PersistenceBundles: Visual Comparison of Topological Features

Adhitya Kamakshidasan, Vijay Natarajan

▶ **To cite this version:**

Adhitya Kamakshidasan, Vijay Natarajan. PersistenceBundles: Visual Comparison of Topological Features. 2020. hal-02569089v2

HAL Id: hal-02569089

https://inria.hal.science/hal-02569089v2

Preprint submitted on 11 Aug 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

PersistenceBundles: Visual Comparison of Topological Features

Adhitya Kamakshidasan*
University of Glasgow

Vijay Natarajan†
Indian Institute of Science, Bangalore

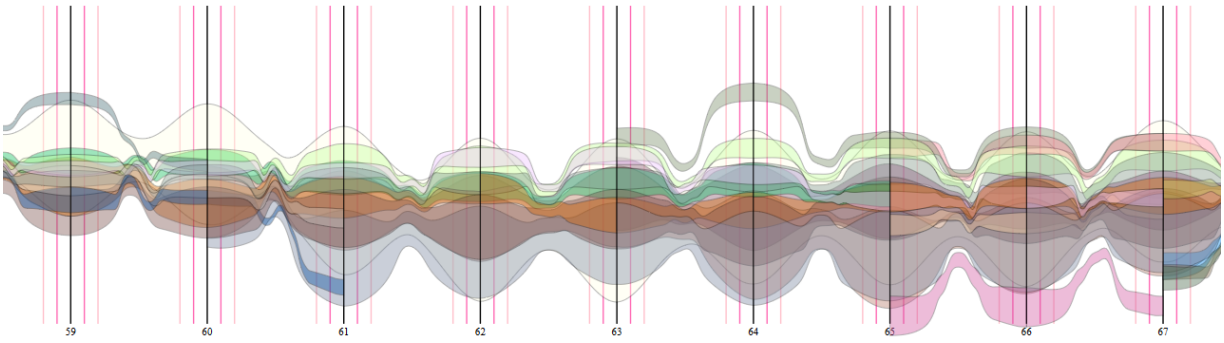


Figure 1: *persistenceBundles* for one ensemble member of the viscous finger dataset [2] between timesteps 59 - 67. The representation immediately helps identify time steps where new features originate as in timestep 61, or where there is a change in the hierarchy as in timestep 65.

Abstract

In time-varying scientific datasets, the temporal evolution of interesting topological features is commonly displayed and explored using isosurfaces and tracking graphs. However, the visual representation of such tracking graphs supports only few interactive capabilities. Further, they capture information at a high level that requires specification of carefully chosen parameter values. To bridge this gap, we propose *persistenceBundles*, a flexible visualization metaphor that utilizes a hierarchical edge-bundling approach for visualizing tracked features using persistence hierarchies, and implicitly allows for intuitive interaction schemes. We demonstrate the effectiveness of our approach using the viscous finger dataset.

Keywords: time-varying data, visual representations, feature tracking, persistence hierarchy

1 Introduction

The rapid growth of computational advancements has transformed the ability to acquire and analyse scientific datasets. Such scientific datasets are usually scalar fields defined over a 2D/3D domain and obtained either using software simulations or advanced hardware systems like cryo-electron microscopy and satellite imagery. The application domain experts are often interested in identifying prominent features present in their data, and to understand how they change over time. Topological Data Analysis (TDA) performs formidably in this regard, as it provides the mathematical tools to derive temporal insights by capturing the underlying structural properties of the data.

Usually, while employing TDA in time-varying contexts, changes are detected by deriving topological features from one temporal snapshot and comparing them to another. Though sophisticated methods exist for such comparisons, an intuitive visual representation is often challenging, owing to a large number of features. For overcoming this, existing visualization methods render static representations that compare a few user-specified thresholds on a global scale with little interaction. However, it is necessary for the visualization method to additionally support the comparison of the entire hierarchy of features along with local comparisons and interactive queries. We present a design called persistenceBundles, that focuses on visual scalability and supports both coarse-grained events of interest, such as aggregated similarities and differences, as well as capabilities to filter fine-grained events based on persistence or other geometric features. Our design is intuitive and builds upon a topological abstraction called the persistence hierarchy.

*e-mail: adhitya.kamakshidasan@glasgow.ac.uk
†e-mail: vijayn@iisc.ac.in

2 Related Work

Identifying and tracking topological feature is a well-studied area within the scientific visualization community. In this section, we highlight some of the relevant past work particularly with respect to the exploratory capabilities supported by the various techniques. We refer the reader to [4] for a more detailed survey on the broader topic of feature tracking.

Topological features in the context of scalar field visualization are defined based on either the evolution of isosurface connectivity or analysis of the gradient field. Topological structures such as the contour tree, Reeb graph, and merge tree provide succinct representations of the evolution of isosurface connectivity and have been used for various tracking applications in weather and climate science [8], combustion studies [35], cricket analysis [13], and high dimensional point clouds [24].

Similarly, multiple developments have been made towards providing visual interfaces for facilitating easier interpretation of tracking in time-varying datasets. Previous work often utilize a *tracking graph* scheme, to capture the evolution of all features across time as a collection of feature tracks that may merge or split. [6, 7] use higher-dimensional space-time Reeb graphs for constructing tracking features in turbulent combustion. Widanagamaachchi et al., [36] pioneered meta-graphs, an abstraction for storing overlap information from consecutive merge trees, and provided node-link interfaces for viewing tracked data. The authors also extended their work to understand atmospheric phenomena by looking at the evolution of pressure-perturbations [37]. Nested Tracking Graphs (NTG) [22], builds upon [36] and additionally allows viewing of the nesting hierarchy of components across different levels. Temporal Treemaps [19] optimises the layout of NTG, and adapts cushion treemaps for emphasising hierarchical relationships. The

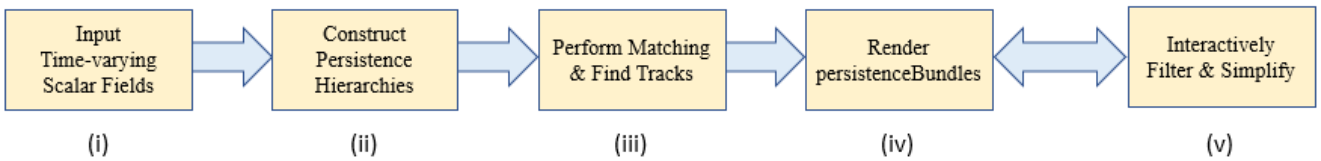


Figure 2: The persistenceBundles workflow. Process the scalar field at each time step, compute its persistence hierarchy, identify correspondences between features in successive time steps, track over time, and display the tracks using persistenceBundles. The user interacts with persistenceBundles to filter, simplify, and compare features.

NTG paradigm is popular in literature, and has also been extended for navigating Cinema databases [21]. FeatureFlow [3] utilizes the meta-graph abstraction, and introduced an interactive hierarchical river metaphor for summarizing feature evolution.

In contrast to the above mentioned techniques, we propose an approach to track all topological features rather than tracking only splits or merges across time. Topological features are represented as pairs of critical points of the scalar field, called persistence pairs. Such a scheme provides visual scalability, usage of native topological abstractions, flexibility in tracking methods, and enhanced interaction.

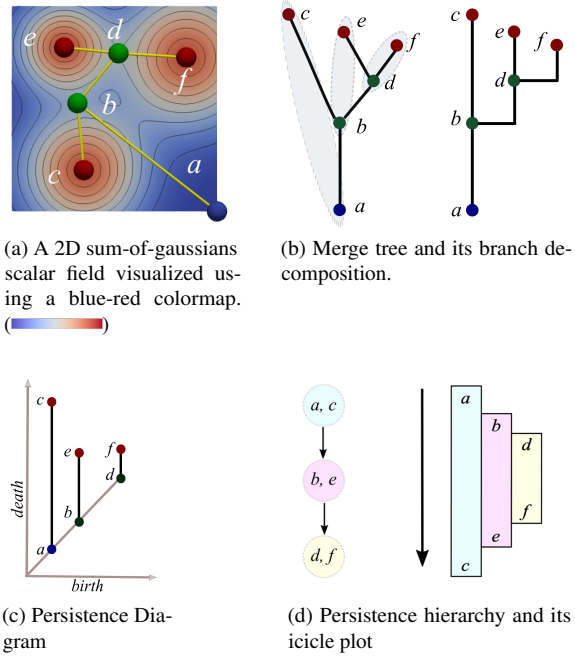
3 Persistence Hierarchy

In this section, we introduce the necessary terms and definitions for describing the construction of a hierarchy of topological features for a given scalar field, see Figure 3. We refer the reader to [31] for further details.

Consider a scalar function $f\colon D \to \mathbb{R}$ defined on a manifold domain D. A value c in the range of f is called an *isovalue*. Given an isovalue, an *isocontour* or *level set* is defined as the collection of all points $x \in D$ such that $f(x) = c$. A merge tree captures the connectivity of sub-level sets $f^{-1}(-\infty, c]$ (*join tree*) or super-level sets $f^{-1}[c, \infty)$ (*split tree*) of f. The split tree consists of maxima M = $\{M_i\}$, split saddles S = $\{s_j\}$, and the global minimum. The join tree consists of minima m = $\{m_i\}$, join saddles S = $\{s_k\}$, and the global maximum.

For the sake of convenience, we use only the split tree for explanation. All maxima can be paired with split saddles based on the notion of topological persistence, except for the global maximum which is paired with the global minimum. Each such pair represents a topological feature and its *persistence* can be defined as the absolute difference between the scalar function values at the two critical points. The persistence pair nodes are referred to as creator/birth (upper) and destroyer/death (lower). All such persistence pairs are collectively mapped on to the plane resulting in the zero-dimensional persistence diagram. Each persistence pair corresponds to a topological feature, and interchangeably referred to as a persistence feature.

A merge tree can be decomposed into a set of branches, such that each branch contains a persistence pair. This generates a nested hierarchy of branches, wherein each parent branch has a persistence greater than that of its children. This hierarchy of branches is called the *branch decomposition* representations of the merge tree [26]. While the branch decomposition is often displayed as a collection of L-shaped branches, it can also be represented as a rooted tree, whose nodes represents individual branches. Such a representation is described as the *persistence hierarchy* [27]. The persistence hierarchy is a hierarchical (or branched) representation of persistence pairs from the zero-dimensional persistence diagram. Henceforth, we refer to the zero-dimensional persistence diagram simply as the persistence diagram.



(a) A 2D sum-of-gaussians scalar field visualized using a blue-red colormap. ( )

(b) Merge tree and its branch decomposition.

(c) Persistence Diagram

(d) Persistence hierarchy and its icicle plot

Figure 3: Constructing the persistence hierarchy for a 2D scalar field.

4 Visual Comparison

We seek a good visual representation of the persistence diagram, in particular one that supports comparative visual analysis. A persistence pair is an ordered pair, where the scalar value of the creator is smaller than that of the destroyer. So, the persistence pairs are often visualized in the form of the persistence diagram, a scatter plot where the x- and y-axis correspond to birth and death respectively. The lower-right triangle of the plot is empty. The birth nodes are plotted on the *birth* = *death* line (*diagonal* of the persistence diagram), and the death nodes are plotted in the upper-left triangle. A vertical line is drawn between the death and birth nodes in order to emphasize the persistence of a pair as shown in Figure 3c. The persistence diagram appears as a hybrid between a bar chart and scatter plot representation. Alternatively, each persistence pair may be represented by the red points (death nodes) resulting in a simple scatter plot.

While the scatter plot representation of the persistence diagram has been effectively used in interactively identifying and filtering out low persistence features, the hybrid representation is potentially capable for interactive selection and annotation of specific persistence features. However, the hybrid representation can attract occlusion effects due to its cluttered nature in larger datasets. Moreover, here it may also be challenging to compare persistence features within

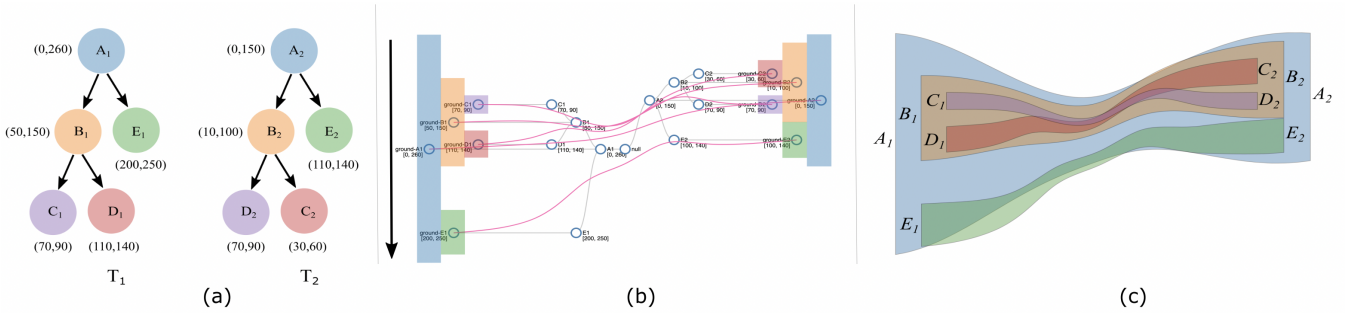


Figure 4: Comparing two simple persistence diagrams using persistenceBundles. (a) Two persistence hierarchies with the same structure but with different valued (birth, death) pairs. (b) Icicle plots, overlaid with the control-tree, and hierarchically bundled edges. Nodes and edges of the control-tree are shown in blue and grey respectively, while bundled edges between the matched icicles are in pink. (c) Modified 2D hierarchical edge bundling.

a diagram, since they are depicted as vertical bars over an inclined diagonal (in contrast to an orthogonal reference). From a design perspective, it seems unstinting to have a large triangular region left unutilized. Nevertheless, an alternate representation called the *persistence barcode* [9], is also used for representing persistence pairs and does not suffer from the above flaws. A persistence barcode lays out all persistence pairs horizontally one above another in ascending value of birth and persistence. However, this representation is not compact and the overall plot area can vary depending upon the count and value of persistence pairs in a dataset.

In general, current visual representations of persistence diagrams, are not readily amenable for comparative tasks. For instance, a common way of visually comparing two different persistence diagrams, is to overlay their scatter plots one upon another, display nodes of one diagram differently (using color or glyphs), and mark changes with lines or arrows. Sometimes, instead of overlapping diagrams, the nodes are animated across multiple frames for illustrating the same changes [17, 20] or persistence diagrams are juxtaposed after scaling nodes by persistence [34]. Such schemes do not scale with the size of the diagram. In particular, for time-varying scalar fields, it is infeasible to analyse a history of multiple transformations. Clearly, there is a need for an augmented or a different representation. We propose to represent the persistence diagram using the persistence hierarchy to address the shortcomings of earlier approaches.

5 Method Overview

Figure 2 presents an overview of our method. We describe each stage of the workflow in this section and highlight the flexibility of the method.

i. *Input time-varying scalar fields:* Scientific datasets are usually defined as a scalar field over a 2D/3D domain. We assume that a time-varying scalar field is available as a sequence of samples at discrete time steps. Multiple such datasets can be found at the SciVis Contest website [1].

ii. *Construct persistence hierarchies:* We first construct the merge trees and persistence diagrams of the scalar field at each timestep, using TTK plugins for Paraview [11, 32]. Next, we construct the branch decomposition of the merge tree, and hence the persistence hierarchy for each timestep, as described in Section 3.

iii. *Perform matching and find tracks:* Our first goal here is to compare the persistence hierarchies of two consecutive timesteps, and find a matching between their nodes. The nodes represent persistence pairs, and the persistence hierarchy represents the persistence diagram. There are many methods available in the literature to compare persistence diagrams and to compute a matching. Persistence diagrams can be compared by computing an optimal correspondence between the pairs. For example, the bottleneck distance [10] is the shortest distance b for which there exists a perfect matching between the points of the two diagrams such that two matched points are at distance at most b. The distance between points is given by the infinity norm in $\mathbb{R}^2$. Other distance measures such as the interleaving distance [23] and Wasserstein distance [5] are also widely used. A tree edit distance [15, 30] was introduced recently to compare merge trees and a weighted version of the Wasserstein distance [29] computes a matching with additional spatial constraints. Any of the above-mentioned or future methods that provide a one-to-one correspondence between the nodes can be utilized in our workflow.

Our second goal is to track a feature over time. After finding matches between all consecutive timesteps, we compute tracks by creating sequence of features that were matched to one another. For example, consider a feature f_1 in timestep t_1, matched to a feature f_2 in timestep t_2, and denoted by $f_1 \rightarrow f_2$. Further, say we compute another matching $f_2 \rightarrow f_3$, by comparing t_2 and t_3. Given a dataset with k timesteps, we may compute a sequence of features correspondences $f_1 \rightarrow f_2 \rightarrow \ldots \rightarrow f_k$. We call such a sequence as a track, and each matched feature in the dataset uniquely belongs to one such track. In general, a track does not span all timesteps and terminates when a subsequent matching cannot be found.

iv. *Render PersistenceBundles:* This is the core step of the workflow and described in detail in Section 6.

v. *Interactively filter and simplify:* We describe an interface for querying and visual analysis using PersistenceBundles in Section 8.

6 PersistenceBundles Design

The algorithm for constructing persistenceBundles follows from a tool for comparing hierarchical stack traces [33]. We adapt their algorithm for analyzing time-varying persistence hierarchies.

Our aim is to develop a representation that supports visual comparison of the persistence hierarchies of two consecutive timesteps of a time-varying dataset, and extend it to study all timesteps. Our design intuition stems from the fact that the (birth, death) interval of a node is contained within that of its parent. This property suggests the suitability of an icicle plot [18]. Each persistence pair denotes an icicle, rendered vertically with its upper-left coordinate equivalent to the pair's birth value and height equivalent to the pair's persistence, see Figure 3d. The two icicle plots are placed facing each other for comparison. A one-to-one correspondence between the icicles is

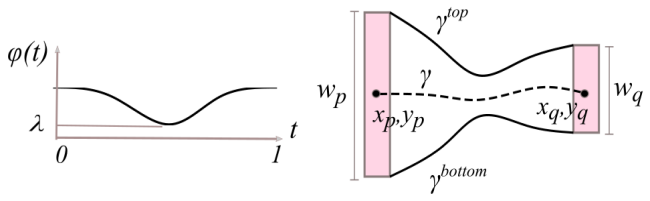


Figure 5: 2D tubes in persistenceBundles. We design a function ϕ that models the gradual thinning of the 2D tube used to represent match-edges.

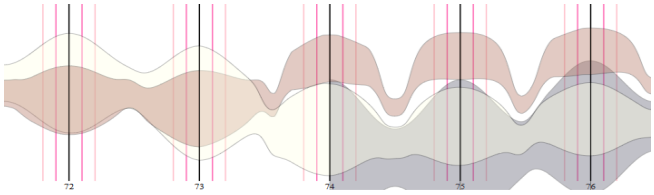


Figure 6: Tracks of large persistent features in the viscous fingers dataset between timesteps 72 - 76. An average persistence filter allows the user to remove insignificant tracks.

computed using one of the methods described in Section 5 (Step iii.) and displayed as match-edges.

Next, we observe that the hierarchical property of our abstraction can be wielded to reduce visual clutter. Hence, we use edge bundles to connect matched pairs in the two hierarchies. We follow the hierarchical edge bundling (HEB) design of Holten [12] and adapt it such that the match-edges are routed along a control-tree structure, which is computed using the icicle plot and mirrored along the y-axis. The match-edges however cannot be directly rendered, since HEB represents edges as 1D curves, making it difficult to locate important edges. To circumvent this, the match-edges are rendered as curved 2D tubes instead of 1D curves. We describe how these 2D tubes are computed below.

Consider two persistence pairs p and q that are connected by a match r. Let $(x_p, y_p), (x_q, y_q)$ denote the centres and h_p, h_q denote the height of the icicle plot rectangle for p and q, respectively. Let γ be the HEB curve that connects the centres. Let $t : [0,1]$ be an arc-length parameterization for γ. We construct two curves γ^{top} and γ^{bottom} which represent the top and bottom curved borders of our tube shape. If $\gamma^{top} = (\gamma_x^{top}(t), \gamma_y^{top}(t))$ and $\gamma^{bottom} = (\gamma_x^{bottom}(t), \gamma_y^{bottom}(t))$,

$$\gamma_x^{top}(t) = \gamma_x^{bottom}(t) = \gamma_x(t)$$
$$\gamma_y^{top}(t), \gamma_y^{bottom}(t) = \gamma_y(t) \pm \phi(t) \left((1-t)\frac{h_p}{2} + t\frac{h_q}{2} \right),$$

where $\phi(t) : [0,1] \rightarrow [0,1]$ is a function that models the gradual shrinking or thinning of the tube from its ends towards its center. We generate bundle-like tubes using

$$\phi(t) = \lambda + \frac{1-\lambda}{2}(1 + \cos 2\pi t),$$

where $\lambda \in [0,1]$ is the tube thickness, see Figure 5. The value of λ can be tuned to low values resulting in thinner tubes with reduced occlusion, or high values resulting in thicker tubes that show nesting properties.

The above method generates a visual metaphor for comparing two hierarchies, that we call as *persistenceBundles*. We compute persistenceBundles between all pairs of consecutive time steps and cascade them horizontally to study the evolution of features.

We use two small persistence diagrams to illustrate how persistenceBundles can be used to compare hierarchies, see Figure 4. Consider two persistence hierarchies T_1 and T_2 with the birth-death pairs and correspondences as shown. The hierarchies are represented as icicle plots facing each other, and their correspondences are hierarchically bundled along a control-tree structure, rendered as 2D tubes. The visual metaphor for the hierarchies shows the changes in terms of persistence – feature A shrinks in size, while the others remain almost unchanged. It also shows the nesting structure of the persistence pairs – features C and D get internally swapped between the two time steps.

7 Viscous Finger Example

The viscous finger dataset [2] describes instability that occurs at the interface between two mixing fluids of different viscosities. Scientists are typically interested in tracking how the fingers evolve across time. We subsample the dataset to a $64 \times 64 \times 64$ grid and track features using a Wasserstein distance based matching between the time steps. Since the matches and tracks are computed before hand, persistenceBundles between all pairs of consecutive timesteps can be computed and rendered in parallel. Due to the large number of levels, we render all icicle plot nodes with the same x-coordinate, while maintaining the hierarchy. Figure 1 provides a coarse-grained view to look at the gradual increase of all stable viscous fingers. We observe that the hierarchical bundling provides an overall aggregated understanding with reduced clutter.

8 Interaction

To preserve visual scalability, persistence features are hierarchically overlapped in our representation, and it can be cumbersome to select nested features. Hence, to interact with persistenceBundles, we require widgets and tools that support brushing + linking, rather than direct manipulation. Depending upon context, it is possible to have multiple designs for such widgets, and here we propose few designs that may be used in conjunction with persistenceBundles.

In a typical topological data analysis workflow, we often pose geometrical feature queries like 'tracks of top or bottom k-persistent features' or 'tracks with lengths in a specified interval'. Such queries are easy to support in our design, since each of the tracks are individually rendered, and can be collectively filtered to provide fine-grained views. Simple widgets like a range slider with single or multiple thumbs can be used for identifying thresholds. For instance, in Figure 6, using such widgets we filter a subset of tracks based on average persistence from the entire collection.

In order to identify such thresholds, it may be important to quickly browse through individual features. We propose the use of a donut widget (inspired by the iPod click wheel) linked to feature tracks. A selected track could then be highlighted and overlaid on top of other tracks with a prominent different color/illumination.

Another suggested widget is *mergemaps* [14], a treemap based representation of the persistence hierarchy, linked to the features of one timestep. This would be straightforward, since the underlying abstraction of both persistenceBundles and mergemaps are persistence hierarchies. Since mergemaps also provide an intuitive interface for individual feature exploration and simplification of hierarchies, such a scheme could be powerful in smaller datasets for comparing variation in discrete features. Spatial exploration of tracks would also be possible using a variation of mergemaps based upon spatially ordered treemaps [38].

Other potential directions include utilizing persistence rings [28], topological landscape profiles [25], and mergescapes [16] for high dimensional data analysis.

9 Conclusion

Existing visualization frameworks for tracking topological features based on comparison of merge trees like Nested Tracking Graphs, require both non-standard abstractions like the meta-graph and specialized tracking schemes dependent upon the abstraction. Moreover, they also display only a few carefully chosen isovalues for all time steps of a time-varying scalar field and provide less interactive capabilities.

In contrast, our persistenceBundles design, allows looking at all topological features, can flexibly track changes based on any approach for comparing persistence diagrams, provides hierarchical visual scalability, captures nesting, and fosters an ecosystem of interactive linked widgets for fine-grained and course-grained analysis. We provide a lightweight implementation of our approach at: **https://persistencebundles.github.io/viscous-fingers**

Acknowledgments

Vijay Natarajan is supported by a Swarnajayanti Fellowship from the Department of Science and Technology, India (DST/SJF/ETA-02/2015-16), a Mindtree Chair research grant, and the Robert Bosch Centre for Cyber Physical Systems, Indian Institute of Science, Bangalore. Adhitya sincerely thanks the gurudwaras of Vancouver and Glasgow for providing free meals.

References

[1] IEEE VIS Scientific Visualization Contest. http://sciviscontest.ieeevis.org/.

[2] Viscous Finger Dataset, 2016. http://www.uni-kl.de/sciviscontest/.

[3] Z. Bai, Y. Tao, and H. Lin. Featureflow: exploring feature evolution for time-varying volume data. *J. Visualization*, pages 927–940, 2019.

[4] Z. Bai, Y. Tao, and H. Lin. Time-varying volume visualization: a survey. *Journal of Visualization*, 06 2020.

[5] J. Berwald, J. M. Gottlieb, and E. Munch. Computing wasserstein distance for persistence diagrams on a quantum computer. *ArXiv*, abs/1809.06433, 2018.

[6] P. Bremer, G. Weber, V. Pascucci, M. Day, and J. Bell. Analyzing and tracking burning structures in lean premixed hydrogen flames. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):248–260, 2010.

[7] P. Bremer, G. H. Weber, J. Tierny, V. Pascucci, M. S. Day, and J. B. Bell. A topological framework for the interactive exploration of large scale turbulent combustion. In *IEEE International Conference on e-Science*, 2009.

[8] H. Doraiswamy, V. Natarajan, and R. S. Nanjundiah. An exploration framework to identify and track movement of cloud systems. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2896–2905, 2013.

[9] R. Ghrist. Barcodes: The persistent topology of data. Technical report, 2007.

[10] F. Godi. Bottleneck distance. In *GUDHI User and Reference Manual*. GUDHI Editorial Board, 3.2.0 edition, 2020.

[11] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Task-based augmented merge trees with fibonacci heaps. In *2017 IEEE 7th Symposium on Large Data Analysis and Visualization (LDAV)*, pages 6–15, 2017.

[12] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE TVCG*, 2006.

[13] A. Kamakshidasan. Spatial comparison of cricketers. *IEEE VIS Poster*, 2018.

[14] A. Kamakshidasan and V. Natarajan. Mergemaps: Treemaps for scientific data. TopoInVis, 2019.

[15] A. Kamakshidasan and V. Natarajan. Topological analysis of the 2D von kármán street. *IEEE VIS Poster*, 2019.

[16] A. Kamakshidasan and V. Natarajan. Understanding merge trees with force-directed landscapes. *IEEE VIS Poster*, 2019.

[17] M. Kramár, R. Levanger, J. Tithof, B. Suri, M. Xu, M. Paul, M. F. Schatz, and K. Mischaikow. Analysis of kolmogorov flow and rayleigh–bénard convection using persistent homology. *Physica D: Nonlinear Phenomena*, 334:82 – 98, 2016. Topology in Dynamics, Differential Equations, and Data.

[18] J. B. Kruskal and J. M. Landwehr. Icicle plots: Better displays for hierarchical clustering. *The American Statistician*, 37(2):162–168, 1983.

[19] W. Köpp and T. Weinkauf. Temporal treemaps: Static visualization of evolving trees. *IEEE TVCG*, 2019.

[20] R. Levanger. Studying fluid flows with persistent homology, 2018. https://youtu.be/cx1l65k0trg?t=635.

[21] J. Lukasczyk, C. Garth, G. H. Weber, T. Biedert, R. Maciejewski, and H. Leitte. Dynamic nested tracking graphs. *IEEE TVCG*, 2020.

[22] J. Lukasczyk, G. Weber, R. Maciejewski, C. Garth, and H. Leitte. Nested tracking graphs. *Computer Graphics Forum*, 2017.

[23] D. Morozov, K. Beketayev, and G. Weber. Interleaving distance between merge trees. *Discrete and Computational Geometry*, 49:22–45, 01 2013.

[24] P. Oesterling, C. Heine, G. H. Weber, D. Morozov, and G. Scheuermann. Computing and visualizing time-varying merge trees for high-dimensional data. In *Topological Methods in Data Analysis and Visualization*, pages 87–101. Springer, 2015.

[25] P. Oesterling, C. Heine, G. H. Weber, and G. Scheuermann. Visualizing nd point clouds as topological landscape profiles to guide local data analysis. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):514–526, 2012.

[26] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. Multi-resolution computation and presentation of contour trees. In *Proc. IASTED Conference on Visualization, Imaging, and Image Processing*, 2005.

[27] B. Rieck, H. Leitte, and F. Sadlo. Hierarchies and ranks for persistence pairs. Workshop on Topology-Based Methods in Visualization (TopoInVis), Feb. 2017.

[28] B. Rieck, H. Mara, and H. Leitte. Multivariate data analysis using persistence-based filtering and topological signatures. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2382–2391, 2012.

[29] M. Soler, M. Plainchault, B. Conche, and J. Tierny. Lifted wasserstein matcher for fast and robust topology tracking. In *8th IEEE Symposium on Large Data Analysis and Visualization, LDAV 2018, Berlin, Germany, October 21, 2018*, pages 23–33. IEEE, 2018.

[30] R. Sridharamurthy, T. B. Masood, A. Kamakshidasan, and V. Natarajan. Edit distance between merge trees. *IEEE Transactions on Visualization and Computer Graphics*, 26(3):1518–1531, March 2020.

[31] J. Tierny. *Topological Data Analysis for Scientific Visualization*. Mathematics and Visualization. Springer International Publishing, 2018.

[32] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. The Topology ToolKit. *IEEE Transactions on Visualization and Computer Graphics*, 2017. https://topology-tool-kit.github.io/.

[33] J. Trümper, J. Döllner, and A. Telea. Multiscale visual comparison of execution traces. In *ICPC*, 2013.

[34] J. Vidal, J. Budin, and J. Tierny. Progressive wasserstein barycenters of persistence diagrams. *IEEE Trans. Vis. Comput. Graph.*, 26(1):151–161, 2020.

[35] G. Weber, P.-T. Bremer, M. Day, J. Bell, and V. Pascucci. Feature tracking using reeb graphs. In *Topological Methods in Data Analysis and Visualization*, pages 241–253. Springer, 2011.

[36] W. Widanagamaachchi, C. Christensen, V. Pascucci, and P.-T. Bremer. Interactive exploration of large-scale time-varying data using dynamic tracking graphs. In *IEEE symposium on large data analysis and visualization (LDAV)*, pages 9–17. IEEE, 2012.

[37] W. Widanagamaachchi, A. Jacques, Bei Wang, E. Crosman, P. Bremer, V. Pascucci, and J. Horel. Exploring the evolution of pressure-perturbations to understand atmospheric phenomena. In *2017 IEEE Pacific Visualization Symposium (PacificVis)*, pages 101–110, 2017.

[38] J. Wood and J. Dykes. Spatially ordered treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1348–1355, 2008.