Mathematical Statistician and Engineering Applications
ISSN: 2094-0343
DOI: https://doi.org/10.17762/msea.v70i1.2517

Investigation of Computational Topology for Data Analysis and Visualization Applications

Robin
Associate Professor, Department of Mathematics, Graphic Era Hill University, Dehradun Uttarakhand India

Article Info
Page Number: 628-636
Publication Issue:
Vol. 70 No. 1 (2021)

Abstract
In a variety of disciplines, from biology and physics to computer science and the social sciences, computational topology has become a potent tool for data analysis and visualisation. This study explores computational topology's uses in data analysis and visualisation, showing its potential for revealing hidden structures and patterns in large, complicated datasets.The paper starts out by giving a general introduction of computational topology and outlining its core ideas and methods. The use of it for extracting important information from big, high-dimensional datasets is then explored in relation to data analysis. Computational topology allows for the discovery of topological properties such as holes, voids, and connection patterns by modelling data as topological structures, such as simplicial complexes or persistent homology diagrams.The study also explores the function of computational topology in data visualisation, highlighting its capacity to offer clear and insightful visual representations of challenging datasets. Computational topology enables the construction of simpler and aesthetically pleasing representations while preserving the fundamental topological properties of the data through methods like topological simplification and dimensionality reduction.The usefulness of computational topology in several application domains, such as genomics, image analysis, and network analysis, is demonstrated through a number of case studies. These instances show how computational topology can improve activities like data exploration, clustering, classification, and anomaly detection, resulting in fresh perceptions and learnings.This study emphasises the enormous potential of computational topology for applications in data processing and visualisation. Computational topology provides a new perspective that complements current approaches by utilising the inherent geometric and topological properties of data. This allows researchers and practitioners to better understand complex datasets and make decisions based on the knowledge gleaned from them.

Article History
Article Received: *25 January 2021*
Revised: *24 February 2021*
Accepted: *15 March 2021*

Keywords: Topology in computation, data evaluation, Visualization, structural topologies, voluminous datasets

Introduction

Computational topology techniques have attracted increasing attention in the field of data analysis and visualisation in recent years as a means of overcoming the difficulties presented by complicated and high-dimensional datasets. Researchers and practitioners can find hidden patterns, extract valuable information, and gain deeper insights into the underlying data by using the mathematical framework that computational topology provides to explore the topological features and structures

of data.In order to comprehend and analyse data, traditional data analysis methods frequently rely on statistical methodologies and geometrical approaches. When dealing with complicated and noisy datasets that have complex structures and non-linear correlations, these methods could, nevertheless, fall short. Beyond statistical summaries and geometric representations, computational topology offers a novel viewpoint by exploring the inherent topological and geometric properties of data.The core principle of computational topology is to describe data as topological structures, such as persistent homology diagrams or simplicial complexes. A potent [1]vlens for analysing and visualising complicated datasets, these representations highlight the connectedness, shape, and higher-dimensional features present in the data. Computational topology makes use of ideas like homology, persistent homology, and topological invariants to detect topological phenomena like holes, voids, and connection patterns that might not be immediately noticeable when using conventional methods [2][3].

Computational topology also provides intriguing directions for data visualisation. It is now possible to simplify and [4]visually intuitively portray complicated information while maintaining crucial topological qualities by using techniques like dimensionality reduction and topological simplification. This makes it easier to explore, interpret, andcommunicate complicated. In-depth research into computational topology's uses in data processing and visualisation is the main goal of this article. We will examine the core ideas, methods, and algorithms of computational topology, emphasising how they can be used to overcome the difficulties associated with analysing and [5]visualising complicated datasets. In order to demonstrate the efficiency of computational topology in revealing hidden structures, clustering, classification, and anomaly detection tasks, case studies from a variety of application fields, including genomics, image analysis, and network analysis, will be examined[7][8].

Computational topology [10] offers a complementary approach to conventional data analysis and visualisation methods by utilising the geometric and topological features of data. It gives us a way to draw important conclusions from large datasets and improves our comprehension of the underlying patterns and relationships.

I. Related Work

Related work by Doraiswamy et al. [16] established the idea of topological saliency, which quantifies the importance of key points in a scalar field by calculating how they are distributed spatially. While this method ranks persistence pairs and produces saliency curves for various smoothing radii, it does not reveal anything about the nesting habits of these pairs. As a result, it is unable to discern between spatial rearrangements, such as bringing peaks closer together, that retain persistence values and relative distances.

The regular persistence hierarchy, on the other hand, was created by Bauer [1] and is a combinatorial method that focuses on identifying cancellation sequences of important points on surfaces. However, as shown in this study, this hierarchy is unable to distinguish between certain nesting relations among persistence pairs.Therefore, by creating a hierarchy based on the topological characteristics of the data, our suggested approach seeks to address these restrictions. In addition to capturing the nesting behaviour of persistence pairs, this hierarchy offers a thorough and insightful description of their spatial configurations. Our method improves the persistence hierarchy's expressive capacity and discriminative skills by combining both topological and

geometric information.Traditional graph architectures like the Reeb graph [15], contour tree [11], merge tree, and split tree have received a lot of attention in scalar field research. These graphs establish connections between pivotal locations but do not help with persistence pair hierarchy calculation. This constraint will be demonstrated in this study using a straightforward one-dimensional example.

The goal of our work is to capture geographical distinctions within the domain, which is a motive shared by recent research in this field. For instance, even when two scalar fields have identical critical pairings, the merge tree, in contrast to conventional graph architectures, can preserve differences in sublevel set merging behaviour. In order to measure these differences, distance measures have been created for merge trees [3], Reeb graphs [2], and extremum graphs [22]. A hierarchical decomposition known as the branch decomposition, which connects the branches of a contour tree, was introduced by Pascucci et al. [23] to reduce the complexity of these tree topologies. Saikia et al. [25] used these graphs more recently as similarity metrics for the structural comparison of scalar data.

II. Structural Topology in Continuity Hierarchies

The determination of pairings between local minima and local maxima (or saddles in higher dimensions) based on the elder rule is necessary to calculate persistence in the setting of zero-dimensional persistent homology, which focuses on related components in the sublevel sets of a function. Other topological properties and calculations[12] for superlevel set are left for further investigation, but zero-dimensional persistent homology and sublevel sets are specifically covered in this study.

1. Regular Persistence Hierarchy

In his work, Bauer [1] emphasises how a hierarchy of persistence pairs might naturally emerge as a result of the merger of two connected components. We specifically consider'to be the parent of'when given two connected components, and ', produced at local maxima and merging into'at a local maximum. Which pairs of crucial points in a Morse function cannot be cancelled without impacting other points is determined by this relationship, but it is not sufficient. The regular persistence hierarchy is another name for this structure, which is also referred to as a merge tree [11].

Sample and Restrictions:

Unfortunately, there are restrictions on how specific connection interactions can be distinguished by the standard persistent hierarchy. This is seen in Figure 1 where we show the typical persistence hierarchy for two straightforward functions. We can observe that the hierarchy for both functions are the same despite their distinct connection behaviours. The two persistence pairs in the case of the red function are specifically coupled through two different branches of the function. In other words, as the threshold of the sublevel sets increases, it becomes impossible to attain both minima without first passing through a third minimum, which is the global minimum.

Mathematical Statistician and Engineering Applications
ISSN: 2094-0343
DOI: https://doi.org/10.17762/msea.v70i1.2517

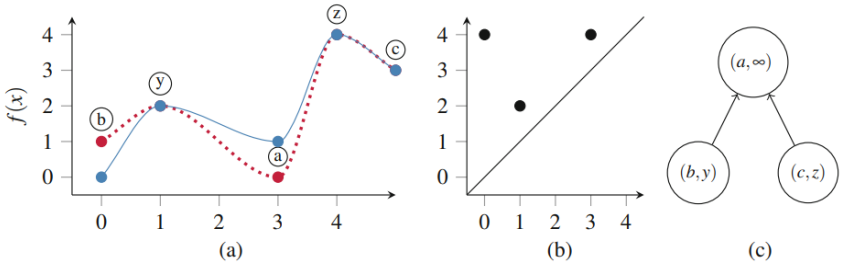

Figure 1: Diagrams for two functions that have different connection but the same persistence. Additionally, the standard persistence structure is shared by both functions. Functions (a). (a) A diagram of persistence. (c) Hierarchy

Figure 1's example serves as a display of how little discriminative information the standard persistence hierarchy offers. The running example illustrates a fundamental finding that not all merges of connected components are topologically similar. A merging can either maintain the existing branches or result in a clear branching structure within the hierarchy. We suggest the interlevel set persistence hierarchy (ISPH), shown in Figure 2, to overcome this restriction.

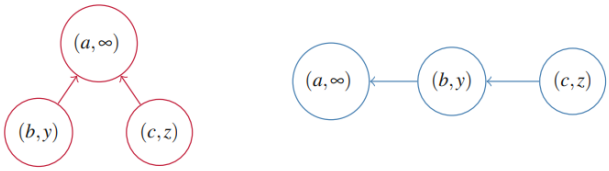

Figure 2: Example of TSPH

Require: A domain $\mathbb{D}$
Require: A function $f : \mathbb{D} \to \mathbb{R}$
U $\leftarrow \emptyset$
Sort the function values of f in ascending order
for function value y of f **do**
  if y is a local minimum **then**
    Create a new connected component in U
    U.critical $\leftarrow y$
  else if y is a local maximum or saddle **then**
    Use U to merge the two connected components meeting at y
    Let C' and C be the two components meeting at y
    if both components have a trivial critical value **then**
      Create the edge (C', C) in the hierarchy
    else
      Let c' be the critical value of the older connected component
      Let c be the critical value of the younger connected component
      $y_l \leftarrow \min(c, c')$
      Create the interlevel set $L := \mathscr{L}_{y_l, y}(f)$
      if the shortest path connecting c, c' in L contains no other critical points **then**
        Create the edge (c', y) in the hierarchy
      else
        Create the edge (C', C) in the hierarchy (as above)
      end if
    end if
  else
    Use U to add y to the current connected component
  end if
end for

Figure 3: Snapshot of calculation of TPSH algorithm [7]

For examining and contrasting time-varying scalar fields, one computational topology-based method is the Time-Parameterized Surface Hierarchy (TPSH) algorithm. It attempts at capturing the evolution and changes in scalar fields' topology across time. To represent the time evolution of the scalar field, the TPSH method builds a hierarchy of topological features, such as critical points, persistence pairs, and interlevel sets.

The TPSH algorithm runs through the following steps:

- Regular Persistence Hierarchy Construction: The algorithm initially determines the regular persistence hierarchy for each timestep separately. This hierarchy captures the behaviour of related components in the scalar field's sublevel sets as they merge and split.
- After that, the TPSH algorithm creates temporal pairings between related topological properties over various timesteps. These pairs are established using the scalar field's evolution, which shows how features change over time and can merge, split, or endure.
- Interlevel Set Persistence Hierarchy: The TPSH algorithm creates the Interlevel Set Persistence Hierarchy (ISPH) using the temporal pairings. This hierarchy depicts the topological feature changes and temporal relationships over several timesteps. The topology of the scalar field's temporal evolution is compactly represented, and it captures the joining and separating of sublevel sets.
- Dissimilarity Measure: Based on the ISPH, the TPSH algorithm proposes a dissimilarity measure. In order to compare and analyse the temporal variations in the scalar field, this measure measures the differences in the topological structures between pairs of timesteps.

Researchers may learn more about the temporal behaviour of scalar fields, spot noteworthy topological changes, and compare the evolution of various datasets by utilising the TPSH algorithm. The TPSH algorithm has applications in many areas, including fluid dynamics, medical imaging, and climate analysis. It provides a computationally effective and expressive framework for analysing time-varying scalar fields.

2. Calculating Ranks

We can rank the topological characteristics of the directed acyclic graph (DAG) that represents the interlevel set persistence hierarchy (ISPH). If there is a directed path that connects two vertices u and v in the hierarchy H, we write u v. The number of vertices that may be reached from a vertex, such as u, is used to determine its rank in the hierarchy.

$$\text{rank}(u) := \text{card}\{v \in H \mid u \sim v\} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

3. Measure of Dissimilarity

Tree edit distance techniques can be used to determine an appropriate dissimilarity measure because the interlevel set persistence hierarchy (ISPH) is a directed tree [7]. Through three simple operations—relabeling a node, removing an existing node, and inserting a new node these algorithms seek to change one tree into another.

- Given two nodes that correspond to minima-maxima pairs, (c1, d1) and (c2, d2), respectively, we define the cost of relabeling in the context of ISPH as follows:
- When c1 and c2 are equal, there is no relabeling expense.
- The relabeling cost is one if c1 and c2 are different but d1 and d2 are equal.
- The relabeling cost is two if c1 and c2 differ and d1 and d2 both differ.

The dissimilarity between ISPH structures is measured using tree edit distance methods using these costs, which indicate the difference between the minima of the two nodes.

$$\text{cost1} = \text{Max}(|c1 - c2|, |d1 - d2|) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

$$\text{cost2} = \text{pers}(c, d) = |d - c| \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

4. Results and Discussion

On a grid with 5000 cells, we created two synthetic datasets, and regardless of whether we computed the standard persistence hierarchy or the interlevel set persistence hierarchy (ISPH), each dataset took about 4.5 seconds to process. Our present implementation may still be performing better, though. The data and the resulting hierarchies for the two-dimensional equivalent of the data are shown in Figure 4. Red stands for high values and white for low values on a conventional colour map. It is remarkable that both datasets' normal persistence hierarchies and persistence diagrams are identical.

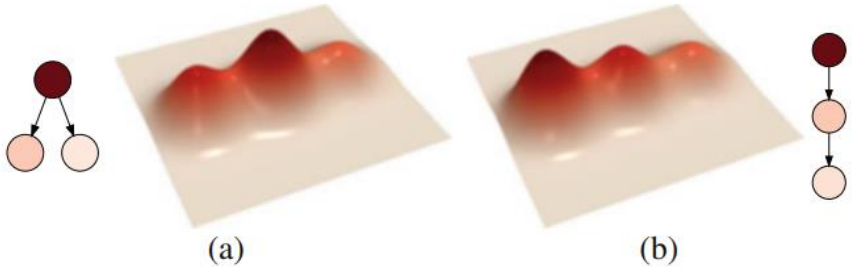

Figure 4: Our hierarchy, in contrast to earlier methods, can tell the difference between two peaks that are connected by a higher third peak (a) and a "ridge" of peaks (b).

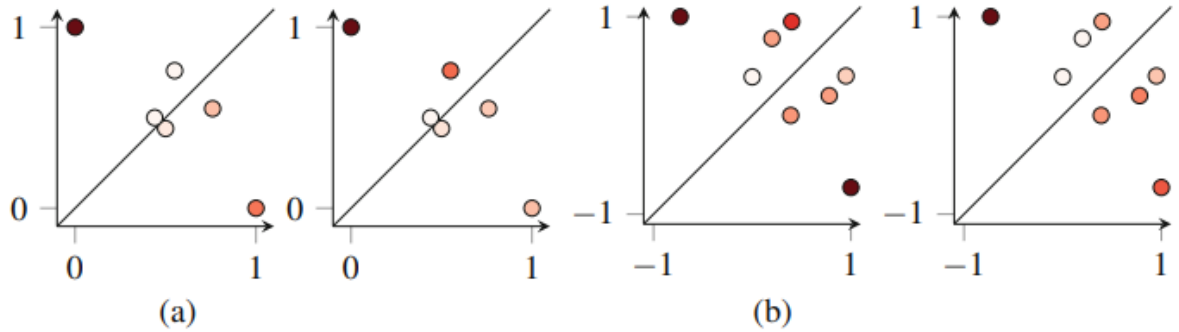

Figure 5: Persistence diagrams that combine the colours for the ranks (upper part) and stability values (bottom portion).

We used time-varying scalar field data from the German Climate Computing Centre (DKRZ) for our investigation. Due to the dataset's vastness, which included 18,422 places and 1462 timesteps, it presented difficulties for comparison. The dataset comprised of surface temperature observations. Figure 6 shows an example from the dataset and oscillatory behaviour associated with the day-night pattern seen in global temperature changes. The resolution of the data indicates that a whole day-night cycle takes place over four timesteps. As a result, we predict that the topological dissimilarity between corresponding timesteps will be largely consistent, or more specifically.

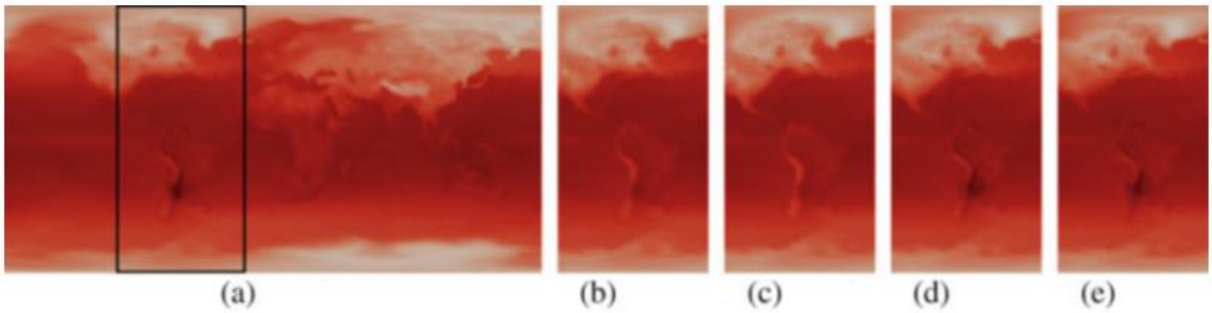

Figure 6: Climate related dataset. (a), along with a few snippets (box) for later timesteps. We can observe that at t = 3 and t = 4, the temperature on the continent of Africa rises.

(a) t = 0.0 (b) t = 1.0 (c) t = 2.0 (d) t = 3.0 (e) t = 4.0

We used pairwise comparisons between the first 36 timesteps of the dataset to compare the Wasserstein distance with the dissimilarity metric defined in to assess its efficacy. Each pair's Wasserstein distance calculation took about 2.1 minutes, making the total computation time for all 38 pairings about 24 hours. In comparison, our hierarchy-based dissimilarity measure computed the dissimilarity for each timestep in just 2.33 seconds and for each pair in roughly 6.7 seconds. Consequently, it took about 1.6 hours to collect the entire distance matrix.

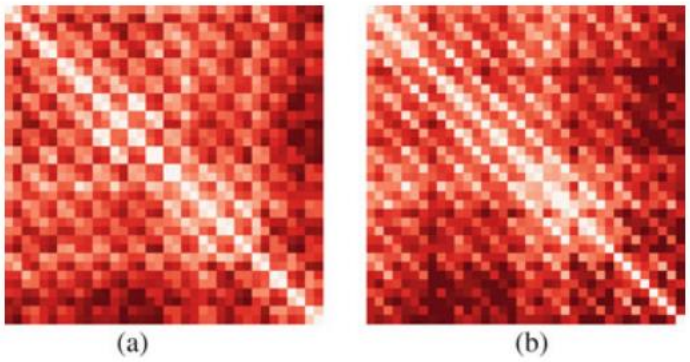

Figure 7. Comparisons between the Wasserstein distance and our suggested ISPH distance, which can identify the oscillatory behaviour (minor diagonals) inherent in the data. Wasserstein distance, to start. (a) ISPH separation

References:

[1] Bauer, U.: Persistence in discrete Morse theory. Ph.D. Thesis, University of Göttingen (2011)

[2] Bauer, U., Ge, X., Wang, Y.: Measuring distance between Reeb graphs. In: Proceedings of the Annual Symposium on Computational Geometry, pp. 464:464–464:473 (2014)

[3] Beketayev, K., Yeliussizov, D., Morozov, D., Weber, G.H., Hamann, B.: Measuring the distance between merge trees. In: Topological Methods in Data Analysis and Visualization III: Theory, Algorithms, and Applications, pp. 151–165. Springer, Berlin (2014)

[4] Bendich, P., Bubenik, P.: Stabilizing the output of persistent homology computations (2015). arXiv:1512.01700

[5] Bendich, P., Edelsbrunner, H., Kerber, M.: Computing robustness and persistence for images. IEEE Trans. Vis. Comput. Graph. 16(6), 1251–1260 (2010)
[6] Bendich, P., Edelsbrunner, H., Morozov, D., Patel, A.: Homology and robustness of level and interlevel sets. Homology Homotopy Appl. 15, 51–72 (2013)
[7] Bille, P.: A survey on tree edit distance and related problems. Theor. Comput. Sci. 337(1), 217–239 (2005)
[8] Carlsson, G., Zomorodian, A.J.: The theory of multidimensional persistence. Discret. Comput. Geom. 42(1), 71–93 (2009)
[9] Carlsson, G., de Silva, V., Morozov, D.: Zigzag persistent homology and real-valued functions. In: Proceedings of the Annual Symposium on Computational Geometry, pp. 247–256 (2009)
[10] Carr, H., Snoeyink, J., van de Panne, M.: Simplifying flexible isosurfaces using local geometric measures. In: IEEE Conference on Visualization, pp. 497–504 (2004)
[11] Carr, H., Snoeyink, J., Axen, U.: Computing contour trees in all dimensions. Comput. Geom. 24(2), 75–94 (2003)
[12] Chazal, F., Guibas, L.J., Oudot, S.Y., Skraba, P.: Persistence-based clustering in Riemannian manifolds. J. ACM 60(6), 41:1–41:38 (2013)
[13] Cohen-Steiner, D., Edelsbrunner, H., Harer, J.: Stability of persistence diagrams. Discret. Comput. Geom. 37(1), 103–120 (2007)
[14] Cohen-Steiner, D., Edelsbrunner, H., Harer, J., Mileyko, Y.: Lipschitz functions have Lp-stable persistence. Found. Comput. Math. 10(2), 127–139 (2010)
[15] Doraiswamy, H., Natarajan, V.: Efficient algorithms for computing Reeb graphs. Comput. Geom. 42(6–7), 606–616 (2009)
[16] Doraiswamy, H., Shivashankar, N., Natarajan, V., Wang, Y.: Topological saliency. Comput. Graph. 37(7), 787–799 (2013)
[17] Edelsbrunner, H., Harer, J.: Computational Topology: An Introduction. AMS, Providence (2010)
[18] Edelsbrunner, H., Mücke, E.P.: Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. ACM Trans. Graph. 9(1), 66–104 (1990)
[19] Gerber, S., Bremer, P.T., Pascucci, V., Whitaker, R.: Visual exploration of high dimensional scalar functions. IEEE Trans. Vis. Comput. Graph. 16(6), 1271–1280 (2010)
[20] Maria, C., Boissonnat, J.D., Glisse, M., Yvinec, M.: The Gudhi library: simplicial complexes and persistent homology. In: Hong, H., Yap, C. (eds.) Mathematical Software – ICMS 2014, pp. 167–174. Springer, Heidelberg (2014)
[21] Milnor, J.: Morse Theory. Princeton University Press, Princeton (1963) Hierarchies and Ranks for Persistence Pairs 17
[22] Narayanan, V., Thomas, D.M., Natarajan, V.: Distance between extremum graphs. pp. 263–270 (2015)
[23] Pascucci, V., Cole-McLaughlin, K., Scorzelli, G.: The Toporrery: computation and presentation of multi-resolution topology. In: Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration, pp. 19–40. Springer, Berlin (2009)

[24] Rieck, B., Leitte, H.: Structural analysis of multivariate point clouds using simplicial chains. Comput. Graph. Forum 33(8), 28–37 (2014)
[25] Saikia, H., Seidel, H.P., Weinkauf, T.: Extended branch decomposition graphs: structural comparison of scalar data. Comput. Graph. Forum 33(3), 41–50 (2014)