Geometry-Aware Merge Tree Comparisons for Time-Varying Data with Interleaving Distances

Lin Yan, Talha Bin Masood, Farhan Rasheed, Ingrid Hotz, Bei Wang

Abstract—Merge trees, a type of topological descriptors, serve to identify and summarize the topological characteristics associated with scalar fields. They have great potential for analyzing and visualizing time-varying data. First, they give compressed and topology-preserving representations of data instances. Second, their comparisons provide a basis for studying the relations among data instances, such as their distributions, clusters, outliers, and periodicities. A number of comparative measures have been developed for merge trees. However, these measures are often computationally expensive since they implicitly consider all possible correspondences between critical points of the merge trees. In this paper, we perform geometry-aware comparisons of merge trees using labeled interleaving distances. The main idea is to decouple the computation of a comparative measure into two steps: a *labeling* step that generates a correspondence between the critical points of two merge trees, and a *comparison* step that computes distances between a pair of labeled merge trees by encoding them as matrices. We show that our approach is general, computationally efficient, and practically useful. Our framework makes it possible to integrate geometric information of the data domain in the labeling process. At the same time, the framework reduces the computational complexity since not all possible correspondences have to be considered. We demonstrate via experiments that such geometry-aware merge tree comparisons help to detect *transitions*, *clusters*, and *periodicities* of time-varying datasets, as well as to *diagnose* and *highlight* the topological changes between adjacent data instances.

Index Terms—Merge trees, merge tree metrics, topological data analysis, topology in visualization

1 Introduction

The effective visualization of large and complex scientific data is an essential component of a modern analytics workflow. The key ingredients of this workflow are the identification, tracking, and comparison of features expressing essential structures in the data. To this end, *topological data analysis* has proven to provide fundamental tools for visual data analysis in terms of abstraction and summarization. Topological descriptors for scalar field data, such as persistence diagrams, barcodes, merge trees, contour trees, Reeb graphs, and Morse–Smale complexes, are among the most widely used applied topological tools in visualization. These descriptors have great potential for analyzing and visualizing time-varying data. First, they give compressed and topology-preserving representations of data instances. Second, their comparisons provide a basis for studying the relations among data instances, such as their distributions, clusters, outliers, and periodicities; see the work of Yan *et al.* [1] for a recent survey.

In this paper, we are interested in merge trees, which are topological descriptors that record the connectivity among the sublevel sets of scalar fields. Merge trees have seen many applications in science and engineering, including cyclone tracking [2], burning structure analysis [3], and symmetry extraction in materials science [4], [5], to name a few. To employ merge trees for time-varying data or ensembles, a key challenge is to choose an appropriate similarity or distance measure for their comparisons. A number of comparative measures have been developed for merge trees in the literature (see Sect. 2 and [1]), of which many effectively “forget” about the geometric information from the data domain in the comparative process.

- *Lin Yan and Bei Wang are with the University of Utah. E-mails: lynne.h.yan@gmail.com, beiwang@sci.utah.edu.*
- *Talha Bin Masood, Farhan Rasheed, and Ingrid Hotz are with Linköping University. E-mails: talha.bin.masood@liu.se, farhan.rasheed@liu.se, ingrid.hotz@liu.se.*

Manuscript received xx; revised xx.

We take a different perspective to utilize merge trees in studying time-varying data, and ask the following question: How can we design a merge tree comparative measure that integrates geometric information from the data domain? We hypothesize that by enriching a merge tree with geometrical information, a comparative measure defined on such enriched merge trees will be sensitive to local or global geometry and thus become beneficial for real-world applications where such geometry is important.

Our work is also motivated by the study of *information content* within a topological descriptor. A few recent efforts have linked information theory with topology (e.g., [6], [7], [8]). We are motivated by the following questions regarding the merge trees:

- How much geometric information could we add to a merge tree, so as to enrich its information content, while at the same time improving the comparative process among a pair of “enriched” merge trees in real-world applications?
- Will geometry-aware comparative measures for a set of merge trees improve our understanding of the time-varying data in terms of its distributions, clusters, and outliers?

We further illustrate the above thought process in Fig. 1. Formally, given a scalar field defined on a connected domain $f_{\mathbb{X}} : \mathbb{X} \to \mathbb{R}$, a merge tree records the connectivity of its sublevel sets and is represented as a pair (T, f), that is, a finite rooted tree T equipped with a function defined on its vertices $f : V(T) \to \mathbb{R}$; where f is a restriction of $f_{\mathbb{X}}$ to its critical points (see also Sect. 3). As we move from left to right in Fig. 1, we increase the geometric content of a topological descriptor for $f_{\mathbb{X}}$. We begin with the persistence barcode [9] of $f_{\mathbb{X}}$ in Fig. 1A. By adding slightly more geometric information, that is, how bars in the barcode are glued

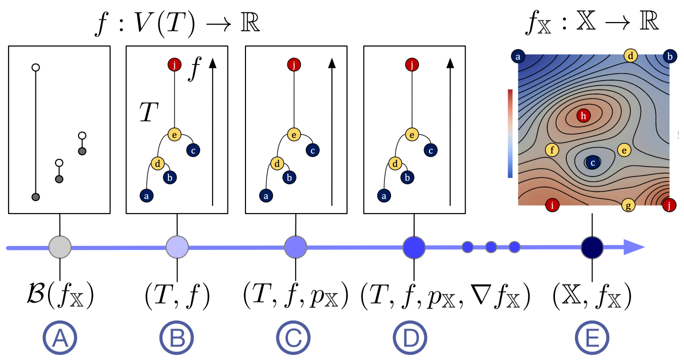
$f : V(T) \to \mathbb{R}$
$f_{\mathbb{X}} : \mathbb{X} \to \mathbb{R}$
T
f
$\mathcal{B}(f_{\mathbb{X}})$
(T, f)
$(T, f, p_{\mathbb{X}})$
$(T, f, p_{\mathbb{X}}, \nabla f_{\mathbb{X}})$
$(\mathbb{X}, f_{\mathbb{X}})$
A
B
C
D
E

Fig. 1. Given a scalar field $f_{\mathbb{X}} : \mathbb{X} \to \mathbb{R}$, from left to right, we increase the geometric content of the topological descriptors associated with $f_{\mathbb{X}}$: (A) the barcode of $f_{\mathbb{X}}$; (B) the "classic" merge tree (T, f); (C) adding the coordinates of critical points $p_{\mathbb{X}}$ of $f_{\mathbb{X}}$; (D) adding the gradient information $\nabla f_{\mathbb{X}}$ of $f_{\mathbb{X}}$; (E) the original scalar field data.

together, we obtain the "classic" merge tree of $f_{\mathbb{X}}$, denoted as (T, f) in Fig. 1B. The connection between barcodes and merge trees first appeared (rather implicitly) via the Elder Rule in [10, Page 150], and was further explored in [11], [12], [13]. Although many existing comparative measures on merge trees rely on such a classic definition of merge trees, the focus of this paper is to develop comparative measures on merge trees that encode the geometry of the data domain. Specifically, we focus on enriching a merge tree by encoding the coordinates of critical points $p_{\mathbb{X}}$ of $f_{\mathbb{X}}$ in Fig. 1C. We also can add the gradient information $\nabla f_{\mathbb{X}}$ in Fig. 1D. Finally, a merge tree can be enriched with the scalar field $f_{\mathbb{X}}$ itself, possibly rendering the tree redundant in Fig. 1E.

Our objective is to perform geometry-aware comparisons of merge trees. We explore a general notion of merge tree called the *leaf labeled merge tree* [14], [15], which is an abstract tree equipped with a scalar function and a labeling of its leaves. Our main idea is to *decouple* the computation of a comparative metric for a pair of (enriched) merge trees into two steps:

1. A *labeling* step that generates a correspondence between the critical points of two merge trees using various geometric information of the data domain;
2. A *comparison* step that computes distances between pairs of labeled merge trees represented as matrices.

The labeling step makes it possible to integrate geometric information of the data domain, such as the locations of critical points and the gradients of the underlying scalar fields. It also allows the encoding of application-specific domain knowledge. Furthermore, it reduces the computational complexity since not all possible correspondences have to be considered. We provide several heuristic strategies for labeling leaves in a merge tree, namely, *tree mapping*, *Euclidean mapping*, and their *hybrid mapping*. We also discuss a *Morse mapping* strategy based on the gradients of an input scalar field. For the comparison step, we use the *labeled interleaving distance* [14], which encodes the labeling information within matrix representations of merge trees.

Using datasets in scientific simulations, we experimentally evaluate and compare our framework against well-established comparative measures for merge trees, using the bottleneck distance and tree edit distance [16] as the baseline. In summary:

- We provide a general and unifying two-step framework that supports geometry-aware comparisons of merge trees for time-varying data;
- We demonstrate that our proposed framework can help detect *transitions*, *clusters*, and *periodicities* of a time-varying dataset, as well as to *diagnose* and *highlight* the topological changes between adjacent data instances.

Finally, our framework is open source at https://github.com/tdavislab/MergeTreeMetric.

2 Related Work

A number of topological descriptors serve to describe and identify the topological characteristics associated with a data instance (e.g., a scalar field, a vector field, a tensor field, or a multifield), see [17] for a survey. In this paper, we focus on merge trees for scalar fields, which are highly relevant in topology-based feature tracking (e.g. [18], [19], [20], [21]). They have been used to identify and track cyclones [2] and bubbles in Raleigh-Taylor instabilities [22], as well as to analyze burning cells [3] from combustion simulations.

We explore scalar fields by comparing their corresponding topological descriptors, which requires a measure of similarity or dissimilarity between them, see [1] for a survey. For merge trees and their variants, a number of metrics have been developed (e.g. [16], [23], [24], [25], [26], [27]). However, many of these metrics remain theoretical and do not have practical implementations. The most relevant work is by Sridharamurthy *et al.* [16], who introduced an edit distance between merge trees that admits efficient computation. Their edit distance is defined as the minimum cost of a set of restricted edit operations (e.g., delete, insert, and relabel) that transforms one merge tree into another. Pont *et al.* [28] extended the above edit distance and introduced a Wasserstein distance between merge trees for feature tracking and ensemble clustering.

Enhancing topological trees with geometric information is not new. For example, Beketayev *et al.* [29] used geometry information for the generation of a topological landscape. They established a correlation between the representation of a contour tree and the geometric proximity of the topological features based on dimension reduction techniques. This representation could be used for pairwise visual comparison of scalar fields defined over the same domain. Lohfink *et al.* [30] introduced the notion of a fuzzy contour tree, which provides a joint layout of an ensembles of contour trees. They used a contour tree edit distance to establish a matching between the nodes and arcs of the tree. Herick *et al.* [31] recently introduced a temporally coherent layout of contour trees. They used a weighted sum of spatial and topological distances to match topological trees from consecutive time steps.

From a methodological point of view, the works of Gasparovic *et al.* [14] and Yan *et al.* [15] are most relevant to the current paper. Gasparovic *et al.* [14] introduced an easily computable metric called the *labeled interleaving distance* that can be used to compare labeled merge trees. Yan *et al.* [15] adapted such a distance in practice for computing average merge trees and visualizing uncertainty. They introduced a few heuristic strategies that generate correspondences between merge trees, which set the foundation for the labeling step in our framework. Compared with [15], we perform a more systematic comparative study on how geometry-aware comparative measures for merge trees improve our understanding of time-varying data under various visualization tasks, including the detection of transitions, clusters, and periodicities. We further introduce time-varying pivot tree and dummy vertex strategies, which are shown to be more effective in the study of the time-varying data, compared with the global pivot tree and dummy leaf strategies first introduced by Yan *et al.* [15].

Finally, merge trees and Morse complexes are closed connected in terms of studying time-varying structures. Such connections provide further justifications of using merge trees to infer the similarity between scalar fields in a time sequence. Tracking spatiotemporal changes in time-varying scalar fields typically requires the study of features surrounding critical points in terms of their appearances, disappearances, merging and splitting. Morse (and Morse–Smale) complexes are natural topological descriptors for feature tracking since they induce a domain segmentation and capture the relative geometric positions of critical points. In particular, Bremer *et al.* [32] demonstrated how features identified by thresholding iso-surfaces can be defined in terms of the Morse complex. They utilized gradient behavior captured by a Morse complex to identify features and track these features across time using a Reeb graph (a variant of the merge tree) constructed in the space-time domain. A follow-up work by Bremer *et al.* [33] utilized hierarchical merge trees (instead of Morse complexes) to provide flexible feature representations. As discussed in [33], a Morse complex encodes gradient-based features whereas a merge tree captures threshold-based ones. Therefore, "a Morse complex needs a secondary data structure within each Morse cell to encode feature segmentation, while the merge tree naturally provides this information" [33]. Merge trees can also be computed more efficiently than a Morse complex. In summary, merge trees provide a more concise and effective representation of the topological relationships among critical points in practice. Thus, introducing geometry-aware schemes is a natural approach for enhancing the expressive capability of such merge trees.

3 Background

In this section, we review the necessary background on scalar field topology surrounding the notions of merge trees, labeled merge trees, and distances between the labeled merge trees.

3.1 Merge Trees and Their Variants

Scalar-field-induced merge trees. Given a scalar field $f : \mathbb{X} \to \mathbb{R}$ defined on a connected domain $\mathbb{X}$, a (scalar field induced) merge tree records the connectivity of its sublevel sets. Two points $x, y \in \mathbb{X}$ are considered *equivalent w.r.t.* f, $x \sim y$, if they have the same function value, that is, $f(x) = f(y) = a$, and if they belong to the same connected component of the sublevel set $\mathbb{X}_a := f^{-1}(-\infty, a]$, for some $a \in \mathbb{R}$. A *merge tree* is the quotient space $\mathbb{X}/\sim$ obtained by gluing together points in $\mathbb{X}$ that are equivalent under the relation $\sim$. Intuitively, it keeps track of the evolution of connected components in $\mathbb{X}_a$ as a increases; see Fig. 2B for an example. Specifically, leaves (*i.e.*, nonroot vertices with degree 1) in a merge tree represent the creation of a component at a local minimum, internal vertices (of degree ≥ 3) represent the merging of components, and the root (a degree 1 vertex) represents the entire space as a single component.

Throughout this paper, we denote our data of interest as a pair $(\mathbb{X}, f)$, that is, a connected topological space $\mathbb{X}$ together with a scalar field $f : \mathbb{X} \to \mathbb{R}$. The quotient space $\mathbb{X}/\sim$ is a new topological space that effectively "forgets" about certain information regarding the data $(\mathbb{X}, f)$ such as the locations of the critical points and the gradient of f.

Labeled merge trees. In this paper, we work with a more general notion of merge trees defined below, which can be considered as an abstract tree coupled with a function f on its vertices, while being oblivious of the possible existence of a scalar field that gives rise to the merge tree.

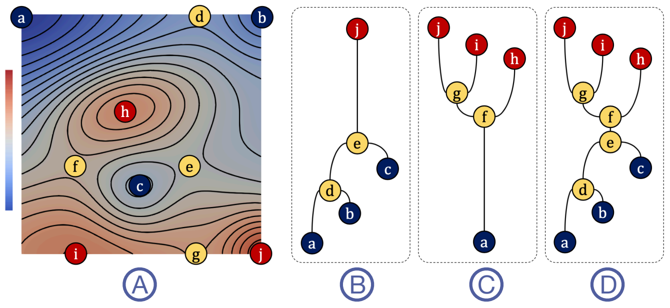

Fig. 2. For the scalar field f in (A): the merge tree of f (*i.e.*, the join tree), the merge tree of $-f$ (*i.e.*, the split tree), and the contour tree of f are illustrated in (B), (C), and (D), respectively.

Definition 3.1. A *merge tree* is a pair (T, f) of a finite rooted tree T with vertex set $V(T)$ and a function $f : V(T) \to \mathbb{R} \cup \{\infty\}$ such that (i) adjacent vertices do not have equal function value, (ii) every nonroot vertex has exactly one neighbor with a higher function value, and (iii) the root is the only vertex with the value ∞ [14, Def. 2.1].

Def. 3.1 is closely related to that of treegrams [34], which is a certain generalization of a dendrogram [35]. Specifically, we focus on the notion of a labeled merge tree. Let $[n]$ denote a set of labels $\{1, 2, \cdots, n\}$.

Definition 3.2. A *labeled merge tree*, denoted as a triple $\mathcal{T} = (T, f, \pi)$, consists of a merge tree (T, f) together with a labeling $\pi : [n] \to V(T)$ that is surjective on the set of leaves [14, Def. 2.2].

π is not required to be injective; thus, a vertex can have multiple labels. π also allows labels for nonleaves by thinking of these as degenerate labeled leaves [14]. For this paper, we work with *leaf-labeled merge trees*, where $L(T) \subset V(T)$ represents leave set of the tree. Unless otherwise specified, they are referred to as labeled merge trees for the remainder of the paper. We study the space of labeled merge trees that share the same label set $[n]$.

Join and split trees. For convenience, we refer to the merge tree of f as the *join* tree and the merge tree of $-f$ as the *split* tree (following the convention in [36]). A join tree (Fig. 2B) tracks the connected component of the sublevel sets of f and a split tree (Fig. 2C) tracks that of the superlevel sets of f. The leaves of a join tree are local minima of f, and the leaves of a split tree are local maxima of f. Combining a join tree and a split tree of f gives rise to its *contour tree* [36] (Fig. 2D), which captures the connectivity among level sets. As illustrated in Sect. 5, studying the join tree or the split tree of time-varying data offers different perspectives on their topological signatures.

Merge trees and Euclidean distances between vertices. Given an unlabeled merge tree (T, f), the *intrinsic tree distance* d_t between pairs of vertices is induced by $f : V(T) \to \mathbb{R}$. For any pair of vertices $x, y \in V(T)$, it is defined as

$$d_t(x,y) = |f(x) - f(lca(x,y))| + |f(y) - f(lca(x,y))|,$$

where $lca(x,y)$ denotes the lowest common ancestor of x and y in T. In other words, $d_t(x,y)$ captures the shortest path between x and y measured by function value differences to their lowest common ancestor.

On the other hand, let $\tau_i : |T^i| \to \mathbb{R}^2$ denote the geometric embeddings of T_i into the spatial domain (for $i = 1,2$). The *Euclidean distance* d_e between a pair of vertices $x, y \in V(T)$ is induced by its geometric embedding τ. It is defined as

$$d_e(x,y) = ||\tau(x) - \tau(y)||_2.$$

3.2 Interleaving Distances Between Merge Trees

A number of metrics may be defined on the space of labeled merge trees. In fact, any metric defined on unlabeled merge trees may be extended to labeled ones by forgetting the label information, which likely turns a metric into a pseudometric [14].

For a labeled merge tree, again let $lca(x,y)$ denote the *lowest common ancestor* of a pair of vertices. We have $lca(x,x) = x$. Let $f(lca(x,y))$ denote its function value.

The *induced matrix* of a labeled merge tree $\mathcal{T} = (T, f, \pi)$, denoted as $M(T, f, \pi)$, is the matrix $M_{ij} = f(lca(\pi(i), \pi(j)))$ [14, Def. 2.6]. As shown in Fig. 3A-B, the induced matrices of labeled merge trees T^1 and T^2 with a shared label set $[3] := \{1,2,3\}$ are defined as follows:

$$M^1 = \begin{bmatrix} 2.0 & 4.7 & 5.2 \\ \cdot & 3.0 & 5.2 \\ \cdot & \cdot & 1.0 \end{bmatrix}, M^2 = \begin{bmatrix} 2.0 & 5.2 & 5.2 \\ \cdot & 3.0 & 4.7 \\ \cdot & \cdot & 1.0 \end{bmatrix}.$$

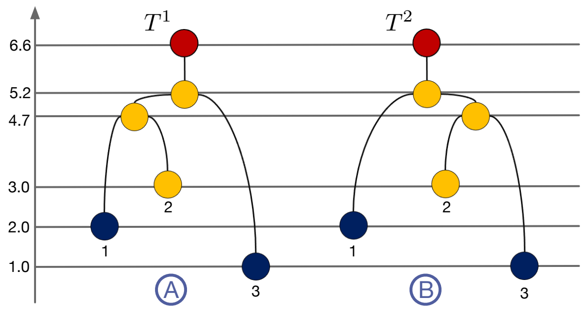

Fig. 3. Labeled merge trees T^1 and T^2 with a shared label set $[3] := \{1,2,3\}$.

$M \in \mathbb{R}^{n \times n}$ is symmetric; therefore, it is common to store M as an upper triangular matrix with $n(n+1)/2$ nonzero entries. We use such a convention in our paper. The nonzero terms in M can be linearized into a vector of length $n(n+1)/2$, called the *cophenetic vector* of $\mathcal{T}$ [37].

Turning a labeled merge tree into a matrix (or a vector) enables us to use distances between matrices (or vectors) to obtain distances between trees. We work with the *cophenetic metrics* first introduced by Cardona *et al.* [37] for phylogenetic trees.

Definition 3.3. Given two labeled merge trees $\mathcal{T}^1 = (T_1, f_1, \pi_1)$ and $\mathcal{T}^2 = (T_2, f_2, \pi_2)$ that share the same set of labels $[n]$, the L^∞-, L^1-, and L^2-cophenetic metrics are defined, respectively, as distances between their corresponding induced matrices M^1 and M^2 [37]:

$$d^\infty(\mathcal{T}^1, \mathcal{T}^2) = ||M^1 - M^2||_\infty,$$
$$d^1(\mathcal{T}^1, \mathcal{T}^2) = ||M^1 - M^2||_1,$$
$$d^2(\mathcal{T}^1, \mathcal{T}^2) = ||M^1 - M^2||_2.$$

$||M||_\infty$, $||M||_1$ and $||M||_2$ denote the vector norms, that is, $||M||_\infty = \max_{ij} |M_{ij}|$, $||M||_1 = \sum_{ij} |M_{ij}|$ and $||M||_2 = \sqrt{\sum_{ij} |M_{ij}|^2}$. In other words, the above distances correspond to the L^∞, L^1, and L^2 distances between the cophenetic vectors of $\mathcal{T}^1$ and $\mathcal{T}^2$, respectively. d^∞, d^1, and d^2 work only on labeled merge trees since we need the (same set of) labels to have a well-defined induced matrix.

d^∞ is also referred to as the *labeled interleaving distance*, denoted as d_I, between a pair of labeled merge trees [14, Def. 2.13] (and subsequently [15, Def. 3.3]), due to its close connection to the interleaving distance of persistence modules [38], [39]. Such a connection is described explicitly in [40]. The interleaving distance of persistence modules has been adapted to merge trees [23], [41], Reeb graphs [42], [43], and Reeb spaces [44] via a category-theoretic perspective [45], [46]. d_I has also been used to compare single-linkage hierarchical clustering [34]. Recent work has established both theoretical [14] and algorithmic [14], [15] foundations for the space of merge trees under the interleaving distances, including computing a form of structural averages for uncertainty visualization [15]. For the remainder of this paper, we work with the interleaving distance d_I with the understanding that $d_I := d^\infty$ for labeled merge trees. For example, as illustrated in Fig. 3, the interleaving distance $d_I(\mathcal{T}^1, \mathcal{T}^2) = ||M^1 - M^2||_\infty = 0.5$. One of the main advantages of the labeled interleaving distance d_I is that, as an L^∞ type distance, it helps diagnose which entries in the induced matrices are likely responsible for the given distance, as shown in Sect. 5.

3.3 Other Distances

We review a few other distance metrics that are applicable for labeled merge trees. Recall any metric defined on unlabeled merge trees is applicable by ignoring the labeling.

Bottleneck and 1-Wasserstein distance. Given two persistence diagrams X^1, X^2 and a bijection $\eta : X^1 \to X^2$, the *bottleneck distance* [47] between X^1 and X^2 is defined as

$$W_\infty(X^1, X^2) = \inf_{\eta: X^1 \to X^2} \sup_{x \in X^1} ||x \in \eta(x)||_\infty. \quad (1)$$

The *q-Wasserstein distance* [10, page 183] is

$$W_q(X^1, X^2) = \left[\inf_{\eta: X^1 \to X^2} \sum_{x \in X^1} ||x \in \eta(x)||_\infty^q \right]^{1/q}. \quad (2)$$

By definition, W_q becomes W_∞ by setting $q = \infty$. In dimension zero, which is the concern of this paper, there is a correspondence between a merge tree T of f and the persistence diagram X of its sublevel set filtration. Specifically, the branch decomposition of T of f gives rise to the points in the persistence diagram of f. More generally, given a merge tree (T, f) in the sense of Def. 3.1, we could directly define the *persistence diagram of the merge tree*, denoted as X_T, by treating T as a topological space and computing the 0-dimensional persistence of its sublevel set filtration, following a similar construction in [25]. Therefore, we can define the bottleneck distance between the merge trees as the bottleneck distance between the persistence diagrams of the merge trees, that is,

$$d_B(\mathcal{T}^1, \mathcal{T}^2) = W_\infty(X_T^1, X_T^2). \quad (3)$$

Similarity, we work with 1-Wasserstein distance between the merge trees,

$$d_W(\mathcal{T}^1, \mathcal{T}^2) = W_1(X_T^1, X_T^2). \quad (4)$$

Edit distance between merge tree. The *edit distance* between merge trees [16] is defined as

$$d_E(\mathcal{T}^1, \mathcal{T}^2) = \min_S \{\gamma(S)\}, \tag{5}$$

where S is a tree edit operation sequence from T^1 to T^2 that includes edit operations such as "relabel", "delete", and "insert"; and γ is a cost function that assigns a non-negative real number to each operation. d_E is an adaptation of and a significant improvement on the constrained unordered tree edit distance [48].

Distances between scalar fields. Finally, there are a number of comparative measures for scalar fields (e.g., [49], [50], [51]). We are given a pair of scalar fields f_1 and f_2; each sampled at N locations $\{x^1, \dots, x^N\} \in \mathbb{X}$, forming a vector of length N, denoted as $v_1 = (v_1^1, \dots, v_1^N)$ and $v_2 = (v_2^1, \dots, v_2^N)$, respectively. We work primarily with Euclidean distances between them in the comparative study, defined as

$$d_F(f_1, f_2) = ||v_1 - v_2||_2.$$

We compare against three additional distances in Sect. 5.4 that are based on Pearson correlation, gradient, and iso-surfaces, respectively. First, the Pearson correlation coefficient between v_1 and v_2 (by treating them as distributions) is defined as

$$\rho(v_1, v_2) = \frac{\sum_{i=1}^N (v_1^i - \bar{v}_1)(v_2^i - \bar{v}_2)}{\sqrt{\sum_{i=1}^N (v_1^i - \bar{v}_1)^2}\sqrt{\sum_{i=1}^N (v_2^i - \bar{v}_2)^2}},$$

where $\bar{v}_1 = (1/N)\sum_{i=1}^N v_1^i$, $\bar{v}_2 = (1/N)\sum_{i=1}^N v_2^i$. We work with a normalized Pearson's distance [51] defined from their correlation coefficient as

$$d_P(f_1, f_2) = \frac{1}{2}(1 - \rho(v_1, v_2)).$$

By definition, $d_P(f_1, f_2) \in [0, 1]$ with $d_P(f_1, f_2) = 0$ when $v_1 = v_2$.

Second, we utilize a gradient-based comparative measure introduced by Nagaraj *et al.* [50], which is defined as the matrix norm of a partial derivative matrix involving the two fields. Formally, given two scalar fields f_1 and f_2 defined on a shared 2D domain $\mathbb{X}$, the matrix of partial derivatives at a point $x^i = (x_1^i, x_2^i) \in \mathbb{X}$ is

$$J(x^i) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1^i}(x^i) & \frac{\partial f_1}{\partial x_2^i}(x^i) \\ \frac{\partial f_2}{\partial x_1^i}(x^i) & \frac{\partial f_2}{\partial x_2^i}(x^i) \end{bmatrix}.$$

Our comparative measure d_G is then defined by averaging the norm of partial derivatives of all N points $\{x^1, \dots, x^N\} \in \mathbb{X}$ in the domain,

$$d_G(f_1, f_2) = \frac{1}{N}\sum_{i=1}^N ||J(x^i)||.$$

Finally, we also consider the work by Fofonov and Linsen [51], who generalized an iso-surface similarity measure. Assuming the range of f_1 and f_2 is normalized to $[0, 1]$, the distance d_S between them is defined as

$$d_S(f_1, f_2) = 1 - \frac{\sum_{i=1}^N (1 - \max(v_1^i, v_2^i))}{\sum_{i=1}^N (1 - \min(v_1^i, v_2^i))}.$$

By definition, $d_S(f_1, f_2) \in [0, 1]$, with $d_S(f_1, f_2) = 0$ when $v_1 = v_2$.

4 Geometry-Aware Merge Tree Metrics

Most of the merge tree metrics described in Sect. 2 are unaware of the geometric information from the data domain, such as the locations of the critical points or the gradient of the scalar field. These metrics are defined over the space of *unlabeled merge trees*. To define a geometry-aware metric, we consider a more general space that allows for encoding such information. A suitable space is the space of *labeled merge trees* reviewed in Sect. 3. Our two-step framework for merge tree comparisons includes:

1. *Labeling*. This step generates a correspondence between the critical points of two merge trees by encoding the geometric information of the data domain. We describe several labeling strategies – namely, *tree mapping*, *Euclidean mapping*, and *hybrid mapping* – based on topological and geometric information of the scalar fields, respectively.
2. *Comparison*. This step transforms labeled merge trees into their *induced matrices*, and computes distances between labeled merge trees by computing distances between their corresponding induced matrices.

In this section, we describe the labeling step in detail. Suppose we are given a time-varying dataset consisting of l instances of scalar fields defined over a common domain with their corresponding merge trees. There are two requirements to transfer a set of unlabeled merge trees into labeled ones. First, we need to assign a common set of labels between leaves of merge trees, that is, to find leaf correspondences that capture properties of the data domain; see Sect. 4.1. Second, we need to ensure the induced matrices of labeled merge trees are comparable. We introduce *dummy vertices* and *dummy leaves* to the trees to ensure that the resulting induced matrices are the same size. The dummy leaf strategy was first introduced in [15]; we perform a more systematic study of both strategies in Sect. 4.2. Finally, we introduce the notion of a *time-varying pivot tree*, which is shown to be effective to capture topological transitions in a time-varying setting; see Sect. 4.3.

4.1 Labeling Strategies

Our leaf labeling strategies take as input a pair of unlabeled merge trees T^1 and T^2 that arise from a pair of 2D scalar fields f_1 and f_2. We consider a tree mapping strategy, a Euclidean mapping strategy, and a hybrid mapping strategy, where the hybrid mapping strategy generalizes the other two strategies.

Given an unlabeled merge tree (T, f), recall that the intrinsic tree distance between a pair of vertices x and y in the tree is denoted by d_t, and the Euclidean distance between their geometric embeddings is denoted by d_e. d_t and d_e capture the topological and geometric relation between x and y, respectively. To achieve a balance between topology and geometry, we define a hybrid distance as

$$d_h(x, y) = \lambda \cdot d_t + (1 - \lambda) \cdot d_e,$$

for $0 \leq \lambda \leq 1$. d_h generalizes both d_t and d_e. For $\lambda = 1$, $d_h = d_t$ and for $\lambda = 0$, $d_h = d_e$. We thus focus on describing the hybrid mapping strategy, which finds a minimum cost matching between leaves based on their similarities among their d_h distances to other vertices in the tree.

Initial label assignment. Given a pair of unlabeled merge trees T^1 and T^2, let $V(T^1)$ and $V(T^2)$ denote their respective vertex sets, and $L(T^1)$ and $L(T^2)$ their leaf sets. The goal of an initial label assignment between $L(T^1)$ and $L(T^2)$ is to utilize topology, geometry, or prior knowledge of the data to establish initial correspondences (labels) between subsets of the leaves that share high similarities. These initially matched labels serve as "anchors" during the labeling process, where distances to these anchors are used to assess topological and geometric similarities among the remaining unlabeled leaves.

We use geometric proximities of critical points in the domain under the Euclidean distance as an example. To define a shared label set, the tree with a larger number of leaves is chosen as the *pivot tree* T_p. The leaves of T_p give rise to a *pivot label set*, denoted $S_p := [n]$, where $n = |L(T_p)|$. Let $\pi_p : S_p \to L(T_p)$ denote a labeling of the pivot tree; *w.l.o.g.*, assume $T_p = T^1$ and π_1 is its leaf labeling. This label set is assigned to $L(T^2)$ via $\pi_2 : S_p \to L(T^2)$ with a minimum weight matching described below.

To initialize a labeling of $L(T^2)$, we assign a subset of labels in S_p to $L(T^2)$ based on Euclidean distances between $L(T^2)$ and $L(T^1)$ in the embeddings. To do so, we construct a weighted, complete bipartite graph between $L(T^1)$ and $L(T^2)$ where the weight w_{xy} between $x \in L(T^1)$ and $y \in L(T^2)$ is their Euclidean distance in the domain; $w_{xy} = d_e(\tau_1(x), \tau_2(y))$. To solve an *assignment* problem of this bipartite graph, we find a matching with a maximum number of edges in which the sum of edge weights is as small as possible, which gives rise to an initial label assignment of leaves in T^2.

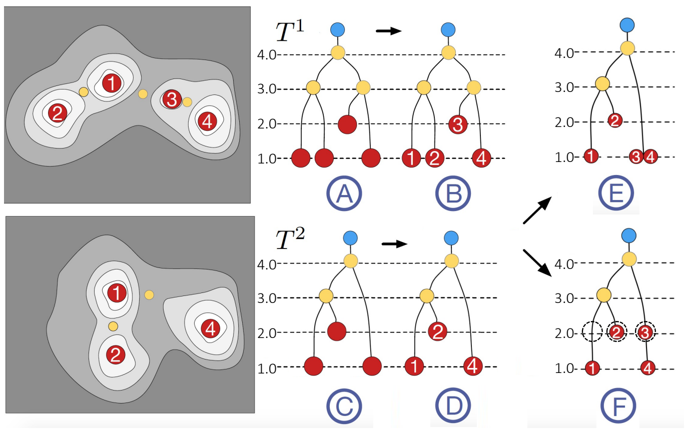

Fig. 4. A labeling of leaves in T^2 (C) against a pivot tree T^1 (A) under the tree mapping strategy, using dummy leaves (E) or dummy vertices (F).

We illustrate this initial label assignment in Fig. 4. We start with a pair of merge trees T^1 and T^2 that arise from a pair of scalar fields. Since $|L(T^1)| = 4 > |L(T^2)| = 3$, T^1 in Fig. 4A is chosen to be the pivot tree, which gives rise to a pivot label set $S_p = \{1,2,3,4\}$, see Fig. 4B. Leaves of T^2 (Fig. 4C) obtain their initial labels $\{1,2,4\}$ by solving the above assignment problem, since the respective leaves are close to one another in the domain; see Fig. 4D. Thus, there is an unmatched label $U_1 = \{3\}$ for T^1.

4.2 Dummy Leaves and Dummy Vertices

In general, after the initial label assignment, there may be unmatched labels in the pivot tree; for example, the label $\{3\}$ remains unmatched for T^2 in Fig. 4D. The next step is to create dummy labels on T^2 to ensure that it uses the entire pivot label set. We introduce both *dummy leaf* and *dummy vertice* strategies. The former allows dummy labels only on the leaves, whereas the latter allows dummy labels on the interior of edges.

Dummy leaves. The dummy leaf strategy [15] finds an assignment for each unmatched label in the pivot tree T^1 by duplicating leaves in T^2. It uses a greedy assignment strategy based on the pairwise distance matrices. The main idea is to find a leaf in T^2 that has a local structure most similar to an unmatched leaf in T^1.

Using the example from Fig. 4E, T^1 has an unmatched label 3. The d_t-based pairwise distance matrix between unmatched labels ($\{3\}$) and matched labels in T^1 $\{1,2,4\}$ is $D_1 = \begin{pmatrix} 5.0 & 5.0 & 3.0 \end{pmatrix}$. The d_t based pairwise distance matrix of T^2 encodes distances between leaves $\{1,2,4\}$ and matched labels $\{1,2,4\}$, forming $D_2 = \begin{pmatrix} 0.0 & 3.0 & 6.0 \\ 3.0 & 0.0 & 5.0 \\ 6.0 & 5.0 & 0.0 \end{pmatrix}$. To find a leaf in T^2 that has the most similar local structure to the unmatched label 3 in T^1, rows from D_1 and D_2 are compared, which leads to a matching of leaf 3 in T^1 to leaf 4 in T^2. T^2 now contains one leaf with two labels $\{3,4\}$; see Fig. 4E.

Dummy vertices. The dummy leaf strategy is well-suited to generate smooth transitions between merge trees [15], but it leads to instabilities in the distance computation. Therefore, we describe a second strategy that adds dummy vertices internal to the branches, which can also be interpreted as adding a branch with zero length. Thus, this strategy ensures that the dummy vertex does not change the tree structure of T^2. To determine the branch in T^2 where we add a dummy vertex, a pairwise distance matrix for all candidates is computed and compared to the distances in T^1.

Let us revisit the example in Fig. 4F. A dummy vertex with a label 3 is added to T^2 with the same scalar value as the leaf 3 of T^1. This dummy vertex has three candidates, which are highlighted by dashed circles in Fig. 4F. The d_t-based pairwise distance matrix between these candidates and matched labels $\{1,2,4\}$ in T^2 is $D_2 = \begin{pmatrix} 1.0 & 2.0 & 5.0 \\ 3.0 & 0.0 & 5.0 \\ 5.0 & 4.0 & 1.0 \end{pmatrix}$. Compared with $D_1 = \begin{pmatrix} 5.0 & 5.0 & 3.0 \end{pmatrix}$, the third candidate has the local structure most similar to leaf 3 of T^1. The result is shown in Fig. 4F, where we add a dummy vertex on the branch of leaf 4 in T^2.

Induced matrices. After the labeling step, we transform a pair of unlabeled merge trees T^1 and T^2 to a pair of labeled merge trees $\mathcal{T}^1$ and $\mathcal{T}^2$. We can transfer $\mathcal{T}^1$ and $\mathcal{T}^2$ to *induced matrices* M^1 and M^2 according to the label assignment, as shown in Fig. 4. The pivot label set then gives rise to the rows and columns of an induced matrix. These induced matrices are shown below for the example in Fig. 4B and Fig. 4E, where labels for $\mathcal{T}^2$ are obtained using a dummy leaf strategy:

$$M^1 = \begin{pmatrix} 1.0 & 3.0 & 4.0 & 4.0 \\ & 1.0 & 4.0 & 4.0 \\ & & 2.0 & 3.0 \\ & & & 1.0 \end{pmatrix}, M^2 = \begin{pmatrix} 1.0 & 3.0 & 4.0 & 4.0 \\ & 2.0 & 4.0 & 4.0 \\ & & 1.0 & 1.0 \\ & & & 1.0 \end{pmatrix}.$$

If we apply a dummy vertex strategy to $\mathcal{T}^2$, as shown in Fig. 4F, the induced matrix is

$$M^2 = \begin{pmatrix} 1.0 & 3.0 & 4.0 & 4.0 \\ & 2.0 & 4.0 & 4.0 \\ & & 2.0 & 2.0 \\ & & & 1.0 \end{pmatrix}.$$

4.3 Time-Varying Pivot Tree

So far we have described strategies to find a leaf-leaf correspondence between a pair of unlabeled merge trees. When moving to

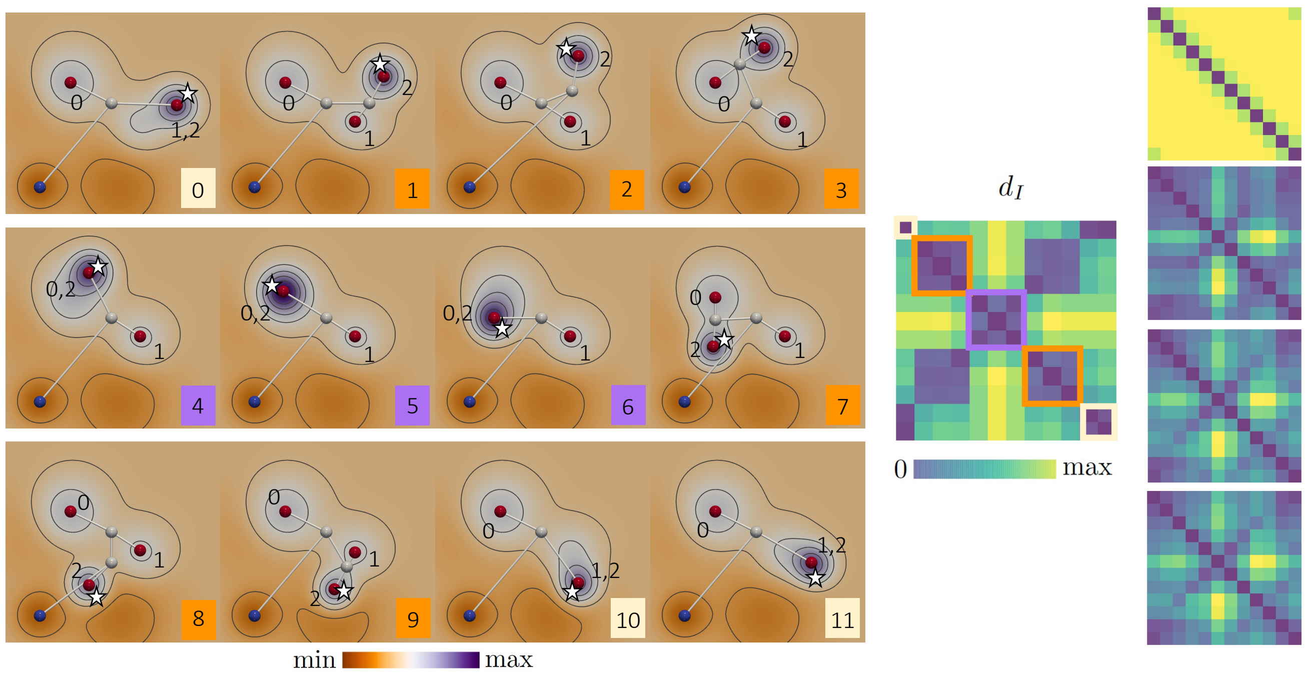

Fig. 5. Moving Gaussian dataset with a hybrid mapping strategy and a time-varying pivot tree. Left: 12 time steps of scalar fields are visualized together with the labeled split trees. Middle and right: distance metrics for d_F, d_B, d_W, d_E, and d_I, respectively, where the zoomed-in version of d_I highlights clusters among the data instances.

a time-varying dataset containing a large number of merge trees, this strategy is, however, no longer sufficient. We need to have a shared label set for all trees, which leads to comparable induced matrices with the same size. Therefore, a pivot tree selection plays an important role, which is discussed below.

Global pivot tree. The first method, introduced in [15], is a direct extension of the strategy from the matching of two trees described in Sect. 4.1 to many trees. It selects a *global pivot tree* as a tree with the largest number of leaves among all input trees. This tree defines the global label set, and we assign labels to all the other trees using the label set from the pivot tree. A major limitation of this approach is the implicit assumption that similarity is a transitive property, which could lead to artifacts for large numbers of trees within a time-varying dataset.

Time-varying pivot tree. To overcome this problem, we introduce a new, *time-varying pivot tree* strategy. The labels are propagated from one tree to the next, capturing temporal changes in a time-varying dataset.

The strategy works as follows: Given a set of merge trees $\{T^0, T^1, \ldots, T^l\}$ that arises from a time-varying dataset, let $T_p := T^i$ (for some i) be an initial pivot tree with the largest n number of leaves, thus defining the label set $[n]$. To assign labels to T^{i-1} that immediately precedes T^i, we use T^i as the pivot; thus, T^{i-1} inherits labels from T^i. To assign labels to T^{i-2}, we use T^{i-1} as its pivot instead. In general, T^j will be the pivot tree for T^{j-1} when $j \leq i$, whereas T^j will be the pivot tree for T^{j+1} when $j \geq i$. In a nutshell, the labels are inherited sequentially as we go through the dataset forward and backward from the initial pivot tree in a time-ordered way. Such a strategy works well with time-varying datasets, as demonstrated in Sect. 5.1 and Sect. 5.2.

Pivot-free strategy. The time-varying pivot tree has its advantages and disadvantages. It is desirable for feature tracking within a time-varying dataset, thus supporting the detection of transitions and clusters in real-world datasets; see Sect. 5.1 and Sect. 5.2. However, if the goal is to detect periodicities within time-varying datasets, we will need to effectively "ignore" geometric dependencies among adjacent time instances and treat these instances independently, which leads to an alternative *pivot-free* strategy. That is, we treat each time instance independently and compute interleaving distances between pairs of instances without requiring a pivot tree or a shared label set across all input trees. In practice, this strategy works reasonably well if we assume the label sets are of roughly the same size; we give an example in Sect. 5.3.

4.4 A Simple Example

To illustrate our analysis pipeline, we give a simple example involving a synthetic time-varying dataset, referred to as the Moving Gaussian dataset. This dataset is generated as a mixture of Gaussian functions centered at seven anchor points in a 2D domain. One of the anchor points (the starred point in Fig. 5 left) performs a circular motion in the domain, while the rest remain stationary. This dataset contains 12 time steps modeled as scalar fields. We compute their corresponding split trees. We include pairwise distance matrices under d_I, d_F, d_B, d_W, and d_E, respectively (see Fig. 5 right). With the Moving Gaussian dataset, we demonstrate how geometric information coupled with topology helps to reveal its clustering structure, using hybrid mapping, dummy leaves, and time-varying pivot tree strategies. For all our experiments, unless otherwise specified, we set $\lambda = 0.5$ for a hybrid mapping strategy; see Sect. 4.5 for a discussion.

Geometric information and a time-varying pivot tree are the two key elements for tracking the moving Gaussian function. As shown in Fig. 5 left (time steps 0 to 11), using a hybrid mapping strategy, our method successfully tracks a moving Gaussian function centered at the starred critical point. This critical point with label 2 performs a counterclockwise circular motion: it

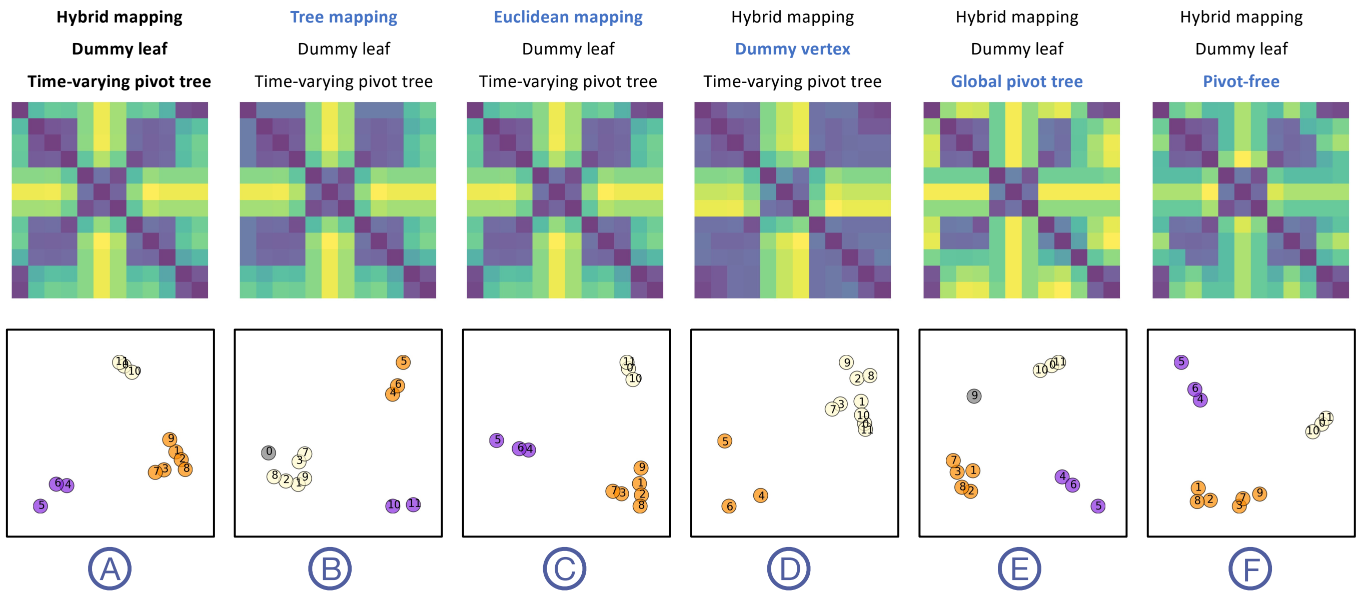

Fig. 6. Moving Gaussian dataset with different labeling strategies. Top: combinations of labeling strategies. Middle: pairwise distances matrices for d_I. Bottom: MDS projections of all time steps.

merges with local maximum with labels 0 and 1 at time steps 4 and 10, respectively, and splits with them at time steps 1 and 7, respectively. A hybrid mapping considers the geometric positions of critical points in the domain. Therefore, for adjacent instances, stationary critical points are more likely to be mapped with each other.

Furthermore, as shown in Fig. 5 right, in comparison with other distance metrics (d_F, d_B, d_W, and d_E), the labeled interleaving distance d_I matrix detects three dominant clusters (bounded by purple, orange, and white squares). These clusters are calculated by DBSCAN clustering algorithm (eps=4 and min_sample=2) from scikit-learn Python library using precomputed distance matrix d_I. These clusters are the results of critical points merging and splitting in the time-varying data. In particular, the interactions of the local maximum 2 with other local maxima 0 and 1 at time steps 1, 4, 7, and 10 directly cause the topological transitions of the underlying split trees.

We now illustrate the time-varying pivot tree strategy. In this experiment, we pick T^1 as the initial pivot tree since T^1 has the largest number of leaves. Then T^0 and T^2 inherit labels from T^1. After that, T^2 becomes the pivot tree for T^3, and T^3 will inherit labels from T^2 and become the pivot tree for T^4, and so forth. If T^p is the initial pivot tree, T^i will be pivot tree for T^{i+1} when $i \geq p$, whereas T^i will be the pivot tree for T^{i-1} when $i \leq p$. The benefit of using the time-varying pivot tree is that such a labeling strategy can propagate both geometric and topological information corresponding to temporal changes. If we used a global pivot tree strategy from [15], T^1 is the only pivot tree, and current labeling results will change, especially when time instances are far from the global pivot tree temporally. For example, labeling T^7 and T^8 using T^1 as a pivot tree will be different with the current labeling result, no matter which mapping strategy we choose.

4.5 Labeling Strategy Selection

We end this section by discussing the labeling strategy selection. We revisit the Moving Gaussian example in Sect. 4.4 and explore how d_I changes across different labeling strategies. We start with the d_I matrix in Fig. 5, which is calculated using hybrid mapping ($\lambda = 0.5$), dummy leaf, and time-varying pivot tree strategies; see also Fig. 6A. We conduct experiments by replacing one strategy at a time.

For the hybrid mapping, we perform a number of experiments by varying the λ parameter for $\lambda \in [0, 0.2, 0.4, 0.6, 0.8, 1.0]$. d_I for Moving Gaussian remains the same for $\lambda \in [0, 0.5]$ (Fig. 6A), and for $\lambda \in [0.6, 1.0]$ (Fig. 6B). By default, we set $\lambda = 0.5$ to strike a balance between geometric and topological information.

Fig. 6 shows d_I distance matrices under different mapping strategies, and their MDS projections colored by DBSCAN clustering result (eps= 0.15 and min_sample=2). We notice that Euclidean mapping in (C) leads to the same d_I as the hybrid mapping with $\lambda = 0.5$ in (A), since both of them detect a moving Gaussian function in the domain. Compared with the dummy vertex strategy (D), the dummy leaf strategy (A) is more sensitive to structural transition than the dummy leaf strategy, where (A) produces three clusters and (D) produces two.

Regarding pivot tree selection, our time-varying pivot tree strategy in (A) works better than global pivot tree strategy in (E) for a time-varying dataset, since the former propagates labels from one tree to the next, which accommodates the inherent temporal nature of time-varying data. Pivot-free strategy in (F) also works fairly well in detecting structural transition and clusters; *cf.* Fig. 6F and Fig. 6A.

5 Experimental Results

In this section, we demonstrate via experiments that geometry-aware merge tree comparisons based on the interleaving distance help detect *transitions*, *clusters*, and *periodicities* of time-varying datasets, as well as to *diagnose* and *highlight* the topological changes between adjacent instances.

In Sect. 5.1 and Sect. 5.2, we use split trees instead of join trees, which encode local maxima. The maxima of the velocity magnitude field in a flow dataset are chosen as features of interest as they distinguish regions of high flow compared to the background. In the flow datasets used in our experiments, such

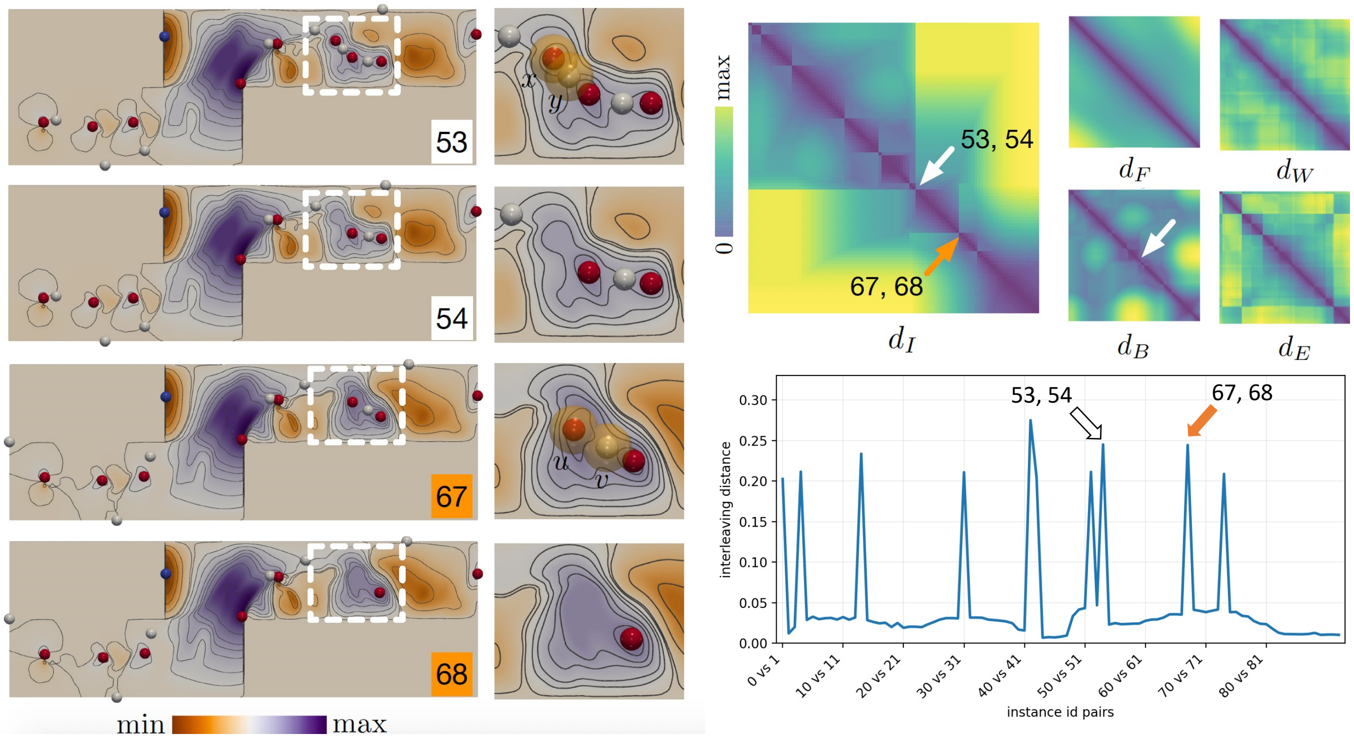

Fig. 7. Geometry-aware merge tree comparisons for a time-varying Corner Flow dataset. Left two columns: selected scalar fields with their zoomed-in views, where local minimum are in blue, saddles are in white, local maximum are in red. Right top: pairwise distance matrices for d_F, d_B, d_W, d_E, and d_I. Right bottom: d_I transition curve. Arrows in d_I and its transition curve highlight selected structural transitions.

maxima also approximate very well the regions with high vorticity. In Sect. 5.3, we conduct experiments based on both join trees and split trees to capture differences in the behavior of local extrema.

The labeling strategies for each dataset can be found in Table 1. The most common approach we employ combines hybrid mapping, dummy vertex, and time-varying pivot tree strategies. We use a hybrid mapping with $\lambda = 0.5$ to strike a balance between geometric and topological information. We choose a dummy leaf strategy with a small dataset Moving Gaussian for a more obvious clustering pattern. However, for datasets that are much larger than the Moving Gaussian, a dummy vertex strategy is better at detecting structural transitions and clusters since it is less sensitive and does not focus on spurious features. A time-varying pivot tree generally outperforms a global pivot tree for time-varying data in detecting structural transitions and clusters. For periodicity detection, a pivot-free strategy appears to be better than a time-varying pivot tree, as mentioned in Sect. 4. Therefore, we use a pivot-free strategy for the Vortex Street dataset; see a detailed comparison between pivot-free strategy and time-varying pivot tree in Sect. 5.3.

Dataset	Mapping	Dummy label	Pivot tree
Moving Gaussian	hybrid	dummy leaf	time-varying
Corner Flow	hybrid	dummy vertex	time-varying
Heated Flow	hybrid	dummy vertex	time-varying
Red Sea	hybrid	dummy vertex	time-varying
Wing	hybrid	dummy vertex	time-varying
Vortex Street	hybrid	dummy vertex	pivot-free

TABLE 1
Labeling strategies for each dataset.

5.1 Detect and Diagnose Structural Transitions

We demonstrate our method for detecting structural transitions using two flow datasets: the Corner Flow and Heated Flow datasets.

Corner Flow dataset. We first demonstrate our method using the 2D Cylinder Flow Around Corners dataset (https://cgl.ethz.ch/research/visualization/data.php), which we refer to as the Corner Flow dataset. This dataset arises from the simulation of a viscous 2D flow around two cylinders [52], [53]. The channel into which the fluid is injected is bounded by solid walls. A vortex street is initially formed at the lower left corner, which then evolves around the two corners of the bounding walls. We generate a set of split trees from the vertical component of the velocity vector fields based on 94 time instances – they correspond to steps 801-894 from the original 1500 time steps. These instances describe the formation of a one-sided vortex street on the upper right corner; see Fig. 7 left, which visualizes the scalar fields associated with time steps 53, 54, 67, and 68.

To separate signals from noise, we apply persistence simplification [54] to the split trees of all instances with a persistence threshold of 0.2. In particular, to guide the selection of the persistence threshold, we employ a set of *persistence graphs*, each representing the number of critical point pairs as a function of persistence [55]. The shape of the persistence graph, in particular, a plateau, indicates a stable range of scales to separate noise from signals in the persistence graph [56], [57]. We demonstrate such a simplification process in Fig. 10, where time instance 54 is shown before and after persistence simplification; the persistence threshold is chosen approximately due to the variabilities across time instances. We also provide d_I distance matrices across multiple persistence thresholds (Fig. 10 bottom). These matrices exhibit

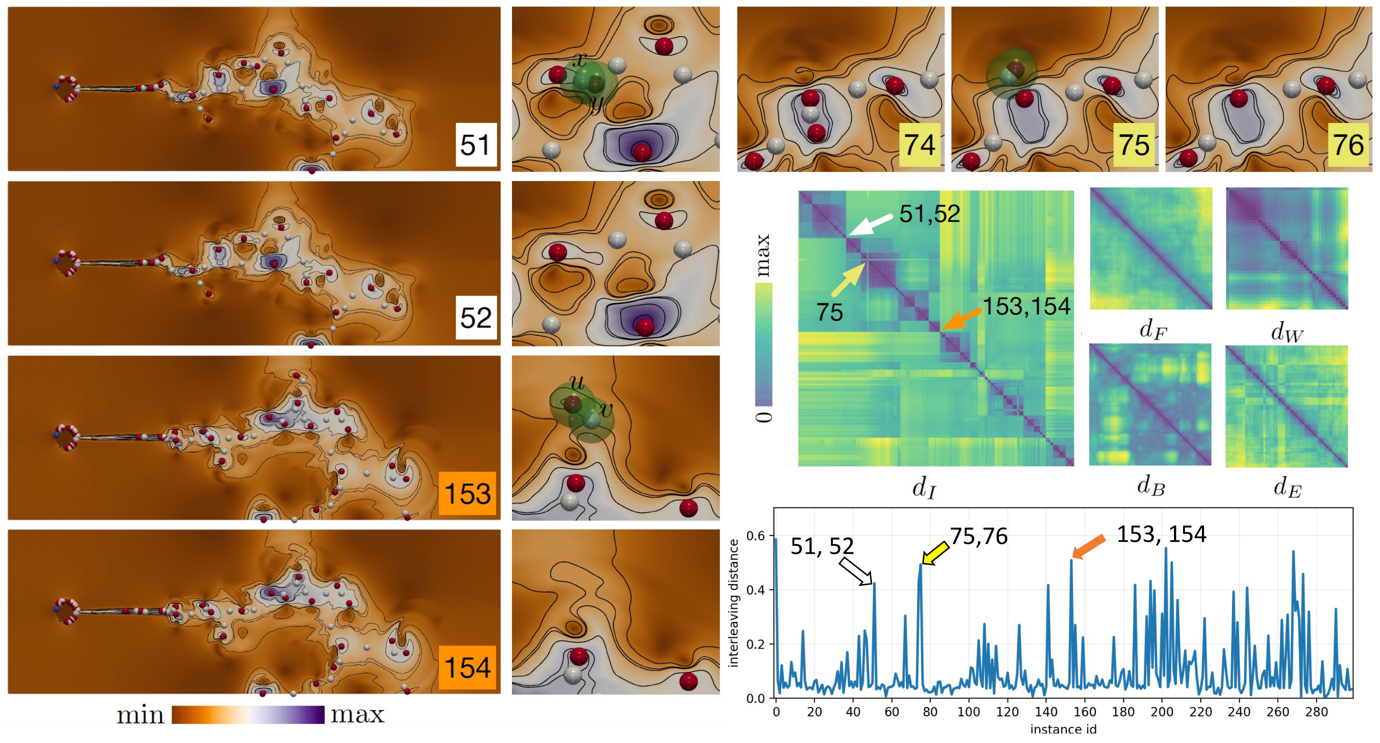

Fig. 8. Heated Flow dataset: geometry-aware merge tree comparisons. Left two columns: selected scalar fields with their zoomed-in views, where local minimum are in blue, saddles are in white, local maximum are in red. Right top: zoomed-in views for the selected scalar fields responsible for light cross lines at instance 75 on d_I. Right middle: pairwise distance matrices for d_F, d_B, d_W, d_E, and d_I, respectively. Right bottom: d_I transition curve. Arrows in d_I and its transition curve highlight detected structural transitions.

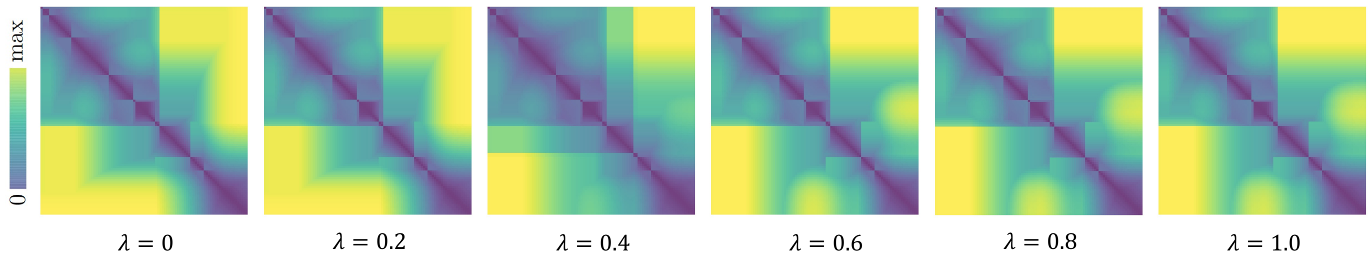

Fig. 9. Pairwise d_I distance matrices for the Corner Flow dataset with varying λ.

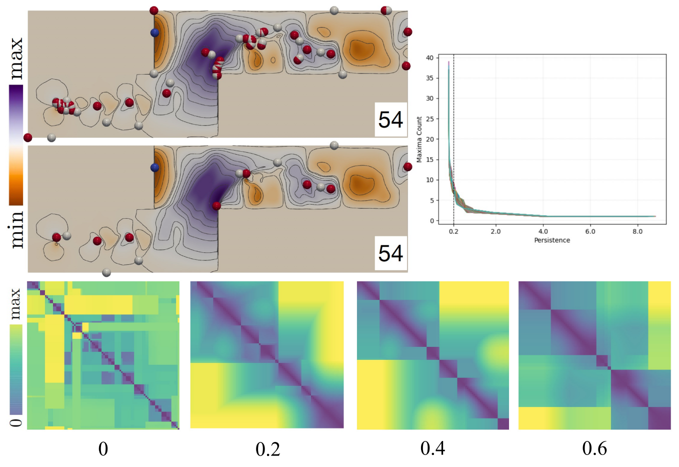

Fig. 10. Corner Flow dataset. Top left: time instance 54 before (top) and after (bottom) persistence simplification. Top right: a set of persistence graphs used to guide the simplification process. Bottom: pairwise interleaving distance matrices d_I with persistence thresholds at 0 (*i.e.*, no simplification), 0.2, 0.4, and 0.6.

different block structures since d_I are sensitive to the appearances and disappearances of critical points due to persistence simplification. However, we can still observe common trends among these matrices: (1) a higher persistence threshold leads to fewer critical points, which leads to fewer block structures in d_I matrices; (2) the overall block structures associated with persistence thresholds 0.2, 0.4, and 0.6 are quite similar, indicating a certain amount of robustness *w.r.t.* persistence simplification.

Our framework analyzes structural transitions via the pairwise distance matrices. For this experiment, we use hybrid mapping, dummy vertex, and time-varying pivot tree strategies. Fig. 9 illustrates our parameter selection of $\lambda \in [0, 0.2, 0.4, 0.6, 0.8, 1.0]$. Specifically, d_I does not change much within a certain range: d_I looks similar for $\lambda = 0, 0.2$, and for $\lambda = 0.6, 0.8, 1.0$.

As shown in Fig. 7 middle, our framework using the interleaving distance d_I captures two obvious structural transitions among adjacent instances, $53 \to 54$ and $67 \to 68$. There appear to be clear block structures within the d_I matrix, where the above transitions are highlighted by arrows at the corners of these blocks. We also use a curve that captures interleaving distance d_I between adjacent time steps, referred to as the d_I transition curve; see Fig. 7 right bottom. This curve records the interleaving distance between instances i and $i+1$ ($0 \leq i < 94$), and highlights positions where obvious structural transitions occur.

Under the diagnostic setting, we locate critical points in the domain that are responsible for the d_I distance between adjacent instances. A close inspection of this time-varying dataset then reveals that from step 53 to 54, a pair of critical points x and y (enclosed by orange spheres) disappears (Fig. 7 top left). Similarly, from step 67 to 68, another pair of critical points u and v (enclosed by orange spheres) disappears. Therefore, d_I highlights structural transitions in the time-varying data, whereas in comparison, only the bottleneck distance d_B is able to capture the same $53 \to 54$ transition in its matrix representation (Fig. 7 white arrow in d_B).

Heated Flow dataset. We give another example using a 2D Heated Cylinder with a Boussinesq Approximation dataset (https://cgl.ethz.ch/research/visualization/data.php), denoted as the HeatedFlow dataset. This dataset comes from the simulation of a 2D flow generated by a heated cylinder using the Boussinesq approximation [53], [58]. It shows a time-varying turbulent plume containing numerous small vortices.

We convert each time instance of the flow into a scalar field using the magnitude of the velocity vector. We then generate a set of split trees from these scalar fields based on 300 time steps, corresponding to steps 1000-1299 from the original 2000 time steps. This dataset captures the evolution of small vortices over time. We use hybrid mapping, dummy vertex, and time-varying pivot tree strategies, and apply simplification with a persistence threshold of 0.06.

The results are shown in Fig. 8. We observe two visible structural transitions based on d_I between steps $51 \to 52$ and $153 \to 154$ (indicated by white and orange arrows in d_I and its transition curve, respectively). Under the diagnostic setting, the structural transition $51 \to 52$ is caused by the disappearance of a pair of critical points x and y at step 51 (highlighted by green bubbles). The transition $153 \to 154$ is a result of the disappearance of the pair u and v at step 153.

Two additional structural transitions are detected at $74 \to 75$ and $75 \to 76$, as indicated by the yellow arrow in the d_I matrix of Fig. 8. Since instance 75 appears to be an outlier under d_I among its neighboring instances, the d_I transition curve shows high values at $i = 75$ and 76, which indicates interleaving distances between instances 74 and 75 and between instances 75 and 76. However, a closer inspection indicates that such a transition is, in fact, an artifact as a result of persistence simplification, which leads to structural changes of the simplified split trees. We will discuss such artifacts further in Sect. 6.

5.2 Detect Clusters

Our method using the interleaving distance helps to cluster time instances based on their structural differences. We demonstrate the utility of the method using 2D simulations of the Red Sea and a dataset generated from the Gerris flow solver.

Red Sea dataset. The RedSea dataset originates from the IEEE Scientific Visualization Contest 2020 (https://kaust-vislab.github.io/SciVis2020/), and is generated using a high-resolution MITgcm (Massachusetts Institute of Technology general circulation model), together with remote sensing satellite observations. It is used to study the circulation dynamics and eddy activities of the Red Sea (see [59], [60], [61]). For the experiment, we use the velocity magnitude fields of a particular dataset (named *001.tgz*) with 60 time steps. We generate split trees from the 2D slices perpendicular to the z-axis ($z = 1$). For this experiment, we use hybrid mapping, dummy vertex, and time-varying pivot tree strategies.

We run the DBSCAN clustering algorithm from scikit-learn Python library using precomputed distance matrix d_I with eps=0.2 and min_sample=2, and find four clusters of data instances in Fig. 11. d_I shows that the scalar fields share similar structures from instances 0-6 (in orange square), 9-27 (in white square), 28-54 (in red square), and 55-59 (in magenta square). Instances 7 and 8 are identified as outliers under this parameter setting. We show the MDS projection of all instances based on the d_I metric and colored by the DBSCAN clustering result, together with selected scalar fields from each cluster in Fig. 11.

Taking a closer look at the corresponding split trees in Fig. 12, each cluster of data instances shares a similar tree structure, especially for some instances in the red cluster (*e.g.*, see instances 40, 45, and 50). In this experiment, d_B and d_W also show some clustering patterns; however, in comparison, d_I offers a clearer and more informative clustering pattern.

Wing dataset. The Wing dataset is generated using the software *Gerris flow solver* (http://gfs.sourceforge.net/). We use its demo flow simulation example involving the “starting vortex”, which is a vortex that forms in the fluid near the trailing edge of an aerofoil (wing) as it is accelerated from rest in a fluid. For our simulation, we set the angle of the aerofoil with respect to the fluid as 20° and generate 40 time steps. The scalar field of interest is the velocity magnitude.

For this experiment, we use hybrid mapping, dummy vertex, and time-varying pivot tree strategies. As shown in Fig. 13, we compute various distance matrices based on the split trees computed for the velocity magnitude field. All the trees are simplified at a simplification threshold of 0.13.

We demonstrate DBSCAN clustering result using d_I in Fig. 13 (eps=0.8 and min_sample=2). The d_I matrix detects five clusters of data instances: 1-16 (in blue square), 17-26 (in orange square), 27-28 and 33-35 (in green square), 29-32 (in purple square), and 36-40 (in red square). Instance 0 is marked as an outlier. The MDS projection of all data instances using d_I further highlights the clustering structures among the scalar fields at different time steps. Whereas d_B, d_W, and d_E also capture some clustering structures in this time-varying dataset, d_I gives rise to clusters that appear to be more separable and visually differentiable. Moreover, based on inspection of the velocity magnitude fields shown in Fig. 13 left, it is apparent that the five clusters separate the time steps into clusters with similar vortex structures. For example, the major change between the orange and green clusters is the appearance of a strong vortex on the top of the wing in the green cluster. Similarly, notice the two strong vortices in the time steps corresponding to the purple cluster. As one of those vortices exits the domain in the subsequent time steps, the instances are grouped into a different cluster marked as red.

5.3 Detect Periodicities

Finally, we demonstrate our framework in detecting periodicities using the classic 2D von Kárman vortex street dataset, which we refer to as the VortexStreet dataset. We consider the region of vortex shedding behind the cylinder, and use the velocity magnitude field for comparison as used earlier by Sridharamurthy *et al.* [16].

We use hybrid mapping, dummy vertices, and pivot-free strategies. As shown in Fig. 14 left, using either the join or split tree, d_B,

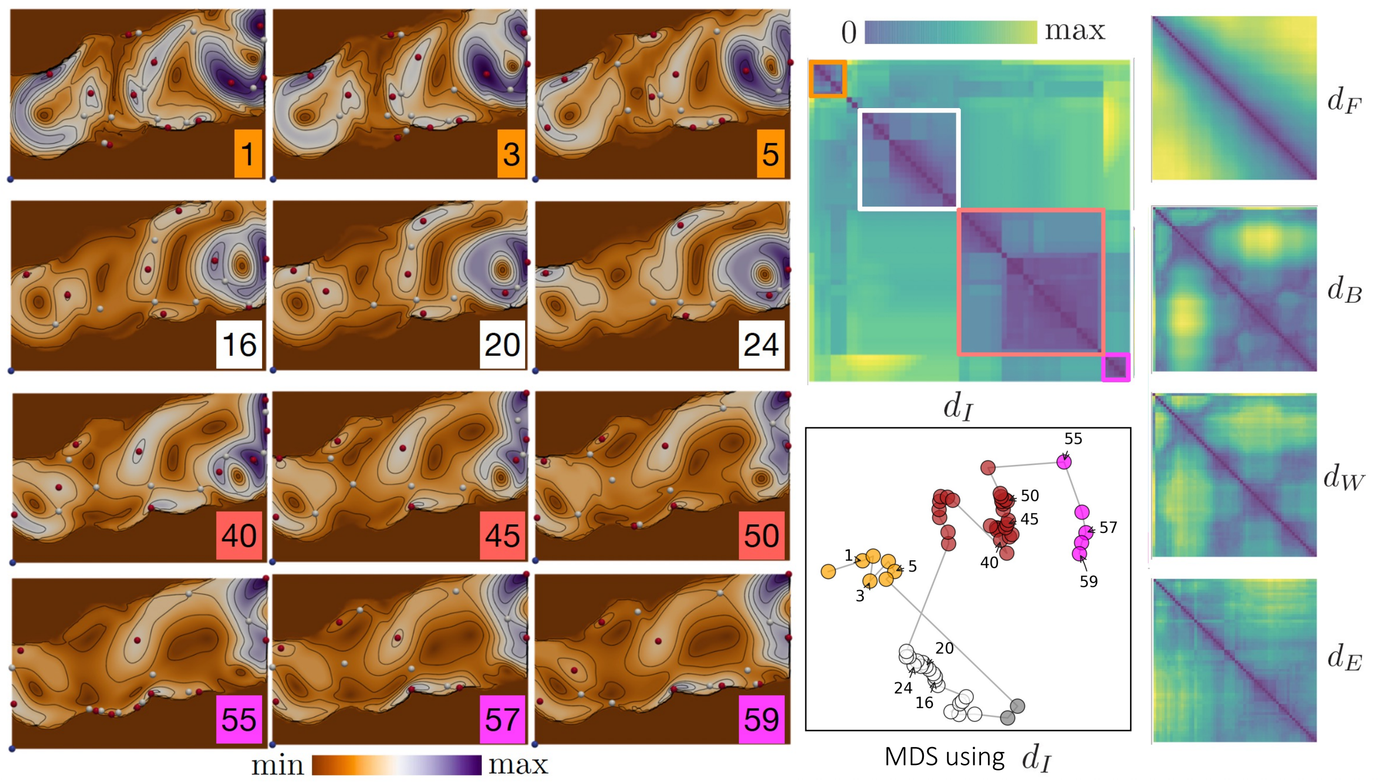

Fig. 11. Red Sea dataset: detect clusters. Left three columns: selected scalar fields drawn from each of the four clusters. Fourth column and second row: MDS projection of all time steps using the d_I metric. Rest of the images: pairwise distance matrices for d_F, d_B, d_W, d_E, and d_I, respectively. Colored boxes in d_I highlight the four detected clusters.

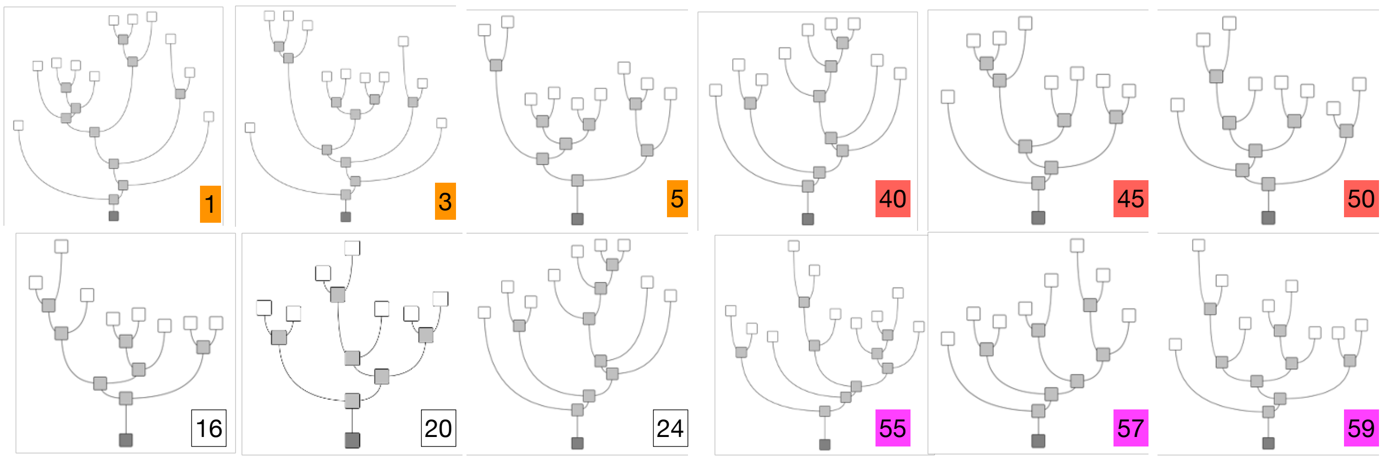

Fig. 12. Red Sea dataset: abstract split trees for selected time steps, *cf.* Fig. 11.

d_W, and d_E all show a periodicity of length 37. The interleaving distance, d_I, captures the same length of periodicity using the join tree. However, using the split tree, we detect a periodicity of length 75 using d_I. Such a longer periodicity coincides with the periodicity detected using d_F. This periodicity can be justified where both d_I and d_F consider more geometric information in the domain, in comparison with other metrics. We also include curves that record distances based on split trees (using d_I, d_F, d_B, d_W, and d_E metrics) between instance 0 and all other instances to show such periodicity patterns; see Fig. 14 middle.

Furthermore, as shown in Fig. 15, the positions of local maxima (red points) change more drastically every 37 time steps, in comparison with the positions of local minima (blue points). This finer difference is captured by the split tree version of d_I; where neither d_B, d_W, nor d_E capture this difference in the behavior of the local extrema.

In this experiment, as mentioned in Sect. 4, a pivot-free mapping strategy works better than a time-varying pivot tree strategy in detecting periodicities. We compare the above results with those obtained using a time-varying pivot tree. Fig. 15 shows labeled local maxima from the scalar fields associated with time steps 1, 37, and 75. Scalar fields at time steps 1 and 75 have similar merge trees *w.r.t.* the locations of critical points. Therefore, they have low distance values under d_F and d_I, d_B, d_W, and d_E in our previous experiment. However, these critical points obtain different labels using a time-varying pivot tree. Since a time-varying pivot tree helps labels propagate from one tree to the next, labeling under this setting indicates that vortices in the Vortex Street dataset

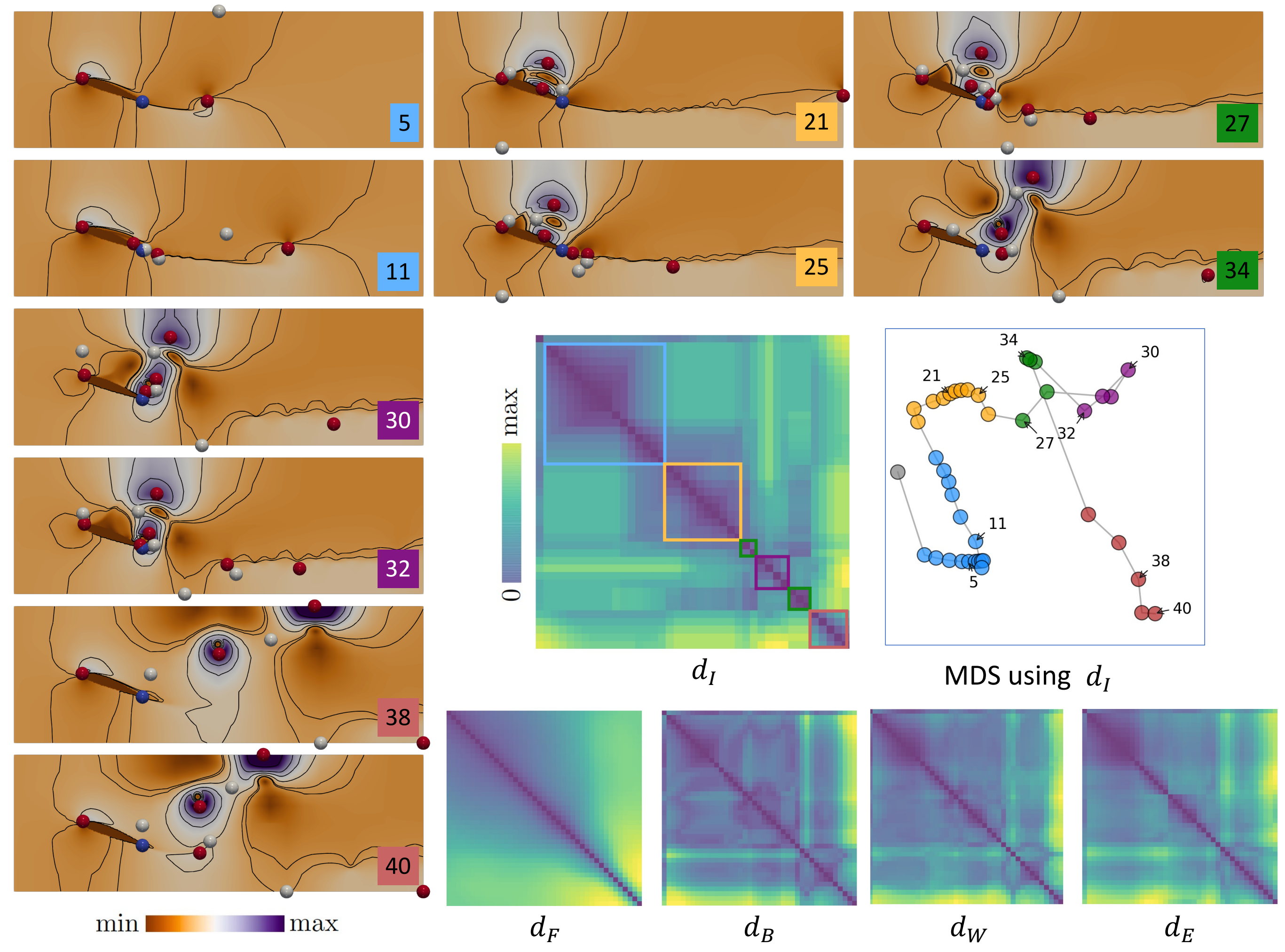

Fig. 13. Wing dataset: detect clusters. Left column and top two rows: selected scalar fields drawn from each of the five clusters. Right column and third row: MDS projection of all the time steps using the d_I metric. Rest of the images: pairwise distance matrices for d_F, d_B, d_W, d_E, and d_I, respectively. Colored boxes in d_I highlight the five detected clusters.

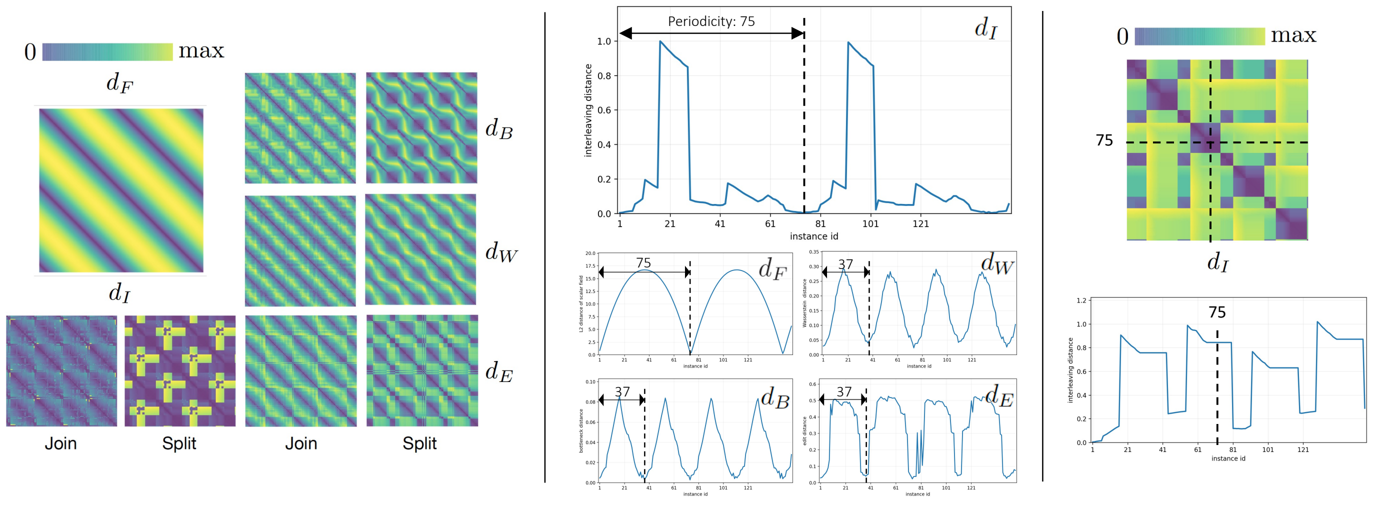

Fig. 14. The VortexStreet dataset. Left: distance matrices for d_F and d_I, d_B, d_W, and d_E, using both join and split trees, respectively. Middle: Distance to instance 0 curves using split tree and under d_I, d_F, d_B, d_W, and d_E metrics. Notice that d_F and d_I have a periodicity of length 75, and the others show a periodicity of length 37. Right: d_I distance matrix and distance to instance 0 curve using split tree with a time-varying pivot tree.

move with almost constant speed to the right; see critical points in white circles in Fig. 15. Therefore, the time-varying pivot tree strategy is better at capturing or inheriting temporal changes, but has poorer performance in detecting periodicities. Fig. 14 right

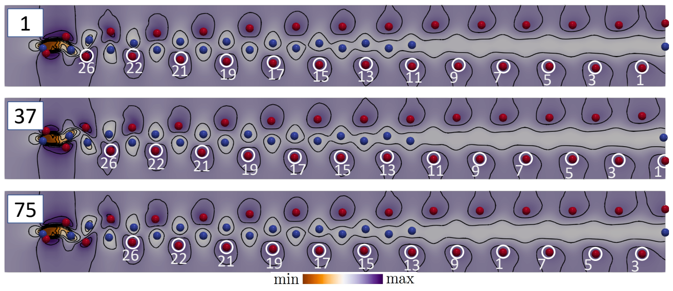

Fig. 15. Selected scalar fields together with some labeling results for the Vortex Street dataset under a hybrid mapping strategy and a time-varying pivot tree.

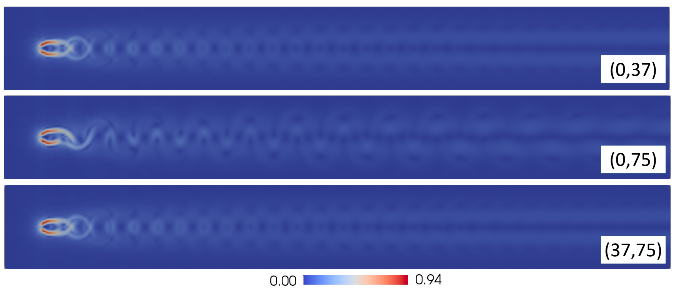

Fig. 17. The gradient-based comparative measure introduced by Nagaraj *et al.* [50] for the Vortex Street dataset. From top to bottom: $||J||$ between instances 0 and 37, 0 and 75, 37 and 75, respectively.

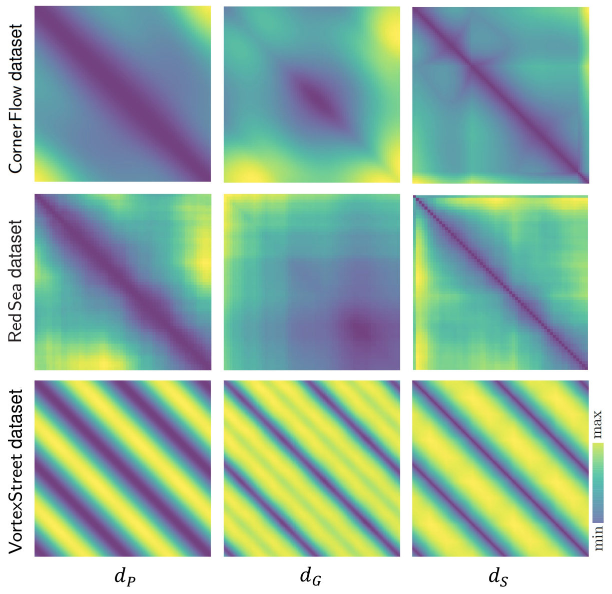

Fig. 16. From left to right: pairwise distance matrices using d_P, d_G, and d_S, respectively for the Corner Flow (1st row), Red Sea (2nd row), and Vortex Street (3rd row) datasets.

shows the results (d_I) under a time-varying pivot tree strategy, where the periodic pattern is less obvious.

5.4 Other Scalar Fields Comparative Measures

As reviewed in Sect. 3.3, there are a number of comparative measures for scalar fields in addition to the Euclidean distance d_F (as shown in Fig. 7, Fig. 8, Fig. 11, and Fig. 13, and Fig. 14, respectively). The pairwise distance matrices for d_P, d_G, and d_S are shown in Fig. 16 for three of our datasets. Compared with d_I in Fig. 7 and Fig. 11, these distance matrices do not contain clear block structures to highlight structural transitions and clusters for the Corner Flow and Red Sea datasets.

For the Vortex Street dataset, d_P, d_G, and d_S capture a periodicity of length 75, since they all consider geometric information in the domain as d_F. However, interestingly, d_G — which encodes average gradient behavior across the domain — also captures a (weak) periodicity pattern of length 37 that is aligned with the one captured by merge trees. We thus investigate this periodicity further. As shown in Fig. 17, we visualize the norm of partial derivatives $||J||$ across the domain (*i.e.*, $||J(x^i)||$ for all $x^i \in \mathbb{X}$) between time instances 0 and 37, 0 and 75, 37 and 75, respectively. Instances 0 and 75 are shown in Fig. 17 (middle) to be very similar in terms of gradient behaviors, in particular, with values closer to zero in regions that are at a distance from the cylinder. These two instances are also shown to be similar to instance 37 in Fig. 17 (top and bottom). We observe that instances 0 and 75 are slightly closer ($d_G = 0.02886$) than 0 and 37 ($d_G = 0.02954$), explaining the weakly observed periodicity.

6 Conclusion and Discussion

In this paper, we introduce a systematic way to integrate geometric information for comparing merge trees. Given a pair of merge trees that arise from scalar fields, our main idea is to decouple the computation of a distance measure into two steps: a labeling step that generates a correspondence between critical points of the merge trees, and a distance computation step that computes the labeled interleaving distance between a pair of labeled merge trees by encoding them as matrices. To encode geometric information, we introduce a hybrid strategy during the labeling step that considers the intrinsic tree distances between critical points and/or the Euclidean distances between their locations in the data domain. We demonstrate that our approach can be used to detect clusters, structural transitions, and periodicity in a way that is either comparable or complementary to existing approaches. There are many directions for future research.

Improved efficiency and robustness. Naively computing the labeled interleaving distance d_I requires access to all entries in the induced matrix, which takes $O(n^2)$ time (n being the number of labels). Developing a more scalable computation would be interesting, possibly taking inspirations from [62].

In addition, the labeled interleaving distance coupled with various labeling strategies has strengths and weaknesses. The strategy is shown to detect periodicity that is more sensitive to the underlying geometry, and it enjoys a certain amount of robustness due to its stability properties. However, at the same time, it is not as robust as some other metrics when the underlying data contain a large amount of noise and a large number of features; improving its robustness is left for future work.

Integration of domain knowledge. Our two-step comparative process opens doors to the integration of domain knowledge. The labeling process can be easily extended to not only integrate geometric information from the data domain but also to encode information from its underlying applications. In particular, domain knowledge will be useful during the initial label assignment described in Sect. 4.1.

Other geometry-based labeling strategies. The hybrid labeling strategy based on the tree distance and/or the Euclidean distance between critical points is just one example among many possible geometry-aware labeling strategies. For example, we could adapt a strategy called *Morse mapping* that has been used for tracking critical points [18]. Given a pair of merge trees T^1 and T^2, the Morse mapping strategy facilitates the gradient flow derived from the scalar fields to define a forward $T^1 \to T^2$ and a backward $T^2 \to T^1$ mapping. The forward and backward assignment builds on the partition of the domain provided by the Morse complex [63], which represents the gradient behavior of the scalar field of the data domain. See Fig. 18 for an example of a Morse complex of a 2D scalar field.

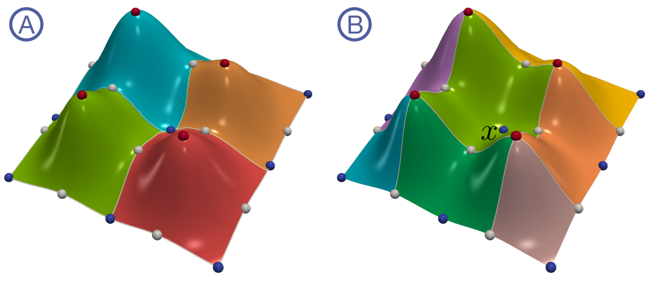

Fig. 18. Given the 2D function f, (a) shows the stable manifolds forming the Morse complex of f, and (b) shows the unstable manifolds forming the Morse complex of $-f$; the green cell surrounding the critical point p is an unstable manifold of p.

Let ∇f denote the gradient of a Morse function $f : \mathbb{X} \to \mathbb{R}$. An *integral line* at a regular point is a maximal path whose tangent vectors agree with the gradient [63]. The function increases along an integral line, which begins and ends at critical points. The *stable manifold* (or *unstable manifold*) surrounding a critical point x includes x itself and all regular points whose integral lines end (or originate) at x [10, Chap. VI, page 131]. The stable and unstable manifolds of x are denoted as $S(x)$ and $U(x)$, respectively. To define the Morse mapping strategy, we use the unstable manifolds surrounding the local minima of the scalar field (Fig. 18b), which correspond to leaves in the merge tree.

Suppose we are given a pair of merge trees T^1 and T^2 that arise from a pair of scalar fields. Let $x \in V(T^1)$ and $x' \in V(T^2)$ denote a pair of leaves (local minima of the underlying scalar fields). Let $U(x)$, $U(x')$ denote their respective unstable manifolds. We say x is *forward mapped* to x' if $x \in U(x')$ and x' is *backward mapped* to x if $x' \in U(x)$, denoted as $x \to x'$ and $x \leftarrow x'$, respectively. In other words, we check to see which unstable manifold a leaf belongs to would determine the label assignment. As illustrated in Fig. 19E-F, given a pair of merge trees T^1 and T^2 (Fig. 19A-B) that arise from scalar fields (Fig. 19C-D), the local minimum x of T^1 is forward mapped to z' since $x \in U(z')$, whereas z' is backward mapped to x since $z' \in U(x)$.

This strategy leads to three categories of matched leaf pairs: *double connected pairs*, which result in a joint label; and *forward* and *backward connected pairs*, which generate a novel label and introduce a dummy node in one of the trees. For example, x and z' form a double connected pair in Fig. 19.

Finally, Fig. 19 further illustrates that different mapping strategies between two merge trees result in different label assignments. In this example, merge trees T^1 and T^2 arise from two synthetic 2D scalar fields generated as mixtures of Gaussians. T^1 (Fig. 19A) contains three leaves that correspond to local minima x, y, z in the domain (Fig. 19C). T^2 (Fig. 19B) contains three leaves that correspond to local minima x', y', z' (Fig. 19C). Using the tree mapping strategy, we obtain a bijective mapping $x \leftrightarrow x'$, $y \leftrightarrow y'$, and $z \leftrightarrow z'$ (*cf.*, Fig. 19A-B). Using the Euclidean mapping strategy, we obtain a different bijective mapping due to proximity, $x \leftrightarrow y'$, $y \leftrightarrow z'$, and $z \leftrightarrow x'$ (*cf.*, Fig. 19C-D). Finally, using the Morse mapping, we obtain sets of forward ($x \to z'$, $y \to y'$, and $z \to x'$, Fig. 19F) and backward ($x \leftarrow z'$, $y \leftarrow y'$, and $z \leftarrow x'$, Fig. 19E) mappings, forming double connected pairs. Understanding such differences is important in choosing the appropriate strategies for particular datasets, which remains an open question.

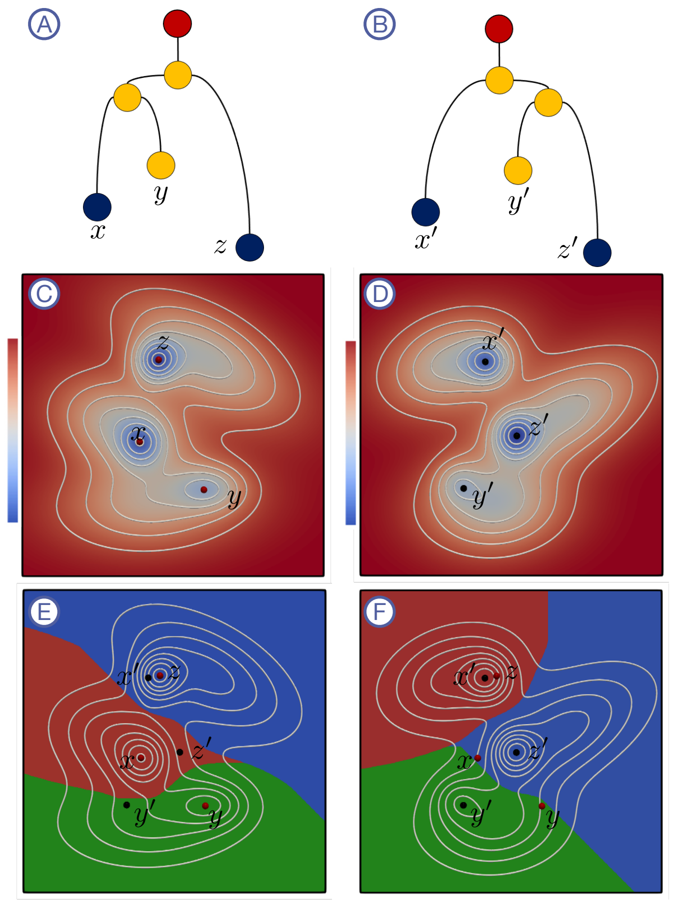

Fig. 19. Comparing three mapping strategies with two synthetic datasets generated as mixtures of Gaussians. Label assignments obtained via tree mapping, Euclidean mapping, and Morse mapping strategies give rise to different labels.

Acknowledgments

This project was partially supported by DOE DE-SC0021015. IH acknowledges support from the following funding sources: Swedish e-Science Research Center (SeRC), Excellence Center at Linköping – Lund in Information Technology (ELLIIT), the Swedish Research Council (VR) under grant 2019-05487, and Wallenberg AI, Autonomous Systems and Software Program (WASP).

References

[1] L. Yan, T. B. Masood, R. Sridharamurthy, F. Rasheed, V. Natarajan, I. Hotz, and B. Wang, "Scalar field comparison with topological descriptors: Properties and applications for scientific visualization," *Computer Graphics Forum, accepted*, 2021.

[2] A. A. Valsangkar, J. M. Monteiro, V. Narayanan, I. Hotz, and V. Natarajan, "An exploratory framework for cyclone identification and tracking," *IEEE Transaction on Visualization and Computer Graphics*, vol. 25, no. 3, pp. 1460–1473, 2018.

[3] P.-T. Bremer, G. Weber, J. Tierny, V. Pascucci, M. Day, and J. Bell, "Interactive exploration and analysis of large scale simulations using topology-based data segmentation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 9, pp. 1307–1324, 2011.

[4] D. M. Thomas and V. Natarajan, "Symmetry in scalar field topology," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2035–2044, 2011.

[5] T. B. Masood, D. M. Thomas, and V. Natarajan, "Scalar field visualization via extraction of symmetric structures," *The Visual Computer*, vol. 29, no. 6-8, pp. 761–771, 2013.

[6] E. Merelli, M. Rucco, P. Sloot, and L. Tesei, "Topological characterization of complex systems: Using persistent entropy," *Entropy*, vol. 17, pp. 6872–6892, 2015.

[7] H. Edelsbrunner, Z. Virk, and H. Wagner, "Topological data analysis in information space," *Proceedings of the 35th International Symposium on Computational Geometry*, vol. 129, pp. 31:1–31:14, 2019.

[8] A. Brown, O. Bobrowski, E. Munch, and B. Wang, "Probabilistic convergence and stability of random mapper graphs," *Journal of Applied and Computational Topology*, vol. 5, pp. 99–140, 2021.

[9] G. Carlsson, A. J. Zomorodian, A. Collins, and L. J. Guibas, "Persistence barcodes for shapes," *Proceedings of the Eurographs/ACM SIGGRAPH Symposium on Geometry Processing*, pp. 124–135, 2004.

[10] H. Edelsbrunner and J. Harer, *Computational Topology: An Introduction*. American Mathematical Society, 2010.

[11] J. Curry, "The fiber of the persistence map for functions on the interval," *Journal of Applied and Computational Topology*, vol. 2, no. 3-4, pp. 301–321, 2018.

[12] M. J. Catanzaro, J. M. Curry, B. T. Fasy, J. Lazovskis, G. Malen, H. Riess, B. Wang, and M. Zabka, "Moduli spaces of Morse functions for persistence," *Journal of Applied and Computational Topology*, vol. 4, pp. 353–385, 2020.

[13] L. Kanari, A. Garin, and K. Hess, "From trees to barcodes and back again: Theoretical and statistical perspectives," *Algorithms*, vol. 13, no. 12, 2020.

[14] E. Gasparovic, E. Munch, S. Oudot, K. Turner, B. Wang, and Y. Wang, "Intrinsic interleaving distance for merge trees," *arXiv:1908.00063*, 2019.

[15] L. Yan, Y. Wang, E. Munch, E. Gasparovic, and B. Wang, "A structural average of labeled merge trees for uncertainty visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 832–842, 2020.

[16] R. Sridharamurthy, T. B. Masood, A. Kamakshidasan, and V. Natarajan, "Edit distance between merge trees," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 1518–1531, 2020.

[17] C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. D. Floriani, G. Schauermann, H. Hagen, and C. Garth, "A survey of topology-based methods in visualization," *Computer Graphics Forum*, 2016.

[18] J. Reininghaus, J. Kasten, T. Weinkauf, and I. Hotz, "Efficient computation of combinatorial feature flow fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 9, pp. 1563–1573, 2012.

[19] J. Reininghaus, N. Kotava, D. Günther, J. Kasten, H. Hagen, and I. Hotz, "A scale space based persistence measure for critical points in 2D scalar fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2045–2052, 2011.

[20] H. Saikia and T. Weinkauf, "Global feature tracking and similarity estimation in time-dependent scalar fields," *Computer Graphics Forum*, vol. 36, no. 3, pp. 1–11, 2017.

[21] M. Soler, M. Plainchault, B. Conche, and J. Tierny, "Lifted Wasserstein matcher for fast and robust topology tracking," in *IEEE 8th Symposium on Large Data Analysis and Visualization (LDAV)*, 2018, pp. 23–33.

[22] D. Laney, P.-T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci, "Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 1, pp. 1053–1060, 2007.

[23] D. Morozov, K. Beketayev, and G. Weber, "Interleaving distance between merge trees," *Proceedings of Topology-Based Methods in Visualization (TopoInVis)*, 2013.

[24] K. Beketayev, D. Yeliussizov, D. Morozov, G. Weber, and B. Hamann, "Measuring the distance between merge trees," *Topological Methods in Data Analysis and Visualization III: Theory, Algorithms, and Applications, Mathematics and Visualization*, pp. 151–166, 2014.

[25] U. Bauer, X. Ge, and Y. Wang, "Measuring distance between Reeb graphs," *Proceedings of the 30th Annual Symposium on Computational Geometry*, pp. 464–474, 2014.

[26] M. Carriére and S. Oudot, "Local equivalence and intrinsic metrics between Reeb graphs," *Proceedings of the 33rd International Symposium on Computational Geometry*, vol. 77, pp. 25:1–25:15, 2017.

[27] U. Bauer, C. Landi, and F. Memoli, "The Reeb graph edit distance is universal," *Foundations of Computational Mathematics*, vol. 21, no. 1441–1464, 2021.

[28] M. Pont, J. Vidal, J. Delon, and J. Tierny, "Wasserstein distances, geodesics and barycenters of merge trees," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 291–301, 2021.

[29] K. Beketayev, G. H. Weber, D. Morozov, A. Abzhanov, and B. Hamann, "Geometry-preserving topological landscapes," in *Workshop at SIGGRAPH Asia (WASA)*, 2012, pp. 155–160.

[30] A.-P. Lohfink, F. Wetzels, J. Lukasczyk, G. H. Weber, and C. Garth, "Fuzzy contour trees: Alignment and joint layout of multiple contour trees," *Computer Graphics Forum*, vol. 39, no. 3, pp. 343–355, 2020.

[31] M. Herick, V. Molchanov, and L. Linsen, "Temporally coherent topological landscapes for time-varying scalar fields," in *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (IVAPP)*, vol. 3, 2020, pp. 54–61.

[32] P.-T. 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*, vol. 16, no. 2, pp. 248–260, 2009.

[33] P.-T. Bremer, G. Weber, J. Tierny, V. Pascucci, M. Day, and J. Bell, "Interactive exploration and analysis of large-scale simulations using topology-based data segmentation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 9, pp. 1307–1324, 2010.

[34] Z. Smith, S. Chowdhury, and F. Memoli, "Hierarchical representations of network data with optimal distortion bounds," *50th Asilomar Conference on Signals, Systems and Comroxiuters*, 2016.

[35] G. Carlsson and F. Mémoli, "Characterization, stability and convergence of hierarchical clustering methods," *Journal of Machine Learning Research*, vol. 11, pp. 1425–1470, 2010.

[36] H. Carr, J. Snoeyink, and U. Axen, "Computing contour trees in all dimensions," *Computational Geometry Theory and Applications*, vol. 24, no. 2, pp. 75–94, 2003.

[37] G. Cardona, A. Mir, F. Rosselló, L. Rotger, and D. Sánchez, "Cophenetic metrics for phylogenetic trees, after Sokal and Rohlf," *BMC Bioinformatics*, vol. 14, no. 1, p. 3, 2013.

[38] F. Chazal, D. Cohen-Steiner, M. Glisse, L. J. Guibas, and S. Y. Oudot, "Proximity of persistence modules and their diagrams," *Proceedings of the 25th Annual Symposium on Computational Geometry*, pp. 237–246, 2009.

[39] F. Chazal, V. de Silva, M. Glisse, and S. Oudot, *The Structure and Stability of Persistence Modules*. Springer International Publishing, 2016.

[40] E. Munch and A. Stefanou, "The ℓ^∞-cophenetic metric for phylogenetic trees as an interleaving distance," in *Research in Data Science*, ser. Association for Women in Mathematics Series. Springer International Publishing, 2019, pp. 109–127.

[41] E. F. Touli and Y. Wang, "FPT-algorithms for computing Gromov-Hausdorff and interleaving distances between trees," *Proceedings of the 27th Annual European Symposium on Algorithms*, pp. 83:1–83:14, 2019.

[42] V. de Silva, E. Munch, and A. Patel, "Categorified Reeb graphs," *Discrete & Computational Geometry*, pp. 1–53, 2016.

[43] J. Curry, "Sheaves, cosheaves and applications," Ph.D. dissertation, University of Pennsylvania, 2014.

[44] E. Munch and B. Wang, "Convergence between categorical representations of Reeb space and mapper," *Proceedings of 32nd International Symposium on Computational Geometry*, vol. 51, pp. 53:1–53:16, 2016.

[45] P. Bubenik, V. de Silva, and J. Scott, "Metrics for generalized persistence modules," *Foundations of Computational Mathematics*, vol. 15, no. 6, pp. 1501–1531, 2014.

[46] V. de Silva, E. Munch, and A. Stefanou, "Theory of interleavings on categories with a flow," *Theory and Applications of Categories*, vol. 33, no. 21, pp. 583–607, 2018.

[47] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, "Stability of persistence diagrams," *Discrete & Computational Geometry*, vol. 37, pp. 103–120, 2007.

[48] K. Zhang, "A constrained edit distance between unordered labeled trees," *Algorithmica*, vol. 15, pp. 205–222, 1996.

[49] P. D. Doncker, "Spatial correlation functions for fields in three dimensional Rayleigh channels," *Journal of Electromagnetic Waves and Applications*, vol. 17, no. 6, pp. 877–878, 2003.

[50] S. Nagaraj, V. Natarajan, and R. S. Nanjundiah, "A gradient-based comparison measure for visual analysis of multifield data," *Computer Graphics Forum*, vol. 30, no. 3, pp. 1101–1110, 2011.

[51] A. Fofonov and L. Linsen, "Projected field similarity for comparative visualization of multi-run multi-field time-varying spatial data," *Computer Graphics Forum*, vol. 38, no. 1, pp. 286–299, 2019.

[52] I. Baeza Rojo and T. Günther, "Vector field topology of time-dependent flows in a steady reference frame," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 280–290, 2020.

[53] S. Popinet, "Free computational fluid dynamics," *ClusterWorld*, vol. 2, no. 6, 2004. [Online]. Available: http://gfs.sf.net/

[54] H. Edelsbrunner, D. Letscher, and A. J. Zomorodian, "Topological persistence and simplification," *Discrete & Computational Geometry*, vol. 28, pp. 511–533, 2002.

[55] S. Gerber, P.-T. Bremer, V. Pascucci, and R. Whitaker, "Visual exploration of high dimensional scalar functions," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, pp. 1271–1280, 2010.

[56] P.-T. Bremer, D. Maljovec, A. Saha, B. Wang, J. Gaffney, B. K. Spears, and V. Pascucci, "ND2AV: N-dimensional data analysis and visualization – analysis for the national ignition campaign," *Computing and Visualization in Science*, vol. 17, no. 1, pp. 1–18, 2015.

[57] T. Athawale, D. Maljovec, L. Yan, C. R. Johnson, V. Pascucci, and B. Wang, "Uncertainty visualization of 2D Morse complex ensembles using statistical summary maps," *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[58] T. Günther, M. Gross, and H. Theisel, "Generic objective vortices for flow visualization," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 141:1–141:11, 2017.

[59] I. Hoteit, X. Luo, M. Bocquet, A. Köhl, and B. Ait-El-Fquih, "Data assimilation in oceanography: Current status and new directions," in *New Frontiers in Operational Oceanography*, E. P. Chassignet, A. Pascual, J. Tintoré, and J. Verron, Eds. GODAE OceanView, 2018.

[60] P. Zhan, G. Krokos, D. Guo, and I. Hoteit, "Three-dimensional signature of the Red Sea eddies and eddy-induced transport," *Geophysical Research Letters*, vol. 46, no. 4, pp. 2167–2177, 2019.

[61] P. Zhan, A. C. Subramanian, F. Yao, and I. Hoteit, "Eddies in the Red Sea: A statistical and dynamical study," *Journal of Geophysical Research*, vol. 119, no. 6, pp. 3909–3925, 2014.

[62] M. Kerber, D. Morozov, and A. Nigmetov, "Geometry helps to compare persistence diagrams," *Journal of Experimental Algorithmics*, vol. 22, no. 1.4, 2017.

[63] H. Edelsbrunner, J. Harer, and A. Zomorodian, "Hierarchical Morse complexes for piecewise linear 2-manifolds," in *Proceedings of the 17th Annual Symposium on Computational Geometry*, Medford, MA, USA, 2001, pp. 70–79.

Lin Yan is a PhD student in the Scientific Computing and Imaging (SCI) Institute, University of Utah. Her research interests include topological data analysis and visualization. Her recent work includes statistical analysis and uncertainty visualization of topological descriptors.

Talha Bin Masood is a Postdoctoral Fellow at Linköping University in Sweden. He received his Ph.D. in Computer Science from the Indian Institute of Science, Bangalore. His research interests include scientific visualization, computational geometry, computational topology, and their applications to various scientific domains.

Farhan Rasheed is PhD student at Linköping University in Sweden. He graduated from Heidelberg University with a degree in Scientific Computing. His research interests includes scientific visualization, topological data analysis, machine learning, and medical image computing.

Ingrid Hotz is currently a Professor in Scientific Visualization at the Linköping University in Sweden. She received her Ph.D. degree from the Computer Science Department at the University of Kaiserslautern, Germany. Her research interests lie in data analysis and scientific visualization, ranging from basic research questions to effective solutions to visualization problems in applications.

Bei Wang is an Assistant Professor at the School of Computing and a faculty member at the SCI Institute, University of Utah. She received her Ph.D. in Computer Science from Duke University. Her research interests include topological data analysis, data visualization, computational topology, computational geometry, machine learning, and data mining.