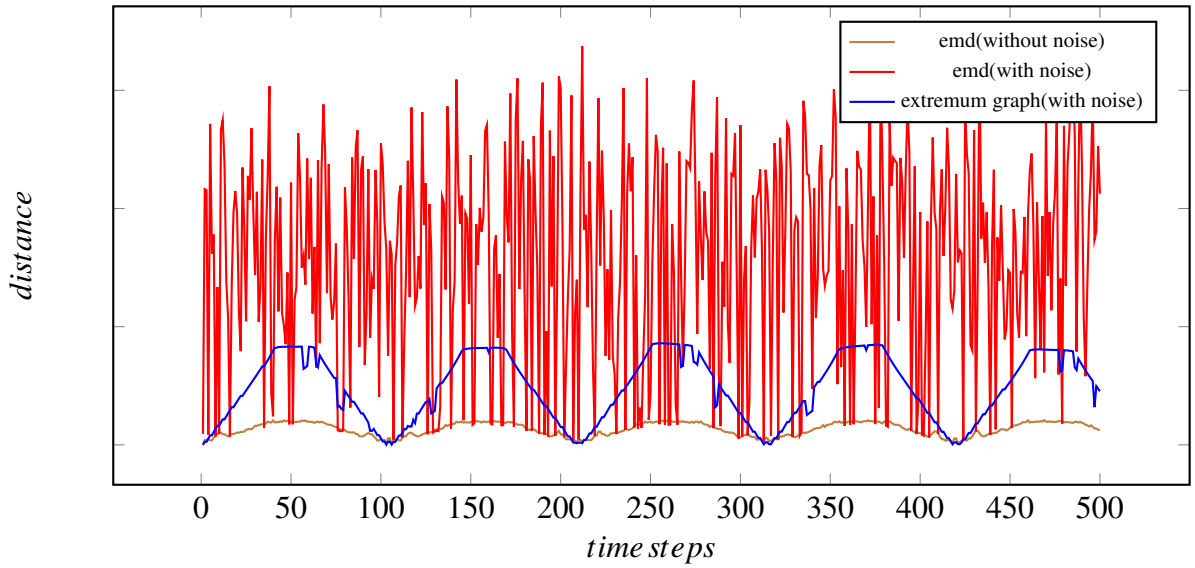


Figure 3.2: Comparing time step 0 of a synthetic dataset with time steps 0-500. Periodic pattern can be identified from the earth movers distance as well as the extremum graph based distance. However, with low persistence noise added, the earth movers distance fails to identify the periodic pattern, the extremum graph based distance which is feature-aware, continues to identify the pattern.

of all subtrees of the merge tree and use this to identify repeating features and periodicity in time varying data through a matching procedure based on dynamic programming [34]. 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 [40].

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. 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 [8, 9] 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. Figure 3.2 compares distances computed between time steps of a scalar

potential field derived from a 3-body choreography [25]. The earth movers distance, which is based on histogram comparisons fails to detect the periodic pattern when noise is added. Since our metric is based on the topological features, we discern patterns in the field even when low persistence noise is added.

Chapter 4

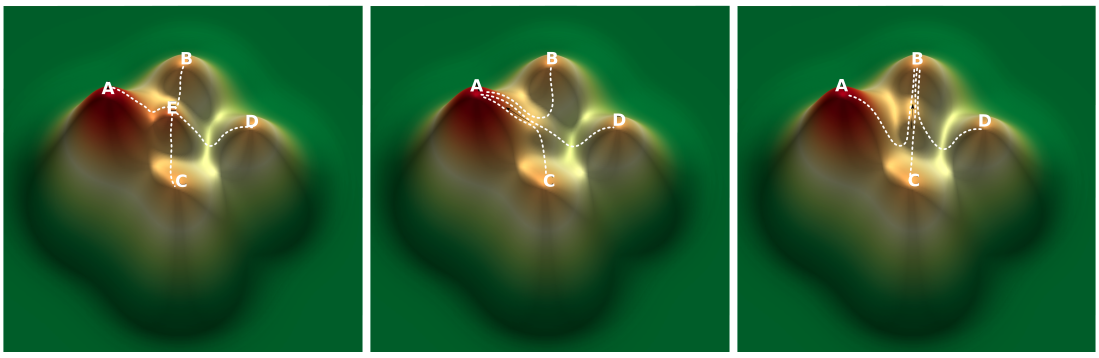
Complete Extremum Graphs

When comparing multiple scalar fields based on their extremum graphs, we have to account for difference in the number of extrema and their neighbourhood relationships. Towards this, in this chapter, we discuss a topological data structure called the *complete extremum graph* and introduce an algorithm to compute it.

The complete extremum graph captures pairwise relationships between pairs of extrema in the extremum graph and records it in the form of a complete graph. Given a pair of extrema, we associate with it, the local perturbation necessary such that the descending manifolds represented by these extrema become adjacent i.e., to introduce a shared saddle between them. We also associate the saliency or importance of each extrema in terms of the local perturbation necessary to eliminate it by suitably modifying the scalar field. The *complete extremum graph* allows us to compare the proximity relationships, between all pairs of extrema.

4.1 Motivation

Consider the extremum graph in Figure 4.1. The maximum E can be *simplified* by merging it with an adjacent maximum A. In terms of the underlying function f, the maximum E and the saddle $\mathrm{S} = \mathcal{S}(\mathrm{E}, \mathrm{A})$ are *cancelled* by reversing the gradient flow direction along that path. The cancellation or simplification can be viewed as a local perturbation of the function by $f(\mathrm{C}) - f(\mathrm{S})$. This introduces adjacencies between the surviving maximum A and the other



(a) A scalar field with its extremum graph overlayed (b) Simplifying the extrema E by merging with the extrema A (c) Simplifying the extrema E by merging with the extrema B

Figure 4.1: Effect of simplification choices on proximity of surviving extrema

neighbours of E. The maximum C and its associated edges are removed.

Hierarchical structures have been introduced for Morse-Smale complexes [4], contour trees [30], and extremum graphs [15]. 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 the 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 makes 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 4.1, the maxima B and C can be made adjacent by merging the intermediate maximum E with B. However, persistence directed simplification will merge E with A. Following this simplification, B and C appear to be further away in the graph. Although persistence provides an intuitive measure of importance for vertices, the ensuing simplification can provide a non-intuitive 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. If there exists no path between a pair of extrema in the extremum graph, such as when the domain is disconnected, the algorithm continues to assign the persistence of its least persistent end-point as the cost of that edge. To better represent the domain, we can assign an *infinite* cost to such pairs.

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 or vertex attribute of the global maximum is set to 1. The cost of the edges present in the extremum graph is set to 0.

We derive a distance measure between extremum graphs based on these vertex and edge attributes.

4.2 Computation

We compute the extremum graph based on an approximate Morse decomposition [40]. 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. Any path between a pair of extrema can be viewed as a one dimensional function. The optimal cost to simplify all intermediate extrema of this path is equivalent to a persistence based simplification, restricted to this path. Further, we bound the cost by the persistence of the end points as mentioned above. We compute the minimum cost for pairs of extrema 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 maximum 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. If an edge appears between a pair of maxima with a cost c and after further simplification a different path is identified between the same pair at a cost $c'>c$ that introduces a higher saddle, only the saddle is updated and the cost of the edge is not changed. Updating the saddle does not affect the priority ordering of edges that are yet to be simplified.

Figure 4.2 shows a few steps in the construction of the complete extremum graph for the graph shown in Figure 2.4. 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

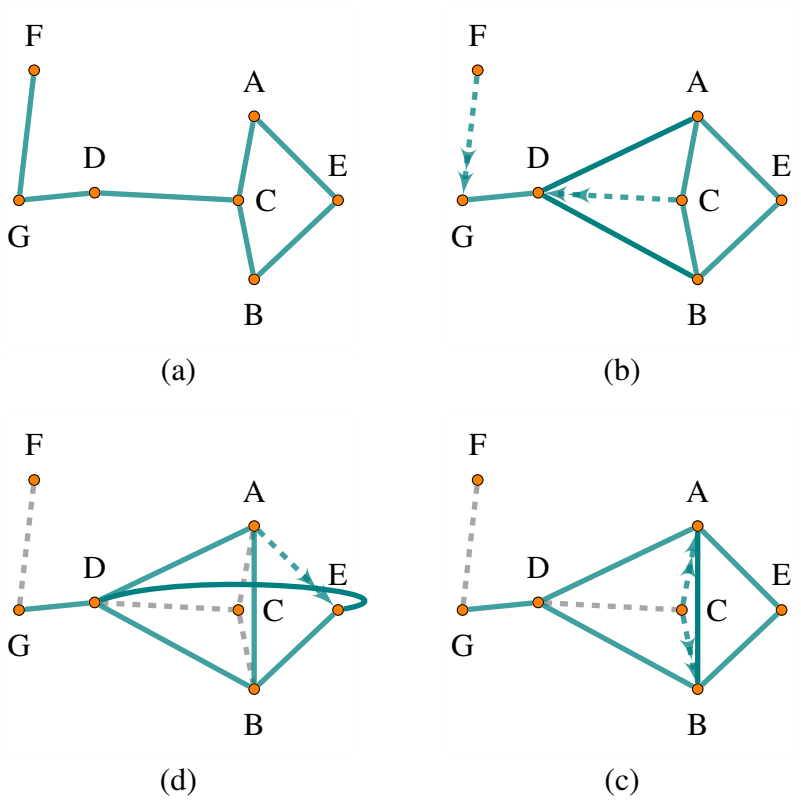


Figure 4.2: Construction of the complete extremum graph. (a) Input extremum graph EG. (b) (F, G) is processed. (C, D) is processed introducing (A, D) and (B, D). Other edges associated with C are retained. (c) (C, A) is processed and (A, B) is introduced. (d) (A, E) is processed introducing (E, D).

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 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 (A, B) and insert it. 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.

Algorithm 1: CEG - Constructs the 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**
    $\mathcal{C}((v_i,v_j)) = min(\mathcal{C}((v_i,v_j), min(\mathcal{P}(v_i), \mathcal{P}(v_j))$

4.2.1 Complexity

For an extremum graph with n maxima, the complete extremum graph consists of $O(n^2)$ edges. Each edge is processed only once and using a min heap, insertions and deletions take $O(\log n)$. Each iteration updates at most n edges, the time complexity of computing the complete extremum graph is $O(n^3 \log n)$.

4.2.2 Correctness

We now show that the algorithm 4.2 assigns the correct cost to all edges in the complete extremum graph. Consider a path $< v_1, p_1, p_2, \ldots, p_n, v_2 >$ between maxima v_1 and v_2 in the extremum graph. Let us assume that this path can be simplified into an edge (v_1, v_2) by perturbing the function by δ - the minimum function perturbation required for introducing (v_1, v_2).

First, we show that our algorithm assigns an edge cost $\mathcal{C}(v_1, v_2) \leq \delta$.
Case 1 : $\mathcal{P}(v_1) < \mathcal{P}(v_2)$ and $\mathcal{P}(v_1) \leq \delta$.
Given that the minimum cost of introducing the edge is δ and that we bound the edge costs by the persistence attributes of its end point vertices, in this case $(v_1, v_2) = \mathcal{P}(v_1) \leq \delta$ trivially.
Case 2 : $\mathcal{P}(v_1) < \mathcal{P}(v_2)$ and $\mathcal{P}(v_1) > \delta$.
As the path $< v_1, p_1, p_2, \ldots, p_n, v_2 >$ can be simplified within the perturbation threshold δ, the intermediate edges in this path can be processed with priority less than δ. As the persistence of v_1 and v_2 is higher, no edge involving the end point is processed before the intermediate edges are processed. Hence the edge (v_1, v_2) is constructed at a cost not greater than δ. The path realizing this edge need not be unique, however the minimum distance is uniquely computed.

Second, we show that if our algorithm assigns a cost $\mathcal{C}(v_1, v_2) = \delta$ and $\mathcal{P}(v1) > \mathcal{P}(v2) > \delta$, there exists no path in the extremum graph that can be simplified to introduce an edge (v_1, v_2) with cost less than δ.
If such a path exists, the edge contractions that realize the edge (v_1, v_2) through this path have

Figure 4.3: Updating the saddle of an existing edge does not violate the monotonicity introduced by the priority queue.

priority less than δ. By construction, these edges will be processed first and the cost assigned must be less than δ. Hence, we always assign the correct cost δ.

Finally, we clarify that the action of updating the saddle of an existing edge does not create inconsistencies in the priority queue ordering. As shown in Figure 4.3, suppose an edge (m_1, m_3) exists in the extremum graph. The edge (m_1, m_2) has minimum priority of p i.e., $f(m_2) - f(s_{12}) = p$ and can introduce a higher saddle s_{23} between (m_1, m_3) after it is processed. Then, the following holds: (1) $f(m_3) - f(s_{23}) > p$ and (2) $f(m_1) - f(s_{23}) > p$ i.e., although the priority value of the edge (m_1, m_2) decreases, it remains greater than p.

(1) trivially holds, since otherwise the edge (m_2, m_3) would be the selected with minimum priority.

(2) Similarly,

$$f(m_2) - f(s_{23}) > p \qquad \because \; p \text{ is minimum}$$

$$f(m_1) > f(m_2) \qquad \because \; p = f(m_2) - f(s_{12}) < f(m_1) - f(s_{12})$$

$$\therefore \quad f(m_1) - f(s_{23}) > p$$

4.2.3 Implementation Details

To compute the extremum graph, we consider as input a surface mesh, structured grid or unstructured grid. The scalar field is descritized and defined on the vertices of the mesh. We assume that neighbourhood information can be computed from the mesh representation. The

neighbourhood of a vertex is defined as the set of vertices with which it shares a common edge. Each vertex is considered to have a unique scalar value, breaking ties based on index ordering.

To identify maxima vertices, we consider the *upper link* of a vertex i.e the mesh induced by the set of neighbouring vertices with scalar value greater than the vertex. The upper link of maxima are empty.

To compute the maximum graph, we first compute an approximate decomposition of the vertices into descending manifolds [40]. We sort the vertices in decreasing order of scalar values. Maxima are labelled uniquely as their upper link is empty. Each vertex is visited in sorted order. The label of a vertex is set to the labels of the vertices in its upper link. Vertices that lie on the shared boundary of two descending manifolds inherit multiple labels. The vertex with the highest scalar value on a shared boundary is considered as the saddle between the two descending manifolds.

The vertices of the maximum graph are the identified maxima. An edge is introduced for every saddle, connecting the maxima on whose shared boundary the saddle lies. The edge structure maintains its *priority* value, defined as the difference between the scalar value of the lower maximum and the saddle. For every maximum, we also maintain a list of neighbouring maxima, defined by edge adjacency.

To compute the complete extremum graph, we insert the edges into a priority queue based on its priority value. Edges with lower value have higher priority. Edges are processed in order of their priority, as discussed in the algorithm 4.2. The processed edge is popped from the priority queue and new edges are pushed in based on the neighbourhood information of the maxima represented by the edge. The priority value of the popped edge is stored as the *cost* for the newly introduced edges. When the priority queue is empty, the complete extremum graph is constructed.

An implementation of the complete extremum graph algorithm is available at `https://bitbucket.org/vgl_iisc/compextgraph`

Chapter 5

Distance between Extremum Graphs

To compute the distance between a pair of extremum graphs, we adapt the maximum common subgraph based measure [9] 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 referred 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 edge distortion in the complete extremum graph 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.

5.1 Maps between extrema

Let $G_f(V,E_v)$ and $G_g(U,E_u)$ be the complete extremum graphs of $f : D \rightarrow [0,1]$ and $g : D \rightarrow [0,1]$ respectively. We extend the vertex set of G_f by a set of dummy vertices to obtain $V \cup \{\phi_{|V|+1}, \phi_{|V|+2}, \ldots, \phi_{|V|+|U|}\}$. 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, we denote edges that correspond as $(v_i, v_j) \mapsto (u_k, u_l)$.

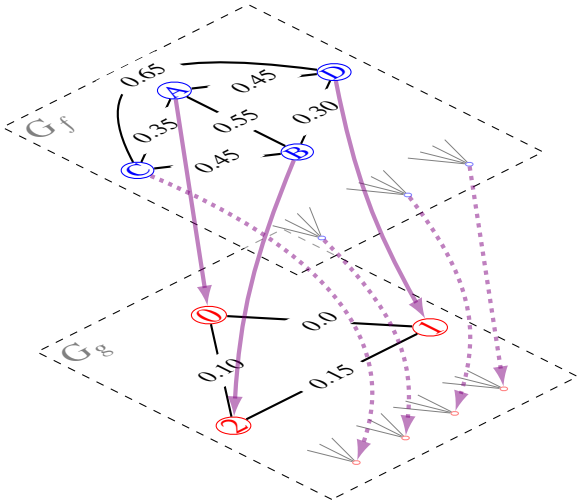
A map $F : V \to U$, is called ρ-valid for $\rho \in [0,1]$ and denoted by F_ρ if it is bijective and the edge distortion of corresponding edges is bounded by ρ. As we extend both vertex sets to contain $|V| + |U|$ vertices, bijective maps can be found for all values of ρ. Given a valid map F_ρ, We measure the vertex distortion between individual maximum that have been mapped. We denote the vertex distortion under a map F_ρ 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, $d^E_{F_\rho}((v_i, v_j) \mapsto (u_k, u_l)) = |\mathcal{C}((v_i, v_j)) - \mathcal{C}((u_k, u_l))| \leq \rho$. We assume the edge distortion involving a dummy edge is 0. Figure 5.1(a) and (b) describes valid maps for $\rho = 0.25$ and $\rho = 0.45$ respectively.

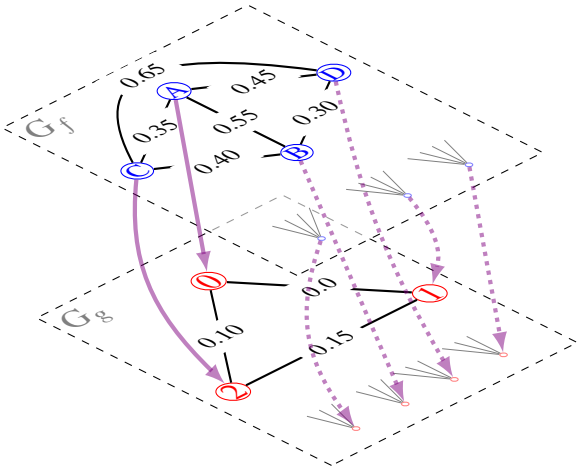
5.2 Distance between extremum graphs

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 $D^V_{F_\rho}(V,U)$ between the vertex sets and $D^E_{F_\rho}(E_u, E_v)$ 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 functions being compared in terms of their persistence. The distance between the edge sets indicates how well the proximity relationship between pairs of extrema are preserved.

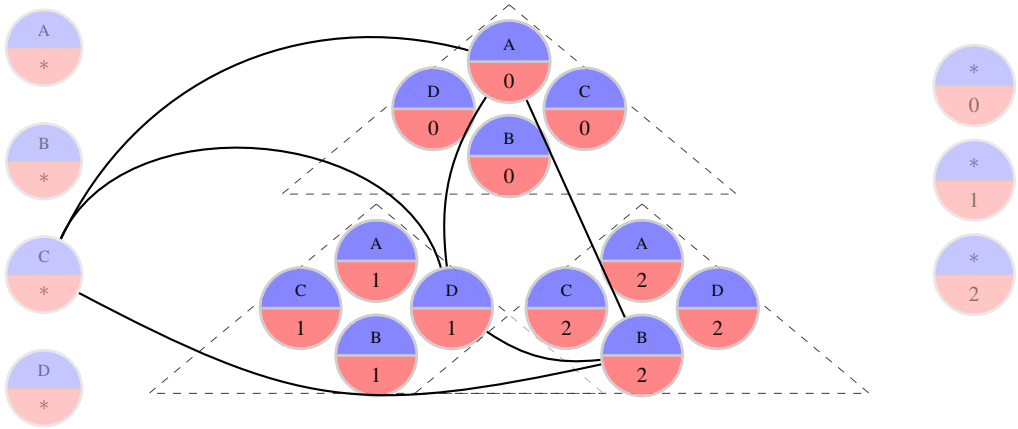
$$D^V_{F_\rho}(G_f, G_g) = max_{v \in V}\, d^V_{F_\rho}(v \mapsto F(v))$$
$$D^E_{F_\rho}(G_f, G_g) = max_{(v_i,v_j) \in E}\, d^E_{F_\rho}((v_i, v_j) \mapsto (F(v_i), F(v_j))$$



(a) A valid map for $\rho = 0.45$ is shown above. Dummy vertices are inserted into each graph to obtain a bijective map. Edges are labelled with their corresponding edge cost. 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.



(b) For $\rho = 0.25$, the figure shows a possible map between the two complete extremum graphs. The correspondence $\mathrm{A} \mapsto 0$ and $\mathrm{C} \mapsto 2$ satisfies the ρ criterion since $|\mathcal{C}((\mathrm{A},\mathrm{C})) - \mathcal{C}((0,2))| = |0.35 - 0.10| \leq 0.25$. The correspondence $\mathrm{B} \mapsto 1$ 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.



(c) Any valid map appears as a maximal clique in the product graph. Here, the map shown for $\rho = 0.45$ in figure 5.1b is shown in the (partial) product graph. It appears as maximal clique. An edge between vertex $A|0$ and $B|2$ is included in the product graph since the edge costs of edges (A,B) and $(0,2)$ have a difference $\leq \rho$. The vertices representing dummy correspondences(*) ensure that all extrema appear in the map and the valid map is always maximal. Note that introducing one dummy vertex for each extremum graph is sufficient to compute the appropriate product graph.

Figure 5.1: Examples of valid ρ-maps and product graphs

Each map gives an estimate on the distance between the two graphs G_f and G_g that measures the distortion in the graph induced by that map. The estimate is the sum of the vertex distance and the edge distance implied by the map. For a fixed ρ, the minimum over all possible maps gives an estimate distance between extremum graphs, D_ρ.

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

To compute the distance between the extremum graphs G_f and G_g, we consider distance estimates for all valid ρ values and assign the minimum value as the distance $D(G_f, G_g)$
$D(EG_f, EG_g) = D(G_f, G_g) = min\{D^G_{F_\rho}(G_f, G_g) | \forall \rho\}$

5.3 Composition of Maps

Let F_ρ be a ρ-valid map between complete extremum graph $G_f(V, E_v)$ 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 $(v_i, v_j) \mapsto (w_m, w_n) = |\mathcal{C}((v_i, v_j)) - \mathcal{C}((w_m, w_n))|$

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

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.

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

5.4 Metric Properties

We now verify that our distance measure $D(G_f, G_g)$ 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 a 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 a $\rho > 0$. 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.

5.5 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 set 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 $p{<}v_i, u_j{>}$. Vertex p indicates a possible correspondence between its *factor* vertices v_i and u_j. The weight of a vertex is defined as $w(p{<}v_i, u_j{>}) = 1 - d(v_i \mapsto u_j) = 1 - |\mathcal{P}(v_i) - \mathcal{P}(u_j)|$. In practice, we only require $|V'| + |U'|$ vertices in the product graph for identifying 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 $p_1{<}v_i, u_j{>}$ and $p_2{<}v_k, u_l{>}$ 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. We define the weight of a maximal clique to be 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 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 correspondence of the optimum map.

To compute the distance between two graphs with n and m vertices respectively, we need to consider $O(n^2m^2)$ ρ values. However, often applications may require that proximity be maintained strongly by the resulting maps. In such cases, we only compute distances for a few small values of ρ. We also minimize the number of ρ values tested by plotting all values and only considering values that significantly differ. These can be identified by peaks in the difference plot.

To compute the optimum correspondence map and thereby distance $D_\rho(G_f, G_g)$, we can

enumerate all maximal cliques in P and identify the maximum weight clique K_* [5]. 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 for only small graphs. We discuss some generic pruning strategies to sparsify product graphs and some approaches to partition extremum product graphs in order to reduce the search space.

5.5.1 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., $deg(p{<}v,u{>}) \cdot w(p{<}v,u{>})$ 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 [18], 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 $x{<}v,u{>}$ in the product graph and $W(S)$ be the weight of the maximum clique restricted to a set of vertices $S \in V \times U$. Let $W_{ub}(S)$ be the maximum bipartite matching cost between the factored vertices of S. Any maximal clique containing edge (x,y) is entirely contained in the intersection of their neighbourhoods by definition of a clique. Therefore, $W(N(x) \cap N(y)) \leq W_{ub}(N(x) \cap N(y))$.

To decide if the edge (x,y) can be a part of some maximum weight clique, we compute an upper bound on the weight of any maximal clique containing this edge and compare it against the weight of the greedily computed clique i.e. $W(x)+W(y)+W_{ub}(N(x) \cap N(y)) < W_{lb}$ implies that the weight of any maximal clique containing the edge (x,y) does not exceed the

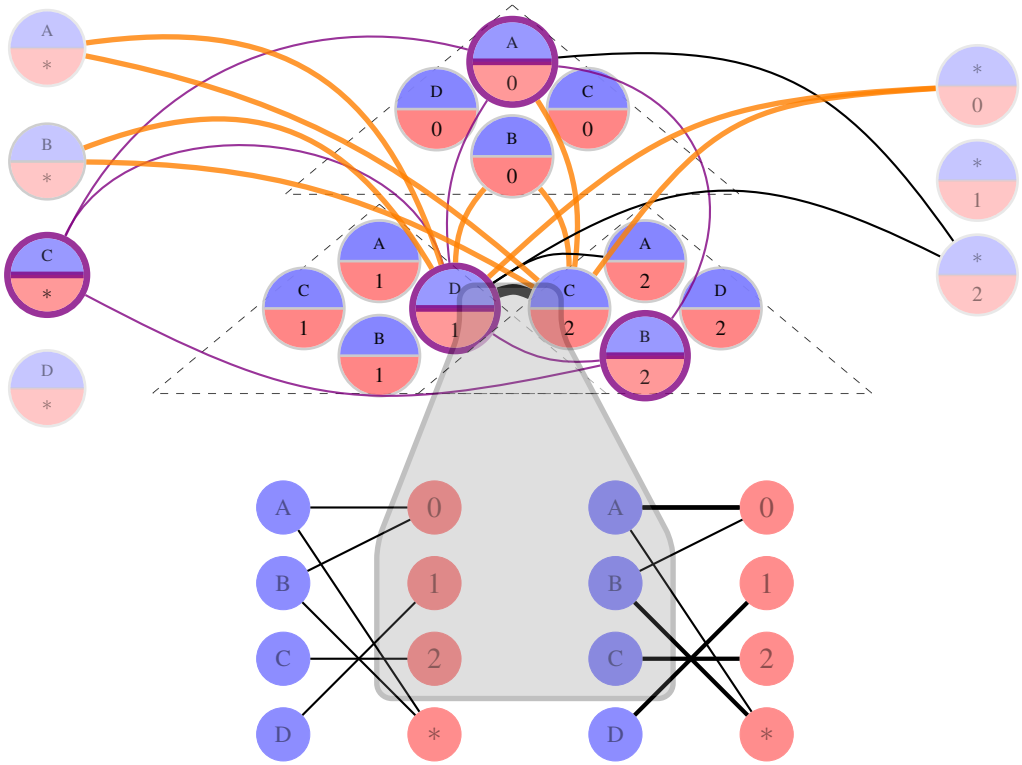


Figure 5.2: A partial product graph. Let the vertices highlighted in violet be the greedy maximal clique. We prune an edge if it cannot be a part of any maximal clique with a weight greater than the current maximum weight clique. Consider the edge $<\texttt{D1},\texttt{C2}>$. Any maximal clique that contains this edge lies in the intersection $N(\texttt{D1})$ and $N(\texttt{C2})$, indicated here by orange edges. By considering the factor vertices represented by these vertices and the possible matchings represented by these edges, we can compute a maximum bipartite matching. The cost derived from this matching gives an upper bound on any clique that contains $<\texttt{D1},\texttt{C2}>$. If this is lesser than the weight of the greedy clique, this edge can be safely pruned.

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 $x<v,u>$ is pruned if $W(x)+W_{ub}(N(x)) < W_{lb}$. Figure 5.2 illustrates edge pruning using bipartite matching in a product graph.

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.

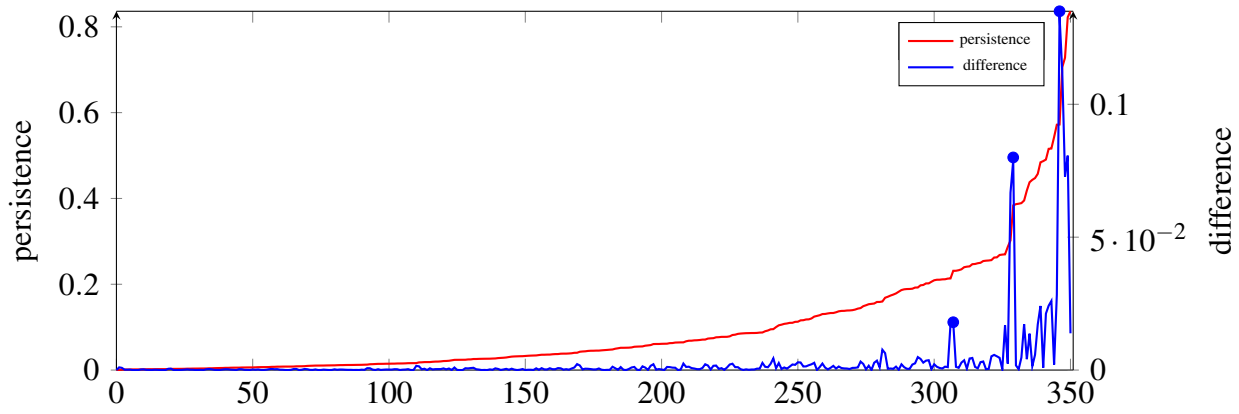


Figure 5.3: Partition thresholds for two time steps of the vortex flow data. Identified thresholds are marked on the difference plot. The thresholds partition vertices into sets of similar persistence values.

5.5.2 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. Figure 5.3 shows thresholds identified for two time steps in the vortex flow dataset.

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 K. At each iteration i, we consider only those vertices $p<v_i,u_j>$, 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 K. We now prune the product graph by eliminating vertices that are not adjacent to all the vertices in K. Vertices that indicate a dummy correspondence are included in the next iteration along with vertices that satisfy the next threshold. In the last iteration, we include all dummy correspondences also in K. Figure 6.5 plots the benefit of pruning and partitioning for the applications discussed in the next chapter.

Chapter 6

Applications

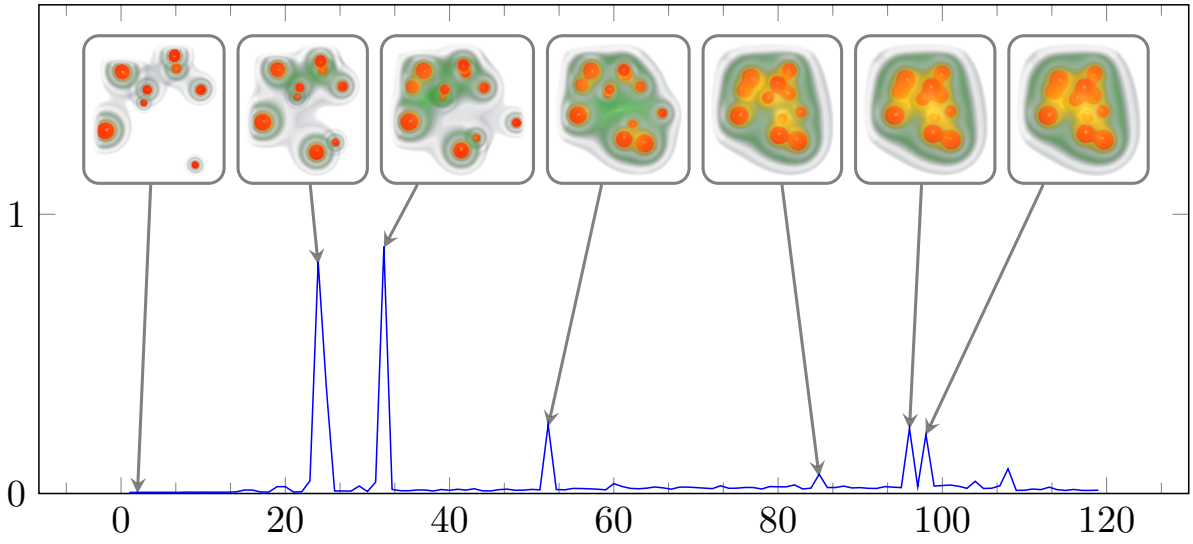


Figure 6.1: The line plot shows distance between the maximum graphs computed between consecutive time steps for a synthetic time-varying dataset. The distance plot helps in summarizing the dataset, peaks in the plot indicate frames that have a large distance with respect to the previous time step, indicating an event of importance. For example, the peak at time step 24 occurs due to the creation of a new feature visible in the bottom right corner.

Simulations that are time dependent or phenomena that are recorded over a large time frame are available as a series of scalar fields and are referred to as time varying data. For example, computational fluid dynamic simulation often observe the behaviour of fluids over time, generating time varying data. Climate experts may simulate or record data such as hourly wind speed during a hurricane period, generating a large series of scalar fields.

Time varying data are often available for a large time period due to the nature of the pysical phenomenon or a high resolution simulation. Visualizing the data frame by frame is tedious. Distance measures can be used to provide an overview or summary of the entire sequence of time varying 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 such as the proposed distance between extremum graphs, also facilitate tracking features across time steps.

Figure 6.1 shows a one dimensional *summary* plot for a synthetic time varying dataset. In this data, blobs or features are created at various time steps, these features merge by moving towards each other in subsequent time steps. The plot has been generated by computing the distance between maximum graphs for consecutive time steps of the data. 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. Such plots help in providing a quick overview of the data.

We now discuss some applications of our distance measure to a few time varying datasets that validate and illustrate its usefulness.

6.1 Periodicity in Time-Varying Data

Identification of periodicity in a time-dependent data is critical to understanding the underlying phenomenon and validating simulations. Figure 6.2 shows a few time steps of a synthetic fluid simulation of flow around a cylinder[1] that exhibits periodic vortex shedding [43]. The flow has been simulated for 1000 time steps and is available as 400×50 uniform two-dimensional grids. The magnitude of the two dimensional vector field is represented using a color map. The maximum graph is overlayed in yellow. The figure shows the Bénard–von Kárman vortex street formed by the flow.

[1]This data set has been simulated by Tino Weinkauf [42] using the Free Software *Gerris Flow Solver* [31] and is available from http://people.mpi-inf.mpg.de/ weinkauf/notes/cylinder2d.html.

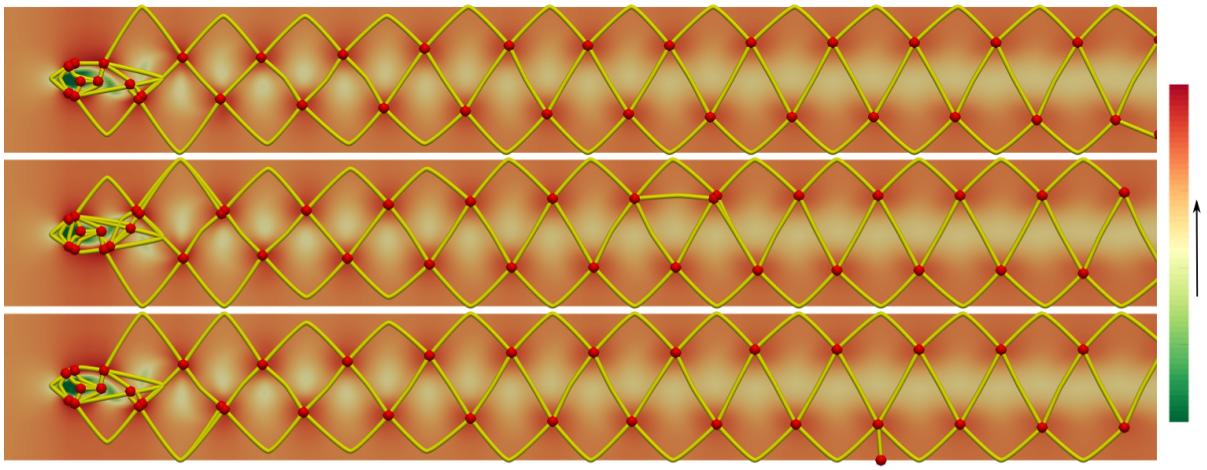


Figure 6.2: Time-step 0 (top), 38 (middle) and 75 (bottom) of the flow around a cylinder simulation. This simulation is time dependent with a time period of 75, these time steps appear similar. The vortex shedding alternates between the two sides of cylinder with a time period of 38. Time step 38 appears symmetric to time step 0 and 75. Maxima are shown as red spheres and the maximum graph is overlayed in yellow.

To verify that our distance function indeed identifies the known periodicity in this data [34], we compare each time step with all other time steps of the dataset. To ensure that edge constraints are strictly enforced, we use a low $\rho = 0.001$ across all comparisons. Figure 6.3 shows the vertex distortion plots for a few time steps. Each time step is compared against all time steps. It is easy to 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 6.2. Since the distance measure is over the extremum graph, the symmetric nature of these oscillations are identified.

Figure 6.5 shows that a speed up of $\sim 2.5\times$ is achieved by pruning edges and nodes in the product graph for this dataset. Graph details and running times are listed in Table 6.6.

6.2 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

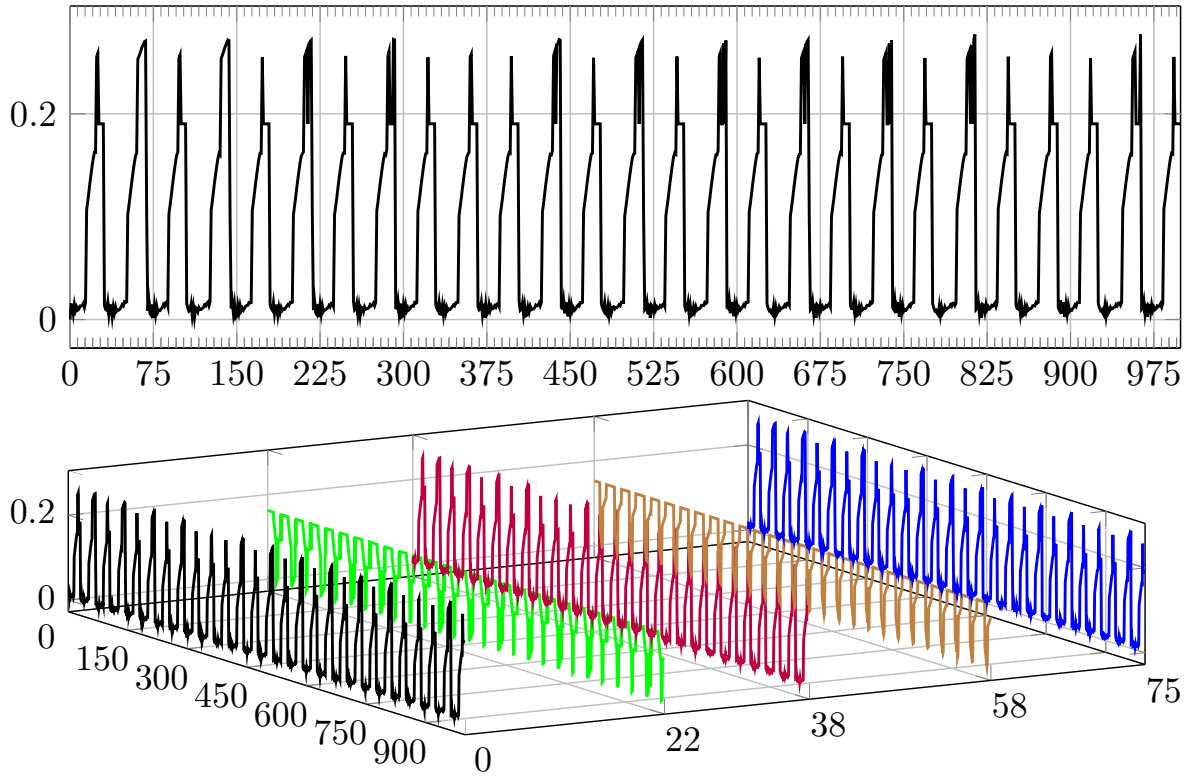


Figure 6.3: To identify periodicity, we compare the extremum graph of each time step with all 1000 time steps of the data. The line plot above shows distances computed between time step 0 and time steps 0-1000. The time steps are indicated on the x-axis and distance is indicated on the y-axis. The plot 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.

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 in a pseudospectral simulation of coherent turbulent vortex structures[2]. The volume data set has 128^3 resolution and has been simulated for 100 time steps.

The features in this data are the vortex cores that can be represented by isosurfaces with isovalues greater than $\sim 50\%$ of the maximum magnitude of the data, in each time step [37]. However, these features can merge or split in successive time steps. An important aspect of feature tracking is to identify these split and merge events across time steps. In order to track these features, we compute a correspondence between the maximum graphs of adjacent time steps. The correspondence is obtained by the map that generates the optimum distance between

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

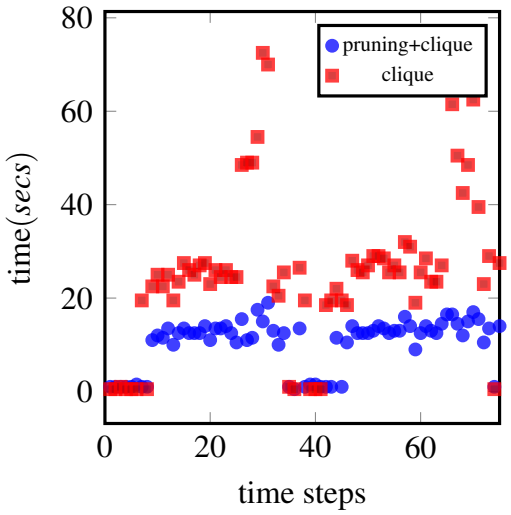
Figure 6.4: To track features in the turbulent vortex data, we compare the complete extremum graphs of consecutive time steps. Tracked features across time steps 2, 4 and 6 are shown on the left. Low opacity values indicate higher structural distortion in the complete extremum graph. Three features are shown in isolation on the right, the violet feature undergoes a split. The green and brown features merge, the purple feature grows.

the two graphs. We propogate the correspondence by transitivity across all time steps. To map correspondence established between the maxima to the isosurfaces representing the vortex cores, we label the isosurfaces based on the maxima whose descending manifolds they lie in.

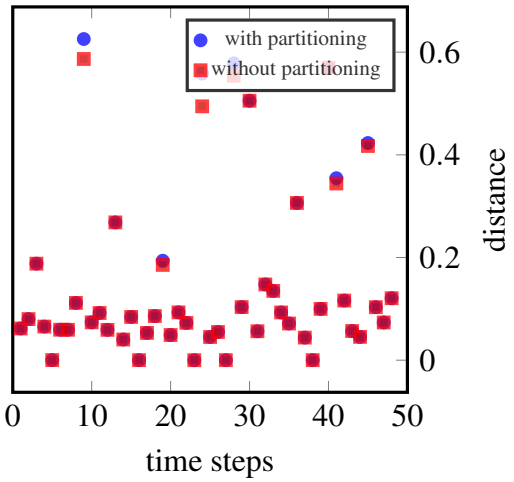
The complete graph structure allows us to compare features irrespective of instabilities in the merging order of the extrema or geometric overlap in the features. Figure 6.4 shows the isosurfaces for the first few time steps of the dataset. Isosurfaces are colored based on the correspondence obtained from the extremum graph. Surfaces that have the same color lie in descending manifolds of maxima that are in correspondence. The complete extremum graph implicitly handles merging and splitting of the extrema features. Figure 6.4 (right) shows three features in isolation from time steps 2 and 7. The violet feature splits into two. 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 in the persistence based simplification.

As the goal here is to compute correspondences and track as many features possible, we compute correspondences at various ρ levels. We identify these ρ values based on the peaks in the difference plot of the edge costs. The lowest ρ value at which a feature is tracked 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 6.4, 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 prune possible correspondences based on the moments of the descending manifold [22]. We compute the maximum clique hierarchically. Figure 6.5 shows the variation in the distance due to hierarchical clique computation. The maximum difference observed between the optimal clique and the hierarchically computed clique is less than 10%, typically the distance is much smaller. Graph details and running times are listed in Table 6.6.



(a) This plot shows the effectiveness of pruning in computing the maximum clique for the flow behind a cylinder dataset. We prune 1% of the edges in the product graph before computing the clique. The total time taken to prune the graph and compute the clique(blue) is less than the direct clique computation(red) by a factor of 2.5× for 75 time steps. The clique computation is done using the Bron-Kerbosch algorithm.



(b) This plot shows the variation in the distance computed for the vortex dataset with and without applying partitioning heuristics. For large datasets, where direct clique computation is computationally infeasible, partitioning allows us to compute the clique in a hierarchical fashion. The maximum difference observed between the optimal clique and the hierarchically computed clique for this dataset is $\leq$ 10% and typically much smaller.

Figure 6.5: Effect of pruning strategies on clique computation time and partitioning on distances computed.

Dataset	Size	# Ext	CEG Time	#Ext after simp	# vertices product graph	#edges product graph	Product Graph Time	Clique Time
Flow around cylinder	400×50	∼35	<1	∼35	∼900	∼300K	<5	∼10
Vortex	128^3	∼1000	∼70	∼60	∼2000	∼1500K	∼30	∼50

Figure 6.6: Running time(in seconds) and graph details for distance computations. Experiments were performed on a 2GHz Intel Xeon processor with 16GB RAM. 'Ext after simp' indicates the number of extrema considered for computing the distance. Note that the complete extremum graph (CEG) is computed using all unsimplified extrema, in order to compute correct edge costs. Pruning and partitioning techniques have been used to speed up the clique computations.

Chapter 7

Conclusions

In this thesis we presented a distance measure to compute the similarity between scalar fields using extremum graphs. We showcased the applicability of this distance in understanding time varying datasets. We introduced a complete extremum graph structure that captures distances 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. The pruning strategies are applicable to any product graph. Our partitioning strategy can be viewed as hierarchical matching, that performs matching at a coarser segmentation and then matches finer segments. Hence, if a large number of features of interest appear at the same scale, partitioning may fail to provide a good approximation of the distance. Further, 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.

The distance between the extremum graph is based on the distance between the edge sets and the vertex sets of the complete graph. We used application specific schemes to identify the edge distances or ρ values. As ρ values increase, the vertex distance decreases. These properties can be used to prune candidate ρ values. However, in general, one may have to compute vertex distance for all values to find the true distance. For extremum graphs with n vertices, there can be $O(n^4)$ candidates. A general approach to efficiently compute optimal

values needs further investigation.

Apart from the distance between extremum graphs, the complete extremum graph structure is an interesting abstraction on its own. It 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 [29], to understand scalar fields.

Finally, the complete extremum graph is designed to mitigate the effects of instabilities due to binary choices that are inherent in simplification based approaches. Also, the separation of vertex and edge distances allows the user to vary the influence of proximity on the correspondence. However, the stability and the discriminative aspects of this distance measure needs to be analyzed further.

References

[1] Chandrajit L Bajaj, Valerio Pascucci, and Daniel R Schikore. The contour spectrum. In *Proceedings of the 8th conference on Visualization'97*, pages 167–ff. IEEE Computer Society Press, 1997.

[2] Ulrich Bauer, Xiaoyin Ge, and Yusu Wang. Measuring distance between Reeb graphs. In *SOCG'14*, page 464, 2014.

[3] Kenes Beketayev, Damir Yeliussizov, Dmitriy Morozov, Gunther H Weber, and Bernd Hamann. Measuring the distance between merge trees. *Topological Methods in Data Analysis and Visualization III, Mathematics and Visualization. Springer-Verlag*, 2013.

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

[5] Coen Bron and Joep Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, September 1973.

[6] Stefan Bruckner and Torsten Möller. Isosurface similarity maps. *Comput. Graph. Forum*, 29(3):773–782, 2010.

[7] Roxana Bujack, Ingrid Hotz, Gerik Scheuermann, and Eckhard Hitzer. Moment invariants for 2d flow fields using normalization. In *Pacific Visualization Symposium (PacificVis), 2014 IEEE*, pages 41–48. IEEE, 2014.

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

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

[10] Gunnar Carlsson, Afra Zomorodian, Anne Collins, and Leonidas Guibas. Persistence barcodes for shapes. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 124–135. ACM, 2004.

[11] Hamish Carr, Jack Snoeyink, and Ulrike Axen. Computing contour trees in all dimensions. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 918–926. Society for Industrial and Applied Mathematics, 2000.

[12] Hamish Carr, Jack Snoeyink, and Michiel van de Panne. Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree. *Comput. Geom. Theory Appl.*, 43(1):42–58, January 2010.

[13] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1.

[14] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

[15] Carlos Correa, Peter 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.

[16] Michael Donoser and Horst Bischof. Efficient maximally stable extremal region (mser) tracking. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 553–560. IEEE, 2006.

[17] Harish Doraiswamy and Vijay Natarajan. Efficient algorithms for computing reeb graphs. *Computational Geometry*, 42(6):606–616, 2009.

[18] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

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

[20] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete and Computational Geometry*, 28(4):511–533, 2002.

[21] Herbert Edelsbrunner, Dmitriy Morozov, and Valerio 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.

[22] Jan Flusser, Barbara Zitova, and Tomas Suk. *Moments and moment invariants in pattern recognition*. John Wiley & Sons, 2009.

[23] Martin Haidacher, Stefan Bruckner, and Meister Eduard Gröller. Volume analysis using multimodal surface similarity. *IEEE Trans. Vis. Comput. Graph.*, 17(12):1969–1978, October 2011.

[24] Masaki Hilaga, Yoshihisa Shinagawa, Taku Komura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *SIGGRAPH*, pages 203–212, 2001.

[25] P Hut and J Makino. The art of computational science. *The Kali Code. Internet source*, 2003.

[26] J Matas, O Chum, M Urban, and T Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761 – 767, 2004. British Machine Vision Computing 2002.

[27] Dmitriy Morozov, Kenes Beketayev, and Gunther Weber. Interleaving distance between merge trees. *Discrete and Computational Geometry*, 49:22–45, 2013.

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

[29] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Transactions on Graphics (TOG)*, 21(4):807–832, 2002.

[30] Valerio Pascucci, Kree Cole-McLaughlin, and Giorgio 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.

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

[32] Georges Reeb. Sur les points singuliers dune forme de pfaff completement intégrable ou dune fonction numérique. *Comptes Rendus de LAcadémie ses Séances, Paris*, 222:847–849, 1946.

[33] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[34] 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.

[35] Dominic Schneider, Christian Heine, Hamish Carr, and Gerik Scheuermann. Interactive comparison of multifield scalar data based on largest contours. *Computer Aided Geometric Design*, 30(6):521–528, 2013.

[36] Dominic Schneider, Alexander Wiebel, Hamish Carr, Mario Hlawitschka, and Gerik 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.

[37] Deborah Silver and Xin Wang. Volume tracking. In *Visualization'96. Proceedings.*, pages 157–164. IEEE, 1996.

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

[39] Dilip Mathew Thomas and Vijay Natarajan. Symmetry in scalar field topology. *IEEE Trans. Vis. Comput. Graph.*, 17(12):2035–2044, 2011.

[40] Dilip Mathew Thomas and Vijay Natarajan. Detecting symmetry in scalar fields using augmented extremum graphs. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2663–2672, 2013.

[41] Bruno Vallet and Bruno Lévy. Spectral geometry processing with manifold harmonics. In *Computer Graphics Forum*, volume 27, pages 251–260. Wiley Online Library, 2008.

[42] 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.

[43] Hong-Quan Zhang, Uwe Fey, Bernd R Noack, Michael König, and Helmut Eckelmann. On the transition of the cylinder wake. *Physics of Fluids (1994-present)*, 7(4):779–794, 1995.

[44] 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.

[45] Jianlong Zhou and Masahiro Takatsuka. Automatic transfer function generation using contour tree controlled residue flow model and color harmonics. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1481–1488, 2009.