DOCTORAL THESIS
SORBONNE UNIVERSITÉ

Speciality: **COMPUTER SCIENCE**

Presented by

MATHIEU PONT

to obtain the degree of

Ph.D. of Sorbonne Université

Analysis of Ensembles of Topological Descriptors

To be defended on December 1st, 2023, before the committee:

David COEURJOLLY	Senior Scientist	CNRS	**Reviewer**
Vijay NATARAJAN	Professor	IISc Bangalore	**Reviewer**
Elsa CAZELLES	Research Scientist	CNRS	**Examiner**
Stanley DURRLEMAN	Senior Scientist	INRIA	**Examiner**
Roland KWITT	Full Professor	University of Salzburg	**Examiner**
Gabriel PEYRÉ	Senior Scientist	CNRS	**Examiner**
Katharine TURNER	Senior Lecturer	Australian National University	**Examiner**
Julien TIERNY	Senior Scientist	CNRS	**Advisor**

SORBONNE UNIVERSITÉ
LIP6 – Laboratoire d'Informatique de Paris 6
UMR 7606 Sorbonne Université – CNRS
4 Place Jussieu – 75005 Paris

Publications

First-Author Publications

Journal Papers

Wasserstein Distances, Geodesics and Barycenters of Merge Trees
Mathieu Pont, Jules Vidal, Julie Delon and Julien Tierny
IEEE Transactions on Visualization and Computer Graphics
Proc. of IEEE VIS 2021

Principal Geodesic Analysis of Merge Trees (and Persistence Diagrams)
Mathieu Pont, Jules Vidal and Julien Tierny
IEEE Transactions on Visualization and Computer Graphics, 2022
To be presented at IEEE VIS 2023

Wasserstein Auto-Encoders of Merge Trees (and Persistence Diagrams)
Mathieu Pont and Julien Tierny
Submitted, 2023.

Other Publications

Journal Papers

Merge Tree Geodesics and Barycenters with Path Mappings
Florian Wetzels, Mathieu Pont, Julien Tierny and Christoph Garth
IEEE Transactions on Visualization and Computer Graphics
Proc. of IEEE VIS 2023

Tutorials

Topological Analysis of Ensemble Scalar Data with TTK, A Sequel
Christoph Garth, Charles Gueunet, Pierre Guillou, Federico Iuricich, Joshua Levine, Jonas Lukasczyk, Mathieu Pont, Julien Tierny, Jules Vidal, Bei Wang, Florian Wetzels
IEEE VIS Tutorials 2022

A Hands-on TTK Tutorial for Absolute Beginners
Christoph Garth, Robin Maack, Mathieu Pont and Julien Tierny
IEEE VIS Tutorials 2023

Contents

Notations

$\mathbb{X}$	Topological space
$\mathbb{M}$	Manifold
$\mathbb{R}^d$	Euclidean space of dimension d
$\sigma,\ \tau$	Simplex and face of a simplex
$Lk(v),\ Lk^-(v),\ Lk^+(v)$	Link, lower link and upper link of a vertex v
$\mathcal{K}$	Simplicial complex
$\mathcal{T}$	Triangulation
$\mathcal{M}$	Piecewise linear manifold
$f : \mathcal{M} \to \mathbb{R}$	Piecewise linear scalar field
$\mathcal{Z}_p(\mathcal{K})$	Group of p-cycles of a simplicial complex $\mathcal{K}$
$\mathcal{B}_p(\mathcal{K})$	Group of p-boundaries of a simplicial complex $\mathcal{K}$
$\mathcal{H}_p(\mathcal{K})$	p^{th} homology group of a simplicial complex $\mathcal{K}$
$\beta_p(\mathcal{K})$	p^{th} Betti number of a simplicial complex $\mathcal{K}$
w	Isovalue
$f^{-1}(w)$	Level set of f at w
$f^{-1}_{-\infty}(w),\ f^{-1}_{+\infty}(w)$	Sub-level and super-level set of f at w
$St(v)$	Star of a vertex v
$St^-(v)$	Lower star of a vertex v
$\mathcal{K}_i$	i^{th} step of a filtration on $\mathcal{K}$
$\mathcal{H}_p^{i,j}$	p^{th} persistent homology group given $\mathcal{H}_p(\mathcal{K}_i)$ and $\mathcal{H}_p(\mathcal{K}_j)$
$\mathcal{D}(f)$	Persistence diagram of f
$d_q(p_i, p_j)$	Ground q-distance between persistence pairs p_i and p_j
$W_q^{\mathcal{D}}\big(\mathcal{D}(f_i), \mathcal{D}(f_j)\big)$	L^q-Wasserstein distance between the diagrams $\mathcal{D}(f_i)$ and $\mathcal{D}(f_j)$
ϕ	Bijective assignment between the *augmented* diagrams $\mathcal{D}(f_i)$ and $\mathcal{D}(f_j)$
Φ	Set of all possible ϕ
$\mathcal{D}^*$	Wasserstein barycenter of persistence diagrams
$\mathcal{T}(f),\ \mathcal{T}^-(f),\ \mathcal{T}^+(f)$	Merge tree, split tree and join tree of f
$D_E\big(\mathcal{T}(f_i), \mathcal{T}(f_j)\big)$	Edit distance between merge trees $\mathcal{T}(f_i)$ and $\mathcal{T}(f_j)$

$\mathcal{B}(f)$	Branch decomposition tree (BDT) of f
$W_2^{\mathcal{T}}(\mathcal{B}(f_i), \mathcal{B}(f_j))$	L^2-Wasserstein distance between the BDTs $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$
$\mathbb{B}$	Metric space induced by the Wasserstein distance between BDTs
ϕ'	Rooted partial isomorphism between $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$
$\Phi' \subseteq \Phi$	Set of all rooted partial isomorphisms between $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$
$\epsilon_1, \epsilon_2, \epsilon_3$	Parameters of the Wassertein distance between BDTs
$\mathcal{N}(\mathcal{B}(f_i))$	Normalization of $\mathcal{B}(f_i)$ (with normalized persistence pairs)
$\mathcal{S}_{\mathcal{B}} = \{\mathcal{B}(f_1), \ldots, \mathcal{B}(f_N)\}$	Ensemble of N BDTs
$E_F(\mathcal{B})$	Fréchet energy of $\mathcal{B}$ with regard to $\mathcal{S}_{\mathcal{B}}$
$\mathcal{B}^* \in \mathbb{B}$	Wasserstein barycenter of $\mathcal{S}_{\mathcal{B}}$
$\vec{\mathcal{G}}(\mathcal{B}(f_i), \mathcal{B}(f_j))$	Geodesic between $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$
$\vec{\mathcal{G}}(\mathcal{E}, \mathcal{E}') \cdot \vec{\mathcal{G}}(\mathcal{E}, \mathcal{E}'')$	Dot product between two geodesics on $\mathbb{B}$
$\overleftrightarrow{\mathcal{A}_i} = \left(\vec{\mathcal{G}}(\mathcal{E}_O, \mathcal{E}_i), \vec{\mathcal{G}}(\mathcal{E}_O, \mathcal{E}_i')\right)$	Geodesic axis i
$\vec{\mathcal{V}}_i = \vec{\mathcal{G}}(\mathcal{E}_i', \mathcal{E}_i)$	Direction vector of the axis $\overleftrightarrow{\mathcal{A}_i}$
$\overrightarrow{\mathcal{A}_i}(\mathcal{B})$	Arc-length parameterization of $\mathcal{B} \in \overleftrightarrow{\mathcal{A}_i}$ along the geodesic axis $\overleftrightarrow{\mathcal{A}_i}$
$\mathcal{B}_{\overleftrightarrow{\mathcal{A}_i}}$	Projection of $\mathcal{B}$ on the geodesic axis $\overleftrightarrow{\mathcal{A}_i}$
$\overleftrightarrow{\mathcal{A}_j}(\mathcal{B})$	Translation of $\overleftrightarrow{\mathcal{A}_j}$ along $\overleftrightarrow{\mathcal{A}_i}$ with origin $\mathcal{B} \in \overleftrightarrow{\mathcal{A}_i}$
$B_{\mathbb{B}} = \{\overleftrightarrow{\mathcal{A}_1}, \overleftrightarrow{\mathcal{A}_2}, \ldots, \overleftrightarrow{\mathcal{A}_{d'}}\}$	Orthogonal basis of d' geodesic axes
$\alpha \in [0,1]^{d'}$	Coordinate vector of $\mathcal{B}$ in $B_{\mathbb{B}}$
$\widehat{\mathcal{B}}$	Reconstruction of $\mathcal{B}$ with $B_{\mathbb{B}}$
$E_{W_2^{\mathcal{T}}}(B_{\mathbb{B}})$	Fitting energy of the basis $B_{\mathbb{B}}$ with regard to $\mathcal{S}_{\mathcal{B}}$
d_{max}	Maximum number of geodesics in the computation of $B_{\mathbb{B}}$
$\beta' = \|\|\overrightarrow{\mathcal{G}_{d'}}\|\| / (\|\|\overrightarrow{\mathcal{G}_{d'}}\|\| + \|\|\overrightarrow{\mathcal{G}_{d'}'}\|\|)$	Ratio describing the relative norm of $\overrightarrow{\mathcal{G}_{d'}}$ with regard to $\overleftrightarrow{\mathcal{A}_{d'}}$
$\mathcal{P}_{\overrightarrow{\mathcal{V}_{d''}}}(\overrightarrow{\mathcal{V}_{d'}})$	Projection of $\overrightarrow{\mathcal{V}_{d'}}$ onto the direction spanned by $\overrightarrow{\mathcal{V}_{d''}}$
N_1	Maximum size of $\mathcal{B}^*$
N_2	Number of samples along each geodesic axis
$\mathcal{V}(\mathcal{B})$	Vector moving each branch of a BDT $\mathcal{B}$ in the birth/death space
$B(\mathcal{O}) = \{\mathcal{V}_1(\mathcal{O}), \ldots, \mathcal{V}_{d'}(\mathcal{O})\}$	Basis of BDT vectors with origin $\mathcal{O}$
$e(\mathcal{B})$	Projection error of a BDT $\mathcal{B}$ on a basis
$\psi(\mathcal{B})$	Projection of a BDT $\mathcal{B}$ on a basis
Ψ_k^{in}	k^{th} input sub-layer
Ψ_k^{out}	k^{th} output sub-layer
$B_k^{in}(\mathcal{O}_k^{in})$	Basis of the k^{th} input sub-layer
$B_k^{out}(\mathcal{O}_k^{out})$	Basis of the k^{th} output sub-layer
$\Pi_k = \Psi^{out} \circ \Psi^{in}$	k^{th} BDT transformation layer
$n_e,\ n_d$	Number of layers in the encoder and in the decoder
n_{it}	Number of basis projection iterations

1 Introduction

Data can be seen as a medium of information containing insights about a specific phenomenon. They can take various forms and are typically represented as raw numbers (such as physical quantities, like temperature, describing a phenomenon) or characters (e.g. words in a text). Regardless of their forms, a challenge then lies in their analysis, in order to create an intuitive understanding about what they describe. Through this process, we say that data *becomes* information, providing knowledge about a particular subject after undergoing analysis.

In this context, this thesis aims at developping tools to analyze and visualize data in order to ease the understanding of the underlying phenomenom they represent. Specifically, we use *topological* methods, allowing to extract in a concise manner the important information from the data, hence facilitating their analysis and visualization by focusing on the main features of interest. An analogy can be made to understand the intuition behind these methods, intuitively, by how we look at an image representing a face for instance. The raw data contained in the image is a collection of pixels, yet, we do not need to examine each pixel individually for a comprehensive understanding of the image, as this would be excessively time-consuming and not really informative. Instead, we find, and look at, *structures* in the face, like the eyes, the mouth and so on, and how these structures are organized together in the image. Through this action, we move towards a higher level of abstraction, from the raw pixels to distinct, intuitive and meaningful structures in the image.

Topological methods allow that kind of abstraction in a more general context, by encoding the structures within the data in a compact representation. They allow to visually convey the extracted information and to perform the analysis on a more relevant level of details.

1.1 General Context and Motivations

1.1.1 Data Collection, Analysis and Visualization

The process of data collection has evolved significantly over time. In the past, data collection relied on manual methods and elementary tools like thermometers, barometers and so on, making the data acquisition process difficult. However, with the advent of modern technology, we have witnessed a transformation in data collection. Today, an ensemble of advanced sensors, electronic devices and remote monitoring systems enables the easier capture of data with greater accuracy and on a larger scale.

In addition to real-world data, numerical simulations have become an indispensable tool in scientific research and various industries. Data simulation involves the generation of synthetic data through mathematical models and computational techniques. This approach allows researchers to explore scenarios that may be difficult or even impossible to replicate in the physical world. Simulation facilitates controlled experimentation, parameter sensitivity analysis, and hypotheses testing, contributing significantly to the understanding of complex systems and phenomena.

This evolution in data acquisition and simulation has opened new possibilities, making large-scale data analysis essential, in order to automatically discover patterns, extract insights, and make informed decisions from data. In that context, methods from machine learning and data mining, usually based on statistical models, can help that process. Such as clustering, revealing the main trends from data by uncovering inherent patterns and structures within, or dimensionality reduction, creating a visual *map* indicating how the data measurements are organized regarding each other. These techniques have improved fields such as healthcare, astronomy, and natural language processing, giving the possibility to extract valuable knowledge from vast datasets and solve complex problems.

In addition to these tools, data visualization plays a crucial role in data analysis, in order to extract information, insights, and trends hidden within datasets, by transforming raw data into visual representations. With the development of advanced visualization tools, dynamic and interactive visualizations can be created to explore data and enhance their interpretability, giving a deeper and more intuitive understanding to those analyzing complex datasets. It can also help in decision-making and communication of findings with visuals such as charts, graphs, and maps, making complex information more accessible and understandable.

1.1.2 Topological Data Representations

Wether they are acquired or simulated, modern datasets are constantly gaining in detail and complexity, as a consequence of the continuous improvement of acquisition devices or computing resources. This geometrical complexity is a difficulty for interactive data analysis and interpretation. This observation motivates the development of concise yet informative data representations, capable of encoding the main features of interest and visually representing them to the users. In that regard, Topological Data Analysis (TDA) [EH09] has demonstrated its ability to generically, robustly and efficiently reveal implicit structural patterns hidden in complex datasets, in particular in support of analysis and visualization tasks [HLH+16]. Examples of successful applications include turbulent combustion [LBM+06, BWT+11, GBG+14], material sciences [GDN+07, GKL+16, FGT16], nuclear energy [MWR+16], fluid dynamics [KRHH11], bioimaging [CSvdP04, BDSS18, AAPW18], quantum chemistry [GABCG+14, BGL+18, OGT19] or astrophysics [Sou11, SPN+16].

Among the feature representations studied in TDA, the merge tree [CSA00] (Figure 2.7), is a popular instance in the visualization community [CSvdP04, BWT+11, BDSS18]. It concisely and visually encodes the number and salience of the features of interest found in the data and also describes how these features are globally connected.

In many applications, on top of the increasing geometrical data complexity, an additional challenge emerges, related to *ensemble datasets*. These describe a phenomenon not only with a single dataset, but with a collection of datasets, called *ensemble members*, in order to characterize the variability of the phenomenon under study. In principle, a topological representation (like the merge tree) can be computed for each ensemble member. While this strategy has several practical advantages (direct representations of the features of interest, reduced memory footprint), it shifts the analysis problem from an ensemble of datasets to an ensemble of merge trees.

1.1.3 The TORI Project

Throughout the recent years, a momentous increase in the amount of acquired or simulated data have taken place while their exploitation speed has not grown as much [CCF+13]. That difference in speed leads to a major bottleneck in the scientific computing pipeline, leaving a large quantity of data unexploited and unanalyzed.

In light of these challenges, the TORI Project[1] (In-Situ Topological Reduction of Scientific 3D Data) aims to tackle them using topological methods. Specifically, with the goal of reducing this bottleneck, it is mainly motivated by the use of topological representations of the data, that have a memory size orders of magnitude smaller than those of the datasets themselves (hence reducing the quantity of data at hand). TORI is focused on two main axes *(i)* scaling topological methods to reduce the exploitation speed (such as creating approximate methods or parallelizing existing methods in a high-performance setting with shared memory) and *(ii)* developing an analysis framework for datasets based on their topological representations.

This thesis is more concerned with the second axis *(ii)*, especially, in order to tackle the large scale nature of the TORI project, in an analysis of an ensemble of topological descriptors. Moreover, in this line of direction of data reduction using topological methods, we are also concerned about how to find a *good* representation of an ensemble of datasets, i.e. a concise representation that allows further analysis. Intuitively, it can be seen as a compression of an ensemble in a format taking less storage while ensuring that they are still usable afterward.

1.1.4 The Topology ToolKit (TTK)

When developing analysis tools, such as those within the TORI project, it is a logical step to centralize them to ease their usage. The concept of computer library (or just library for short) was introduced in a paper of 1888 of Charles Babbage [Bab88], a pioneer in computer science. It can be seen as a collection of pre-written and reusable codes that can be utilized by others to simplify their process, like by programmers to simplify their code or scientists to use a specific function of a machine. At that time, programs were stored in physical punched cards (a cardboard sheet with a series of non-holes and holes representing respectively 0 and 1). A collection of that kind of cards was said to constitutes a computer library (like a collection of books is said to be a library).

The Topology ToolKit or TTK[2] [TFL+17] is a library for data analysis and visualization using topological methods. It aims at developing efficient and robust data analysis tools and to popularize these topological methods among end users and developers by providing a unified frame-

[1] https://erc-tori.github.io/

[2] https://topology-tool-kit.github.io/

work. Happily for the developers, the underlying code is not stored in punched cards but in text files corresponding to C++ programs. TTK is open-source, the code is accessible to everyone, and is integrated to ParaView [AGL05], a widely-used data analysis and visualization software developed by Kitware, also open-source, based on the Visualization ToolKit (VTK). This integration to ParaView also facilitates the use of TTK for end users, allowing them to use algorithms without knowing how to code and to use topological methods without knowing all their mathematical details. The different functionalities of TTK can also be used by other programmers in their own C++ or Python code.

All the research done within the TORI project, including that presented in this thesis, have their corresponding code implemented and integrated into TTK and are thus publicly available. It also allows the other participants of the project to use the developed methods to go beyond.

1.2 Problem Formulation

The previous section provided some context about data analysis, topological methods as well as the overall research project within which this Ph.D. thesis was carried out. We now further specify the context of our work by describing the scientific problems addressed by this thesis.

1.2.1 Enhancing the Discriminability of Topological Methods

Persistence diagrams (Figure 2.5) are one of the most used representations in TDA if not the most. This trend of research can be explained, at least, by their desirable mathematical properties such as their stability [CEH05] and their simpler structure compared to other topological representations. Several tools such as distances (to compare them), geodesics (to visualize optimum transitions between them), and barycenters (to visualize one persistence diagram *representative* of a set), have already been well studied for persistence diagrams [TMMH14, LCO18, VBT20].

However, persistence diagrams only consist of a set of features (or structures) within a dataset, represented independently, and do not contain any information about how they are connected and *organized* together in the data. It results in a lack of specificity for persistence diagrams which can yield identical data representations for significantly distinct datasets (Figure 3.2), preventing the identification of distinct feature trends within an ensemble. This motivates the use of more discriminant descriptors.

However, no tools such as those developed for persistence diagrams are available for these more discriminant descriptors. The arising question is then: how can we adapt the available framework for persistence diagrams to other descriptors? In this thesis we will address this research question, specifically by focusing on merge trees.

1.2.2 Variability Analysis of Topological Descriptors

While the existing analysis framework for persistence diagrams allows insightful analysis, it does have certain limitations. One of them arises from the notion of barycenter, which only provides information about what we expect to find on average within an ensemble, without conveying any details regarding its variability. A natural direction would involve the extension of the analysis of average to the analysis of variability of an ensemble.

In that context, some machine learning methods allowing to analyze the variability within an ensemble could be used. Several attempts have already been made to conciliate topological descriptors and machine learning methods. They usually consist in transforming these topological descriptors into a different representation that is usually given to machine learning methods (e.g. points in Euclidean space) through a process called *vectorization* [ACE+17, Bub15, RT16, AVRT16, LPW21]. However, vectorizations have several limitations in practice. First, they are prone to approximation errors (resulting from quantization and linearization). Also, they can be difficult to revert (especially for barycenters), which makes them impractical for visualization tasks, especially regarding interpretability. Moreover, their stability is not always guaranteed.

The question that emerges is therefore: how can we enrich this analysis framework to capture variability and without vectorization? In this thesis, we will focus on methods operating directly on topological descriptors taking also into account the previous problem related to discriminability.

1.3 Contributions

We are interested in this thesis in the analysis and the visualization of an ensemble of datasets with the goal of providing tools to end-users to ease their task. Given that objective of tackling ensemble of large datasets within the TORI project (Section 1.1.3), we use methods from Topological Data Analysis (Section 1.1.2) to first represent each member of the ensemble as a topological descriptor and then perform the analysis on

the resulting ensemble of descriptors. Specifically, we want to develop analysis tools for an ensemble of topological descriptors such as the persistence diagrams or the merge trees. The first axis of this thesis focuses on the extension of the existing analysis tools already available for persistence diagrams to the specific case of merge trees, being more informative descriptors. Once these building blocks have been introduced, we can go further by extending the available tools for persistence diagrams and merge trees to variability analysis, constituting the second axis of this thesis. Finally, the third axis encompasses both by defining all these methods, in the light of data reduction, acting as a compression seeking an informative and concise representation of an ensemble of topological descriptors. All methods developed in this thesis are implemented and available in the open-source library Topology ToolKit [TFL+17], and we provide usage examples for each of them with visualization pipelines in ParaView and also using programming languages. Overall, we present in this thesis the following contributions to the development of discriminant analysis methods for ensembles of topological descriptors:

Analysis Framework for Merge Trees

To tackle the lack of discriminability of persistence diagrams, we propose to adapt the tools available for these objects to merge trees. For that, we introduce novel methods to compute distances, geodesics and barycenters of merge trees. In particular, we bridge the gap between the edit distance between merge trees [SMKN20] and the existing methods for geodesics and barycenters computation of persistence diagrams [TMMH14]. The naive combination of these two methods can result in objects not respecting the essential properties of merge trees, we therefore introduce mechanisms to ensure their preservation. First, we modify and simplify the original edit distance between merge trees by constraining it, in order to prevent these inconsistencies cases. Then, and more importantly, we propose a *local normalization* strategy guaranteeing the preservation of the mentioned properties. We design parallel algorithms for the computation of the introduced methods and, additionally, we extend them in order to revisit the k-means clustering algorithm using merge trees. This contribution is presented in Chapter 3.

Variance Analysis of Ensembles of Topological Descriptors

In order to develop more advanced tools for the analysis of ensembles of topological descriptors we want to go beyond the simple notion of average that is the barycenter and study the variability of such ensembles. Inspired by previous work on the optimal transport of histograms [SC15, CSB+18], we adapt to merge trees and persistence diagrams the Principal Component Analysis (PCA) framework, a popular dimensionality reduction method designed to identify the principal trends of variability of an ensemble of datasets. We provide tools for visually inspecting the features within the ensemble being the most responsible for the variance, those varying the most. These tools help to understand how the features are related to each other and to identify those that are influenced by similar patterns of variation in the ensemble. It also allows to relate features and members of the ensemble by revealing how features are responsible for the arrangement of the members with each other in the ensemble. This contribution is detailed in Chapter 4.

Non-Linear Pattern Analysis of Ensembles of Topological Descriptors

PCA will find linear relationships among the features of the ensemble. For instance, it can indicates that when a first structure is prominent in a member of the ensemble, then a second structure is also expected to be prominent, but only in a proportional manner. Therefore, it could miss more complex patterns such as exponential or cyclical relationships or when structures are related in a piecewise manner (when the presence of a first structure implies the presence in a proportional manner of a second one for some members, but in an inversely proportional manner for other members). To tackle this problem a generalization of PCA can be used, based on a neural network architecture called the Auto-Encoder. It results in a method allowing to *unfold* non-linear variability patterns in a linear manner, making them easily detectable and analyzable. Inspired by how PCA can be generalized to Auto-Encoder we extend the previous approach in order to adapt to merge trees and persistence diagrams the Auto-Encoder framework. Specifically, we propose a novel neural network layer capable of processing natively such topological descriptors without prior vectorization. This contribution is described in Chapter 5.

Applications to Ensemble Analysis

We show the utility of all our different contributions with visualization, machine learning and data reduction tasks.

We use the assignment between structures given by our distance between merge trees (Section 3.3) for feature tracking in a time-varying ensemble of datasets (Section 3.6.1).

By extending the proposed barycenter method for merge trees (Section 3.5) to the k-means algorithm we can operate a clustering of an ensemble of datasets based on their merge trees (Section 3.6.3), to summarize and automatically form homogeneous groups in an ensemble, revealing the main trends of features within.

Our adaptation to persistence diagrams and merge trees of PCA (Section 4.2) and Auto-Encoder (Section 5.2) provide a dimensionality reduction framework (Section 4.4.2 and Section 5.4.2), allowing to visually represent in $2D$ all members of the ensemble based on their topological descriptors, revealing clusters, trends and relationships among the members.

Finally, our contributions can be used for data reduction. Our geodesic method for merge trees Section 3.4 can be used for temporal reduction, in order to sub-sample a temporal sequence of merge trees, by automatically identifying key frames of the sequence used to reconstruct the other frames Section 3.6.2. And our adaptation of PCA (Section 4.2) and Auto-Encoder (Section 5.2) allow to compress and reconstruct an ensemble of persistence diagrams or merge trees being still usable for usual tasks such as feature tracking or clustering (Section 4.4.1 and Section 5.4.1).

1.4 Outline

After this introduction, the manuscript is structured in the following way.

First, in Chapter 2 we introduce the theoretical background on Topological Data Analysis upon which our work is based.

Then, in Chapter 3, we present our methods for computing distances, geodesics and barycenters of merge trees, as well as its extension to a clustering algorithm using merge trees.

After this, in Chapter 4, we detail our adapation of the Principal Component Analysis framework to persistence diagrams and merge trees, resulting in a dimensionality reduction method for these objects.

Chapter 5 describes our novel network layer allowing to natively pro-

cess persistence diagrams and merge trees and its use in the context of Auto-Encoders.

Finally, Chapter 6 marks the endpoint of this thesis by offering a succinct overview of its different contributions and limitations along with a discussion about open problems that deserve future exploration.

Theoretical Background

2

Contents

This chapter introduces the various concepts upon which our work is based. We start with the formalism underlying the data representation, from its support to how the data itself is encoded with the notions of *manifold* and *scalar field* respectively. Then, we introduce notions of *homology* allowing to characterize data under the perspective of topology. Finally, we introduce different topological data representations used in our work, such as *critical points*, *persistence diagrams* and *merge trees*, along with analysis tools available for them such as distances and barycenters.

This chapter contains definition adapted from [Tie18] and [EH09]. We refer to the reference books [EH09] and [Zom10] for detailed introduction to computational topology.

2.1 Input Data Representation

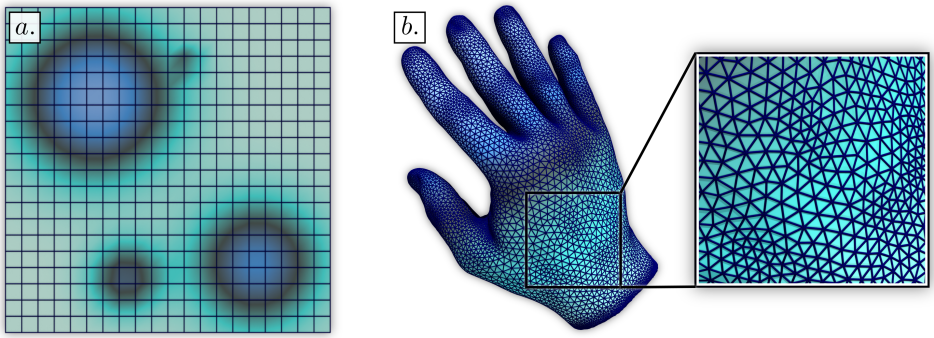


Figure 2.1 – *Examples of structured grid (a) and unstructured grid (b) being, in practice, the support of data on which the information, the* values*, are encoded (visualized here with the color map from the low values in green to the high values in blue).*

In practice, we are interested in data in the form of measurements defined on geometrical objets which we call *grids*. These are defined as collections of points connected with edges, where each point is associated with a value. We can enumerate mainly two types of grids, regular grids that can be seen as an image (a rectangular grid of pixels) and unstructured grids exhibiting arbitrary structures. Examples are shown in Figure 2.1. We will first describe the mathematical formalism allowing to describe these grids by introducing the concept of *manifold*. We will then detail how they can be represented in practice as grids through the general notion of *simplicial complex*, allowing to represent even more complicated building blocks than points and edges. Finally, we will present the mathematical notions related to how the values are encoded in the grid with the notion of *scalar field*.

2.1.1 Domain Representation

A manifold can be intuitively seen as a curved object with arbitrary shape which looks flat when we zoom in small regions of it. It is locally equivalent to a Euclidean space, a flat space where straightforward geometric rules apply, but can have more complex structures globally that are curved and deviates from flatness. A common example of manifolds is the surface of a planet, which is spherical when we look from far but look like a flat space when we are on them.

Formally it is an ensemble of points that are related to each other with a structure called topology. This relation between the points is expressed with the notion of open sets, that are usually chosen to describe the neigh-

borhood of the points. They can be seen as various subsets of all the points, indicating which points are considered as neighbors. The topology specifies which subsets of the space are considered open. Its definition is designed to build the essential properties needed for continuity and other topological concepts without having to rely on specific distance metrics (as in Euclidean spaces) to convey this notion of closeness between points.

Definition 2.1 *(Topological Space, Topology, Open Sets)* A set $\mathbb{X}$ is called a *topological space* if there exists a collection T of subsets of $\mathbb{X}$ such that:

- The empty set $\emptyset$ and $\mathbb{X}$ itself belong to T.
- Any union of elements of T belongs to T.
- Any finite intersection of elements of T belongs to T.

T is said to be a *topology* of $\mathbb{X}$ and its elements are the *open sets* of $\mathbb{X}$.

Definition 2.2 *(Function)* Given two topological spaces $\mathbb{X}_1$ and $\mathbb{X}_2$, a *function* $f : \mathbb{X}_1 \to \mathbb{X}_2$ associates each element of $\mathbb{X}_1$ to a unique element of $\mathbb{X}_2$.

Definition 2.3 *(Injection)* A function $f : \mathbb{X}_1 \to \mathbb{X}_2$ is an *injection* if for each pair $x_1, x_2 \in \mathbb{X}_1$ such that $x_1 \neq x_2$ then $f(x_1) \neq f(x_2)$. f is said to be injective.

Definition 2.4 *(Bijection)* A function $f : \mathbb{X}_1 \to \mathbb{X}_2$ is a *bijection* if for each element $x_2 \in \mathbb{X}_2$ there is a unique element $x_1 \in \mathbb{X}_1$ such that $f(x_1) = x_2$. f is said to be bijective.

Definition 2.5 *(Continuous Function)* A function $f : \mathbb{X}_1 \to \mathbb{X}_2$ is *continuous* if for each open set t_2 of $\mathbb{X}_2$, $f^{-1}(t_2)$ is an open set of $\mathbb{X}_1$. With $f^{-1} : \mathbb{X}_2 \to \mathbb{X}_1$ being the inverse of f.

A continuous function will transform a topological space into another such that points in a given neighborhood in the first space will be mapped to points in a specific neighborhood in the second space. It can be seen as if it is preserving the neighborhood of each point, points being neighbors in the first space will also be neighbors after the mapping.

Definition 2.6 *(Homeomorphism)* A function $f : \mathbb{X}_1 \to \mathbb{X}_2$ is a *homeomorphism* if it is a continuous bijection and if its inverse f^{-1} is also continuous. $\mathbb{X}_1$ and $\mathbb{X}_2$ are said to be homeomorphic.

In topology, homeomorphisms is usually used to compare topological spaces, in several contexts, two topological spaces are often considered as *equivalent* if they are homeomorphic. More specifically, homeomorphic

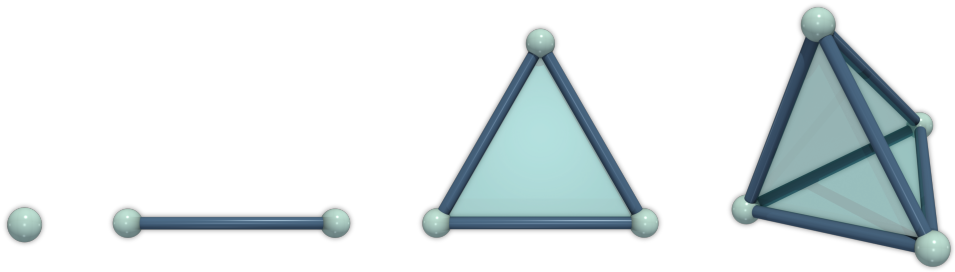


Figure 2.2 – *Visualization of d-simplices for $d \leq 3$, from left to right: a vertex, an edge, a triangle and a tetrahedron.*

spaces can be transformed into each other by mappings that do not involve tearing or gluing parts of the spaces.

Definition 2.7 *(Manifold)* A topological space $\mathbb{M}$ is a *d-manifold* if every point p of $\mathbb{M}$ has a neighborhood homeomorphic to an open subset of the Euclidean space $\mathbb{R}^d$.

Manifold can be seen as a smooth definition of a shape. In order to represent it on numeric devices we need to discretize it with building blocks such as pixels are discrete building blocks of an image.

The most fundamental building blocks are points. To describe how they can be connected, this concept of points can be generalized to the notion of simplices. These are defined as all the points *inside* a specific ensemble of points. For instance, for a single point it is itself. For two points it will be all the points between them (by drawing a line between the two points). For three points it will corresponds to all the points in the triangle formed by these points and so on. Examples of them are illustrated in Figure 2.2.

Definition 2.8 *(Convex Set)* A set $\mathbb{C}$ of an Euclidean space $\mathbb{R}^n$ is *convex* if for any pair of points x and y of $\mathbb{C}$, the point $tx + (1-t)y$ is also in $\mathbb{C}$, with $t \in [0,1]$.

Definition 2.9 *(Convex Hull)* The *convex hull* of a set of points $\mathbb{P}$ of an Euclidean space $\mathbb{R}^n$ is the unique minimal convex set containing all points of $\mathbb{P}$.

Definition 2.10 *(Simplex)* A *d-simplex* is the convex hull of $d+1$ affinely independent points of an Euclidean space $\mathbb{R}^n$, with $0 \leq d \leq n$. For instance:

- A 0-simplex is a *vertex*.
- A 1-simplex is a *edge*.
- A 2-simplex is a *triangle*.
- A 3-simplex is a *tetrahedron*.

Definition 2.11 *(Face)* A *face* τ of a d-simplex σ is a m-simplex (with $m < d$) of a nonempty subset of $m+1$ points among the $d+1$ points of σ, called a m-face of σ and noted $\tau < \sigma$.

The faces of a simplex correspond to all the lower-dimensional simplices it contains. Considering this, we can think about a simplex as a way of expresssing neighbor relations between lower-dimensional simplices (similar to how it is encoded by open sets in a topological space).

These building blocks, the simplices, can then be combined together to create a more complex structure called a simplicial complex. It is an ensemble of simplices that are *glued* together on the faces they have in common.

Definition 2.12 *(Simplicial Complex)* A *simplicial complex* $\mathcal{K}$ is a set of simplices such that:

- Every face of a simplex of $\mathcal{K}$ is also in $\mathcal{K}$.
- The intersection of any two simplices of $\mathcal{K}$ is either empty or a common face of both of them.

Definition 2.13 *(Triangulation)* A *triangulation* $\mathcal{T}$ of a topological space $\mathbb{X}$ is a simplicial complex $\mathcal{K}$ such that the union of its simplices is homeomorphic to $\mathbb{X}$.

Definition 2.14 *(Piecewise Linear Manifold)* A *piecewise linear (PL) manifold* $\mathcal{M}$ is the triangulation of a manifold $\mathbb{M}$.

By definition, a PL manifold is therefore a simplicial complex that accurately represent a manifold.

2.1.2 Scalar Field Representation

Given a simplicial complex, we can then define some notions to represent the values that are stored on its simplices (like the grey intensity of each pixel of a greyscale image). These values are the information that we consider in input, while the simplicial complex is *only* a support for them to lie on.

Given a function mapping each 0-simplex of the simplicial complex to a value, we now define how to compute the value on any point inside any d-simplex. As an example, given three points and three edges constituing a triangle along with a function on the points, the following definition allows to express the value of any points inside the triangle.

It uses the notion of barycentric coordinates, expresssing the coordinates of a point as a linear combination (a weighted sum) of the coordi-

nates of other points. Then, once we have the weights to express a point as a combination of other ones, we can express its value as a linear combination, using the same weights, of the values of the other points.

Definition 2.15 *(Barycentric Coordinates)* Let a point p in $\mathbb{R}^d$ and a d-simplex σ. The *barycentric coordinates* of p relatively to σ are the coefficients $\alpha_0, \ldots, \alpha_d$ of a linear combination of the 0-simplices $v_0, \ldots, v_d$ of σ such that $p = \sum_{i=0}^{d} \alpha_i v_i$ with $\sum_{i=0}^{d} \alpha_i = 1$ and $\alpha_i \in \mathbb{R}$.

Definition 2.16 *(Piecewise Linear Scalar Field)* Let a triangulation $\mathcal{T}$ and a function h that maps its 0-simplices to $\mathbb{R}$. A *piecewise linear (PL) scalar field* f on $\mathcal{T}$ is a function that maps any point p of a d-simplex σ of $\mathcal{T}$ to a value $f(p) = \sum_{i=0}^{d} \alpha_i h(v_i)$ with $\alpha_0, \ldots, \alpha_d$ being the barycentric coordinates of p relatively to σ and $v_0, \ldots, v_d$ the 0-simplices of σ.

2.2 Notions of Homology

Once the domains at hand have been introduced, a natural question that arises is how can we compare them? Answering this question allows to put in groups, to classify, topological spaces that we consider equivalent according some properties, or on the contrary, to know the properties that differ between some topological spaces. One way to compare topological spaces is to verify if they are homeomorphic.

To avoid to search for homeomorphisms between two spaces one can instead look at the properties of a space that are preserved through those transformations, those properties are called *topological invariants*. If two spaces have different invariants, it implies that they are not homeomorphic, highlighting their distinct topological nature (the opposite is however not true, spaces having same invariants are not necessarly homeomorphic). Those properties that are unaffected by homeomorphisms simplify the study of spaces by reducing complex shapes to their essential topological attributes. This abstraction allows to focus on the core and global properties of a space without being encumbered by geometric details.

In that context, tools from algebraic topology allow to study these topological spaces from a global point of view and to capture their essential and intrinsic properties. Specifically, homology is a framework that defines these topological invariants as *holes* of the domain. Holes are properties that can not be changed with homeomorphisms, making them robust and reliable properties to study and compare spaces. The goal of

simplicial homology is to mathematically define this notion of holes in a simplicial complex. Additionally, persistent homology captures not only the invariants of a space but their evolution across different scales according to a scalar field representing the input data.

2.2.1 Simplicial Homology

The first building block needed to detect holes in a simplicial complex $\mathcal{K}$ is the notion of a p-chain, corresponding intuitively to a collection of p-simplices in $\mathcal{K}$. This notion is defined formally as a sum of these simplices, including a coefficient for each of them representing how many times they appear in the chain.

Definition 2.17 *(p-chain)* A *p-chain* of a simplicial complex $\mathcal{K}$ is a sum of p-simplices of $\mathcal{K}$, defined as: $\sum_i m_i \sigma_i$ with $m_i \in \mathbb{Z}$ and each σ_i is a simplex of $\mathcal{K}$.

Here, we will restrict the coefficients m_i to be 0 or 1 (as it is usually done), in the space of natural numbers modulo 2 : $\mathbb{Z}/2\mathbb{Z}$.

Given a p-chain we will now define how to determine if it indicates the presence a hole. In homology, holes can be classified by their dimension and a hole of dimension p will be *surrounded* by a p-chain. Intuitively, holes of dimension 0 corresponds to connected components, dimension 1 to loops, dimension 2 to voids (or cavities) and so on. In order to detect holes in a simplicial complex, one first need to detect *cycles* within it, i.e. a chain surrounding a hole (like the perimeter of a circle).

To know if a p-chain c is a cycle we need to search for *extremities* in c, being formally defined with the notion of boundary. The boundary of a p-chain will yield the $(p-1)$-faces of the simplices of the chain that are not shared with another simplex of the chain.

The boundary of a p-chain will be intuitively, if there exists, a collection of $(p-1)$-simplices i.e. a $(p-1)$-chain. If the boundary is empty the p-chain corresponds to a cycle called a p-cycle.

Definition 2.18 *(Boundary of a p-simplex)* The *boundary of a p-simplex* σ is the sum of its $(p-1)$-faces, defined as the boundary operator $\partial_p(\sigma) = \sum_i \tau^i_{p-1}$ with τ^i_{p-1} being the i^{th} $(p-1)$-face of σ. The boundary of a 0-simplex is 0 (empty).

Definition 2.19 *(Boundary of a p-chain)* The *boundary of a p-chain* c is the sum of the boundary of each of its simplices: $\partial_p(c) = \sum_i m_i \partial_p(\sigma_i)$.

The idea that $(p-1)$-faces shared by two p-simplices (like vertices by

two edges) are not included in the boundary is formalized here through the usage of coefficients modulo 2.

For instance, considering a 1-chain c of 3 edges AB, BC and CD (being 1-simplices linking the 0-simplices A, B and C according to their name). The boundary operator gives us:

$$\begin{aligned}\partial_1(c) &= \partial_1(AB) + \partial_1(BC) + \partial_1(CD)\\ &= A + B + B + C + C + D\\ &= A + D\end{aligned}$$

Definition 2.20 *(p-cycle)* A *p-cycle* of a simplicial complex $\mathcal{K}$ is a p-chain with empty boundary.

Definition 2.21 *(Group of p-cycles)* The *group of p-cycles* of a simplicial complex $\mathcal{K}$ corresponds to all p-cycles of $\mathcal{K}$, noted $\mathcal{Z}_p(\mathcal{K})$.

Equivalently, the group of p-cycles is noted $\ker(\partial_p)$, the kernel of the boundary operator ∂_p, representing all possible inputs of ∂_p (the p-chains of $\mathcal{K}$) such that ∂_p gives 0 (the empty boundary) when applied on them.

Detecting cycles is not sufficient for detecting holes. Indeed, a cycle could be the border of *something filled* (like the perimeter of a disk). Therefore, we need a way to differentiate between the cycles that are filled and those that are not (corresponding to actual holes). The idea relies in the fact that for a p-cycle to be filled there needs to exist a $(p+1)$-chain that has this cycle, and only it, as boundary.

We can then define the set of all cycles that are not the boundary of a higher dimensional chain, thus indicating the presence of a hole. More formally, the p^{th} homology group consists of all the p-cycles that are not the boundary of a $(p+1)$-chain. Cycles indicating the presence of the same hole are called homologous and are grouped together in the same *homology class*.

Definition 2.22 *(p-boundary)* A *p-boundary* of a simplicial complex $\mathcal{K}$ is the boundary of a $(p+1)$-chain.

Definition 2.23 *(Group of p-boundaries)* The *group of p-boundaries* of a simplicial complex $\mathcal{K}$ corresponds to all p-boundaries of $\mathcal{K}$, noted $\mathcal{B}_p(\mathcal{K})$.

Equivalently the group of p-boundaries is noted $\text{im}(\partial_{p+1})$, the images of the boundary operator ∂_{p+1}, representing all the possible outputs of ∂_{p+1} applied on the $(p+1)$-chains of $\mathcal{K}$.

Definition 2.24 *(Homology Group)* The p^{th} *homology group* of a simplicial complex $\mathcal{K}$ is the quotient group of its p-cycles modulo its p-boundaries: $\mathcal{H}_p(\mathcal{K}) = \mathcal{Z}_p(\mathcal{K})/\mathcal{B}_p(\mathcal{K})$.

Equivalently $\mathcal{H}_p(\mathcal{K}) = \ker(\partial_p)/\operatorname{im}(\partial_{p+1})$.

A quotient space V/N can be intuitively seen as the space V with everything in N *collapsed* to 0. For instance, with $\mathbb{Z}/2\mathbb{Z}$ we consider as 0 everything in $2\mathbb{Z}$ (that is $2, 4, 6, \ldots$). Moreover, we consider equivalent everything that has a difference of an element in $2\mathbb{Z}$ such as 1 and 3, having a difference of 2, since $2 \in 2\mathbb{Z}$ and that everything in $2\mathbb{Z}$ is *collapsed* to 0 then 1 and 3 are considered equivalent.

The formal definition of the p^{th} homology group tells us, like we have already mentioned, that p-cycles being p-boundaries will be considered 0, inexistent in the group. But most importantly, it allows to define formally how two p-cycles are considered homologous. Given the intuition of a quotient space and the formal definition of the p^{th} homology group, two cycles are therefore considered equivalent if their difference corresponds to an element in $\mathcal{B}_p(\mathcal{K})$, a p-boundary. Intuitively (and since we work with coefficients modulo 2), it tells us that if there exists a $(p+1)$-chain whose boundary corresponds to the formal sum of two p-cycles then they are considered homologous and are grouped in the same class. Another intuitive way of considering two cycles homologous is if they can be *continuously* transformed into each other while preserving neighbourhood during the transformation.

Each p^{th} homology group $\mathcal{H}_p(\mathcal{K})$ can be seen as a collection of all homology classes representing a p-dimensional hole of $\mathcal{K}$. For instance, $\mathcal{H}_0(\mathcal{K})$ will contains classes representing connected components, $\mathcal{H}_1(\mathcal{K})$ the loops, $\mathcal{H}_2(\mathcal{K})$ the voids (or cavities) and so on.

Finally, we can define the p^{th} Betti number being the number of independent classes of $\mathcal{H}_p(\mathcal{K})$. It corresponds to the number of independent p-dimensional holes in $\mathcal{K}$. By independent, we mean here that a p-cycle could be the sum of two other p-cycles and, as a result, is not included in the count of the p^{th} Betti number.

Definition 2.25 *(Betti Number)* The p^{th} Betti number of a simplicial complex $\mathcal{K}$ is the rank of its p^{th} homology group: $\beta_p(\mathcal{K}) = rank(\mathcal{H}_p(\mathcal{K}))$, corresponding to its number of independent classes.

2.2.2 Persistent Homology

The tools of persistent homology are designed to track the evolution of the homology groups during a *filtration* of a topological space. Intuitively, a filtration of a topological space $\mathbb{X}$ is a process that begins by selecting a subset of $\mathbb{X}$ and then, at each subsequent step, adds another subset of $\mathbb{X}$ to the ones already included. One can then compute the homology groups at each step to understand how they evolve along the filtration, to know if some classes appear at a specific step or disappear at another one. For instance, adding a subset of $\mathbb{X}$ being disconnected from the previously added subsets will create another connected component, hence changing the 0^{th} homology group by adding a new class. On the contrary, adding a subset of $\mathbb{X}$ resulting in the merge of two distinct connected components will change the 0^{th} homology group by merging a class into another.

Definition 2.26 *(Filtration)* A *filtration* is an ordered and indexed collection of topological spaces $\{\mathbb{X}_\alpha\}_{\alpha\in\mathbb{R}}$ that are nested by inclusion such that $\mathbb{X}_i \subseteq \mathbb{X}_j$ for $i \leq j$.

A scalar field induces a filtration based on the notion of sub-level set. This filtration consists in varying a threshold from the lowest values of the function to the highest ones and to consider, at each step of the sequence, all the points having a function value below the current threshold. For a given scalar field, one can then look at which homology classes persist the most during its filtration.

Definition 2.27 *(Level Set)* The *level set* $f^{-1}(w)$ of an isovalue $w \in \mathbb{R}$ relatively to a PL scalar field $f : \mathcal{M} \to \mathbb{R}$ on a PL manifold $\mathcal{M}$ is the inverse of w by f defined as $f^{-1}(w) = \{p \in \mathcal{M} \mid f(p) = w\}$, i.e. the ensemble of points $p \in \mathcal{M}$ such that $f(p) = w$.

Definition 2.28 *(Sub-Level and Super-Level Set)* The *sub-level set* $f^{-1}_{-\infty}(w)$ of an isovalue $w \in \mathbb{R}$ relatively to a PL scalar field $f : \mathcal{M} \to \mathbb{R}$ on a PL manifold $\mathcal{M}$ is the inverse of $(-\infty, w]$ by f defined as $f^{-1}_{-\infty}(w) = \{p \in \mathcal{M} \mid f(p) \leq w\}$. The *super-level set* is defined similarly as $f^{-1}_{+\infty}(w) = \{p \in \mathcal{M} \mid f(p) \geq w\}$).

However, when working with simplicial complexes, we need to ensure that each step of the sequence of the filtration is indeed a simplicial complex and is nested by inclusion with the subsequent steps. To do so, the faces of a simplex σ need to be added before σ itself and the simplices should not be *cut* during the filtration as the sub-level set does. It can be easily achieved with the so-called lower star filtration [EH09], a combina-

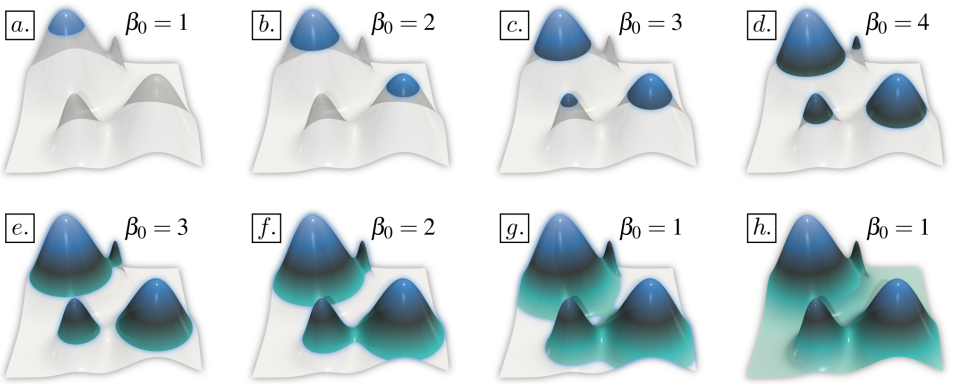


Figure 2.3 – *Given a PL scalar field, the lower star filtration varies a threshold from the lowest scalar value to the highest. The color map indicates the low values in blue and the high values in green.At each step of the filtration, the homology group of the simplices with function value below this threshold is considered. The change in the topology of the domain occurs after specific threshold values that corresponds to the creation of new connected components (a, b, c, d) or the merge of existing components (e, f, g).*

torial equivalent and adaptation of the sub-level set filtration to simplicial complexes.

Definition 2.29 *(Star)* The *star* $St(v)$ of a vertex v of a simplicial complex $\mathcal{K}$ is the set of simplices having v as a face, defined as $St(v) = \{\sigma \in \mathcal{K} \mid v < \sigma\}$.

Definition 2.30 *(Lower Star)* The *lower star* $St^-(v)$ of a vertex v of a simplicial complex $\mathcal{K}$ relatively to a PL scalar field $f : \mathcal{K} \to \mathbb{R}$ is the subset of the simplices in the star of v whose vertices have strictly lower value than v, defined as $St^-(\sigma) = \{\sigma \in St(v) \mid \forall u \in \sigma, f(u) < f(v)\}$.

The lower star filtration adds for each vertex its entire lower star to the filtration. When the function f is injective on the vertices, it results in a filtration in n steps, with n being the number of vertices, starting with the vertex having the lowest value and ending with the one having the highest (Figure 2.3). Making any scalar field injective can be easily achieved without even modifying the scalar field using a symbolic perturbation [EM90], in practice we use an injective offset field on the vertices corresponding to their index in the simplicial complex allowing to disambiguate equalities.

The sequence of sub-complexes $\mathcal{K}_i$ given by the lower star filtration (with $0 \leq i < n$, and n being the number of vertices of $\mathcal{K}$) results in a sequence of mappings between their homology groups:

$$\mathcal{H}_p(\mathcal{K}_0) \to \mathcal{H}_p(\mathcal{K}_1) \to \ldots \to \mathcal{H}_p(\mathcal{K}_{n-1}) = \mathcal{H}_p(\mathcal{K}) \tag{2.1}$$

These mappings describe how p-dimensional holes evolve throughout

the filtration. Using these mappings, we can define the notion of p^{th} persistent homology groups. These correspond, for each step of the filtration, to the classes of the homology group of a previous step remaining at the current step, those that *persists* through the filtration.

Definition 2.31 *(Persistent Homology Groups)* Let a filtration and the mappings induced by inclusion, $f_p^{i,j} : \mathcal{H}_p(\mathcal{K}_i) \to \mathcal{H}_p(\mathcal{K}_j)$ for $0 \leq i \leq j \leq n$, between the corresponding sequence of homology groups. The p^{th} *persistent homology groups* are the images of these homomorphisms noted $\mathcal{H}_p^{i,j} = \text{im } f_p^{i,j}$.

Each p^{th} homology group $\mathcal{H}_p^{i,j}$ is a subset of $\mathcal{H}_p(\mathcal{K}_j)$ corresponding to what have become the classes of $\mathcal{H}_p(\mathcal{K}_i)$ at the j^{th} step of the filtration.

They allow to differentiate between the classes that appear at a specific step and the remaining classes of the previous step. Specifically, we can say that a class γ is *born at* $\mathcal{K}_i$ (the i^{th} step of the filtration) if $\gamma \in \mathcal{H}_p(\mathcal{K}_i)$ (it is a class of the current step) and $\gamma \notin \mathcal{H}_p^{i-1,i}$ (that does not come from the previous step). Moreover, a class γ that is born at $\mathcal{K}_i$ *dies at* $\mathcal{K}_j$ if $f_p^{i,j-1}(\gamma) \notin \mathcal{H}_p^{i-1,j-1}$ (at the $(j-1)^{th}$ step, the class γ is still not merged to any classes existing before the i^{th} step) and $f_p^{ij}(\gamma) \in \mathcal{H}_p^{i-1,j}$ (at the j^{th} step, γ merges to a class existing before the i^{th} step).

When two classes merge at some point of the filtration (for instance, an edge is added resulting in the merge of two connected components) we say that the most recently born class *dies* in favor of the oldest. This is commonly refered as the *Elder rule* [EH09]. This allows to associate to each class two values, its *birth* and its *death*, the scalar values for which this class appears in the filtration and, respectively, merges with another one. We can then define the *persistence* of a class corresponding to the difference between its death and its birth. Intuitively, classes with small persistence could be seen as noise in the data, classes that appear and die very quickly during the filtration (like very small pit of values for instance). On the contrary, classes with high persistence are often considered as the important structures of the scalar field.

2.3 Critical Points

During the lower star filtration, the number of independent classes in each p^{th} homology group (the p^{th} Betti number) changes at specific values and at specific vertices of the domain called the critical points [Ban67]. These points indicate where the function stops changing in the same *direction*, when it transitions from increasing to decreasing or vice-versa.

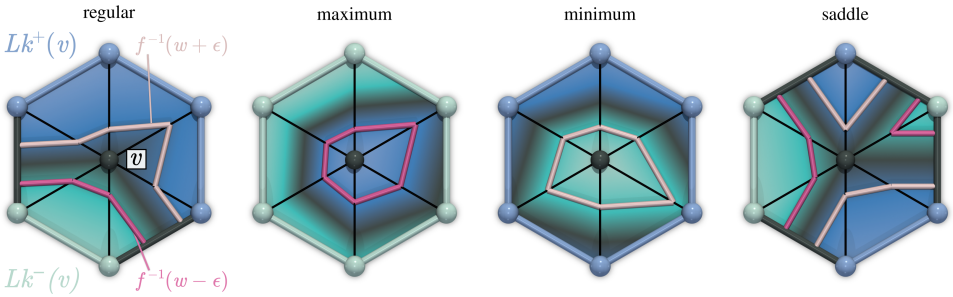


Figure 2.4 – *Regular and critical points on a 2D PL manifold. A vertex v (black sphere) can be classified according the connectivity of its lower link $Lk^-(v)$ (light green spheres and edges) and its upper link $Lk^+(v)$ (light blue spheres and edges). Level sets above and below the value $w = f(v)$ are shown respectively in light pink and pink. A change in the topology of the level sets occurs as they pass w if v is a critical point.*

For instance, a vertex having the lowest value among its *neighbors* will be considered as a critical point (being a local minima), during the filtration it will be the first among its neighbors to appear hence corresponding to the creation of a connected component. To define the neighborhood of a vertex v we can look at the faces of the simplices containing v and keep only those that do not contain v. This is defined as the link of a vertex.

Definition 2.32 *(Link)* The *link* $Lk(v)$ of a vertex v of a simplicial complex $\mathcal{K}$ is the set of faces of the simplices having v as a face (i.e. of $St(v)$) not containing v, defined as $Lk(v) = \{\tau \in \mathcal{K} \mid \tau < \sigma, \sigma \in St(v), \tau \cap v = \emptyset\}$.

To determine if a vertex v is a critical point, we need to examine how two specific subsets of the link of v, called the lower and the upper link, are organized. They include the simplices of the link having vertices with lower (respectively higher) values than v.

Definition 2.33 *(Lower and Upper Links)* The *lower link* $Lk^-(v)$ (respectively the *upper link* $Lk^+(v)$) of a vertex v given a PL scalar field f is the subset of the simplices of the link $Lk(v)$ such that each of their vertices has a strictly lower (respectively higher) function value than v. It is defined for the lower link as $Lk^-(v) = \{\sigma \in Lk(v) \mid \forall v' \in \sigma, f(v') < f(v)\}$ (respectively $Lk^+(v) = \{\sigma \in Lk(v) \mid \forall v' \in \sigma, f(v') > f(v)\}$).

We say that a vertex is a regular point if both its lower and upper links consist of a single connected component without any holes. We can think of this as if a regular point is a transition between two chunks of respectively lower and higher valued vertices, hence, there is no change in the direction of the function at that point. Slightly more formally, each of these links should be *simply connected*, meaning that all paths between

two points in a link can be continuously transformed into each other. The other vertices are called critical points (Figure 2.4).

Definition 2.34 *(Critical Point)* Let a PL scalar field $f : \mathcal{M} \to \mathbb{R}$ defined on a PL manifold $\mathcal{M}$. A vertex v of $\mathcal{M}$ is a *regular point* if and only if both $Lk^-(v)$ and $Lk^+(v)$ are simply connected. Otherwise, v is a *critical point* of f and $f(v)$ is called a critical isovalue.

Definition 2.35 *(Extremum)* Let a PL scalar field $f : \mathcal{M} \to \mathbb{R}$ defined on a PL manifold $\mathcal{M}$. A critical point v is a *minimum* (respectively *maximum*) of f if and only if $Lk^-(v)$ (respectively $Lk^+(v)$) is empty.

Definition 2.36 *(Saddle)* Let a PL scalar field $f : \mathcal{M} \to \mathbb{R}$ defined on a PL manifold $\mathcal{M}$. A critical point v is a *saddle* if and only if it is neither a minimum nor a maximum of f.

Critical points can be classified by their index $\mathcal{I}$, on a d-manifold, minima will have index 0, maxima index d and saddle points from 1 to $d-1$. There will be no saddle points on a d-manifold when $d \leq 1$ and $d-1$ different types of saddle points when $d > 1$. Saddle points correspond to vertices where the level set merges or split connected components locally.

In 3D, connected components of $f^{-1}_{-\infty}(w)$ are created at local minima and destroyed at 1-saddles. One-dimensional cycles (loops) are created at 1-saddles and destroyed at 2-saddles and voids are created at 2-saddles and destroyed at maxima.

2.4 Persistence Diagrams

During the lower star filtration, each (independent) homology class can be associated to a pair of critical points (themselves associated to scalar values), corresponding to the *birth* and the *death* of this class. All these classes can be encoded in a mathematical object called the persistence diagram, corresponding to a visual summary of the topological features (i.e. connected components, independent cycles, voids) of $f^{-1}_{-\infty}(w)$.

Specifically, in the domain, each topological feature of $f^{-1}_{-\infty}(w)$ (corresponding to a homology class) is associated with a unique pair of critical points (c, c'), its birth and its death. The Elder rule [EH09] states that critical points can be arranged in pairs according to this observation, such that each critical point appears in only one pair (c, c'), with $f(c) < f(c')$ and $\mathcal{I}(c) = \mathcal{I}(c') - 1$. For instance, if two connected components of $f^{-1}_{-\infty}(w)$

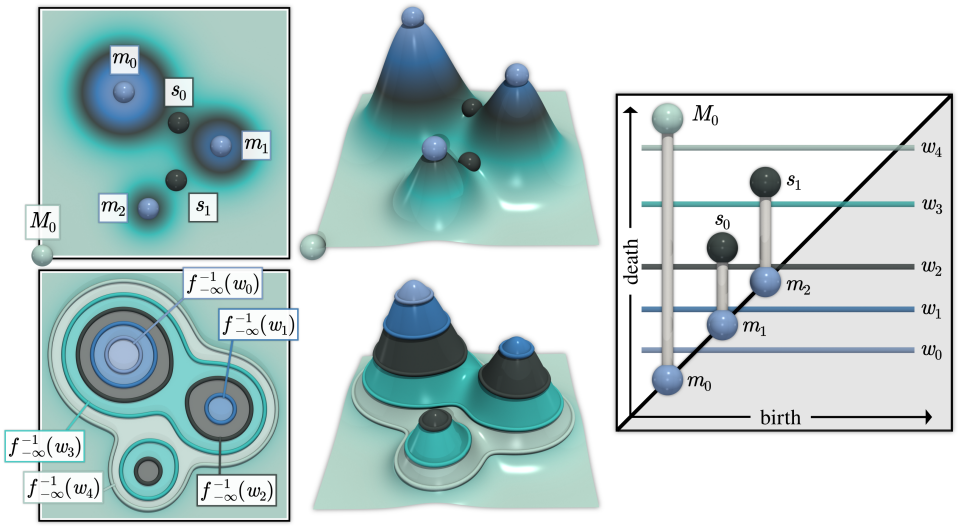


Figure 2.5 – *Given a PL scalar field f defined on a 2D PL manifold (top left), the minimum-saddle persistence diagram (right) tracks the evolution of the connected components of the sub-level sets of f (bottom). The color map of the PL scalar field (top left) indicates here the low values in blue and the high values in light green (i.e. the considered function f is the opposite of the elevation). Critical points of f are represented with spheres, in blue for minima, black for saddles and white for maxima. During the filtration, connected components appear at specific values corresponding to the minima m_0, m_1 and m_2. When the filtration reaches s_0 the connected component born at m_1 dies in favor of the older one born at m_0 creating the pair (m_1, s_0) in the diagram. Then, when reaching s_1, the component born at m_2 dies creating the pair (m_2, s_1). By convention the global minimum m_0 and the global maximum M_0 are paired together.*

meet at a critical point c', the *younger* component (created last, in c) *dies*, in favor of the *older* one (created first). The persistence diagram $\mathcal{D}(f)$ embeds each pair to a single point in 2D at coordinates $\big(f(c), f(c')\big)$ and we define the *persistence* of a pair by its height $f(c') - f(c)$.

It provides a visual overview of the features of a dataset (Figure 2.5), where salient features stand out from the diagonal while pairs corresponding to noise are located near the diagonal (Figure 2.7).

The diagonal in this 2D birth/death space plays an important role, it consists of all the features with zero persistence i.e. the *non-existent* structures of the data. A persistence diagram contains all the persistence pairs given a filtration and an infinite number of points in the diagonal.

Definition 2.37

(Persistence Diagram) The p^{th} *persistence diagram* of a filtration is a set of points in $\mathbb{R}^2$ (with possible multiplicity) encoding the birth and death of each homology class of the p^{th} persistent homology group, along with all the points in the diagonal $\{(x, y) \in \mathbb{R}^2 \mid x = y\}$ with infinite multiplicity.

During the filtration, a class γ that never dies is considered to have an

infinite persistence. The first class appearing in the filtration, at the global minimum, is one of them since it never merges with an older class. In practice, we crop the death value of these pairs to the maximum of the function. It results, at least, in a pair involving the global minimum and the global maximum of the function. It allows to clearly define a persistence pair corresponding to the most prominent structure of the domain and to use it in algorithms without having to deal with the infinite value.

The 0^{th} persistence diagram tracks the evolution of the connected components of the lower star filtration. It consists of the minimum-saddle persistence pairs, components being born at a minimum and dying at a saddle, along with the global min-max pair. These pairs are of particular interest since they represent structures of the data related to *pit* of values. On a d-manifold, the $(d-1)^{th}$ persistence diagram also encodes useful structures. Specifically, the duality argument [EH09, ELZ02] states that, under certain conditions, the $(d-1)^{th}$ persistence diagram coincide with the 0^{th} persistence diagram of the opposite filtration. Intuitively, it means that it tracks the evolution of the connected components of the upper star filtration (in opposition of the lower star filtration) corresponding to *peak* of values encoded by the saddle-maximum persistence pairs.

In some applications the structures with small persistence can be important, in order to inspect how the noise is organized. However, in the applications within this thesis we are more interested with structures having high persistence, corresponding to large peak or pit of values. In light of this, we often simplify the persistence diagrams to remove the pairs with small persistence (being not useful in our applications). This has the effect to facilitate the use of these objects by decreasing their size and hence the computation time of analysis tools using them (e.g. distances).

2.4.1 Metric between Persistence Diagrams

In the following, we introduce an established metric between persistence diagrams, the Wasserstein distance. After introducing this metric, we detail further derived concepts, such as the notions of geodesics and Wasserstein barycenters.

In practice, persistence diagrams are represented only by its persistence pairs, without the points in the diagonal with infinite multiplicity. To measure the distance between two diagrams $\mathcal{D}(f_i)$ and $\mathcal{D}(f_j)$, a typical pre-processing step consists in augmenting each diagram [KMN17], with

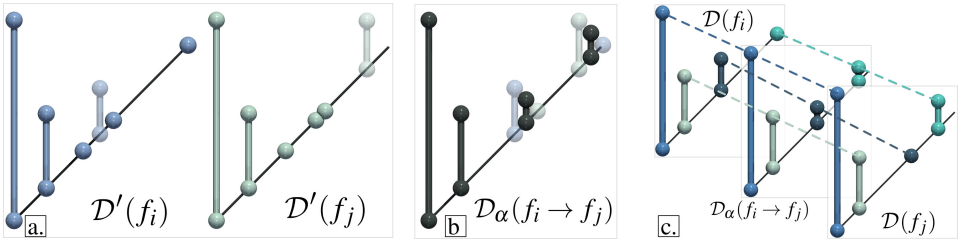


Figure 2.6 – *Distance and geodesic computation between two persistence diagrams $\mathcal{D}(f_i)$ and $\mathcal{D}(f_j)$, given the metric $W_2^{\mathcal{D}}$. The matching induced by the optimal partial assignment ϕ is shown with dashed lines (c). The pairs matched to the diagonal are shown in transparent (a). For two diagrams, the Wasserstein barycenter (b) is given, thanks to the d_2 distance in the birth/death space, by the arithmetic mean of the matched points. Then, the linear interpolation of the matchings (c) describes a geodesic [TMMH14].*

the diagonal projection of the off-diagonal points of the other diagram (Figure 2.6a):

$$\mathcal{D}'(f_i) = \mathcal{D}(f_i) \cup \{\Delta(p_j) \mid p_j \in \mathcal{D}(f_j)\}$$
$$\mathcal{D}'(f_j) = \mathcal{D}(f_j) \cup \{\Delta(p_i) \mid p_i \in \mathcal{D}(f_i)\},$$

where $\Delta(p_i) = (\frac{x_i+y_i}{2}, \frac{x_i+y_i}{2})$ stands for the diagonal projection of the off-diagonal point $p_i = (x_i, y_i) \in \mathcal{D}(f_i)$. Intuitively, this augmentation phase inserts dummy features in the diagram (with zero persistence, along the diagonal), hence preserving the topological information of the diagrams. This augmentation guarantees that the two diagrams now have the same number of points ($|\mathcal{D}'(f_i)| = |\mathcal{D}'(f_j)|$), which facilitates the evaluation of their distance, as described next.

Given two points $p_i = (x_i, y_i) \in \mathcal{D}'(f_i)$ and $p_j = (x_j, y_j) \in \mathcal{D}'(f_j)$, the ground distance d_q ($q > 0$) in the 2D birth/death space is given by:

$$d_q(p_i, p_j) = (|x_j - x_i|^q + |y_j - y_i|^q)^{1/q} = \|p_i - p_j\|_q. \tag{2.2}$$

By convention, $d_q(p_i, p_j)$ is set to zero between diagonal points ($x_i = y_i$ and $x_j = y_j$). Then, the L^q-Wasserstein distance $W_q^{\mathcal{D}}$ is:

$$W_q^{\mathcal{D}}(\mathcal{D}'(f_i), \mathcal{D}'(f_j)) = \min_{\phi \in \Phi} \left(\sum_{p_i \in \mathcal{D}'(f_i)} d_q(p_i, \phi(p_i))^q \right)^{1/q}, \tag{2.3}$$

where Φ is the set of all bijective assignments ϕ mapping a point $p_i \in \mathcal{D}'(f_i)$ to a point $p_j \in \mathcal{D}'(f_j)$ (possibly its diagonal projection, indicating the destruction of the corresponding feature, transparent pairs in Figure 2.6a).

Intuitively, the Wasserstein metric optimizes a matching between the two diagrams, and evaluates their distance given the resulting mismatch. The augmentation (Figure 2.6a) preserves the distance, while making the assignment problem balanced, and thus easily solvable with traditional algorithms [Mun57, Ber81].

2.4.2 Geodesic and Barycenter of Persistence Diagrams

Given a set $\mathcal{S}_\mathcal{D} = \{\mathcal{D}(f_1), \ldots, \mathcal{D}(f_N)\}$ of persistence diagrams, let $F(\mathcal{D}, \alpha)$ be the Fréchet energy of the set, under the metric $W_2^\mathcal{D}$, with the coefficients $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_N\}$, such that $\alpha_i \in [0,1]$ and $\sum_i \alpha_i = 1$:

$$F(\mathcal{D}, \alpha) = \sum_{\mathcal{D}(f_i) \in \mathcal{S}_\mathcal{D}} \alpha_i W_2^\mathcal{D}\big(\mathcal{D}, \mathcal{D}(f_i)\big)^2. \tag{2.4}$$

Then a diagram $\mathcal{D}^* \in \mathbb{D}$ (where $\mathbb{D}$ is the space of persistence diagrams) which minimizes $F(\mathcal{D}, \alpha)$ is called a *Wasserstein barycenter* of the set $\mathcal{S}_\mathcal{D}$ (or its Fréchet mean under the metric $W_2^\mathcal{D}$).

$\mathcal{D}^*$ can be efficiently estimated with an optimization strategy [TMMH14, VBT20] which resembles a Lloyd relaxation [Llo82], and which alternates an assignment and an update procedure. First, $\mathcal{D}$ is initialized at an arbitrary diagram of $\mathcal{S}_\mathcal{D}$. Then, the assignment step computes an optimal assignment $\phi_i : \mathcal{D} \to \mathcal{D}(f_i)$ between $\mathcal{D}$ and each diagram of $\mathcal{S}_\mathcal{D}$. Next, the update step updates the candidate $\mathcal{D}$ to a position in $\mathbb{D}$ which minimizes the sum of its squared distances to the diagrams of $\mathcal{S}_\mathcal{D}$ under the current set of assignments $\{\phi_1, \ldots, \phi_N\}$. This is achieved by replacing each point $p_i \in \mathcal{D}$ by the arithmetic mean (in the birth/death space) of all its assignments $\phi_i(p_i)$. The overall sequence assignment/update is iterated until convergence of the Fréchet energy.

In practice, the coefficients α_i are all set to the same value ($\alpha_i = 1/N$, $\forall i \in \{1, \ldots, N\}$). When $N = 2$ and $\alpha_1 = \alpha_2 = 0.5$ (Figure 2.6b), $\mathcal{D}^*$ becomes a midpoint between $\mathcal{D}(f_i)$ and $\mathcal{D}(f_j)$ and the set of possible values for α_1 and α_2 (Figure 2.6c) describes a geodesic in $\mathbb{D}$ (i.e. length minimizing path) with regard to the L^2-Wasserstein metric [TMMH14].

2.5 Merge Trees

We now introduce the main topological data representation studied in this thesis: the *merge tree*. We also describe a specific representation of the

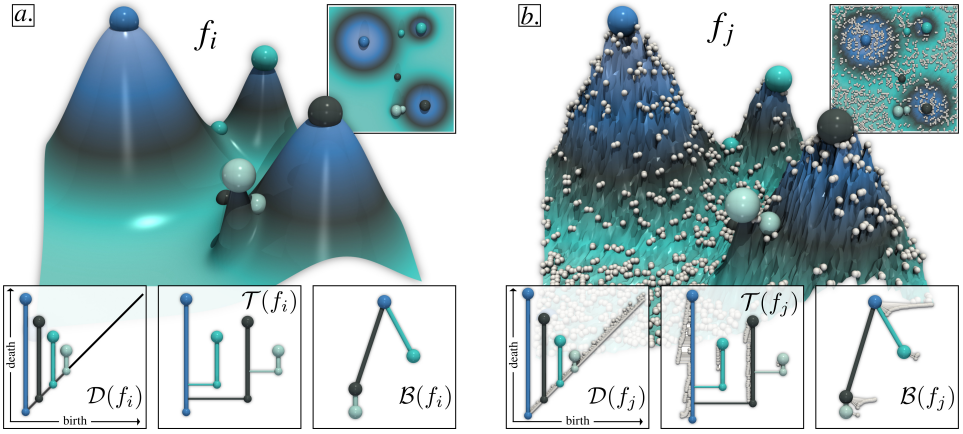


Figure 2.7 – *Illustration of the topological descriptors considered in this work, on a clean* (a) *and a noisy* (b) *variant of a 2D toy dataset. For all descriptors, the color code indicates the persistence of the corresponding saddle-maximum pair. Critical points are represented with spheres (larger ones for maxima). Persistence diagrams, merge trees and branch decomposition trees (BDTs) are respectively represented in the left, center and right insets. For both datasets, the four main features (the larger hills) are represented with salient pairs in the diagram and the merge tree. To avoid clutter in the visualization, the branches with low persistence (less than* 10% *of the function range) are rendered with small white arcs while larger, and colored arcs represent persistent branches (more than* 10% *of the function range).*

merge tree called the *branch decomposition tree*, which can be interpreted as a generalization of the extremum persistence diagram.

The *join* tree, noted $\mathcal{T}^-(f_i)$, is a visual summary of the connected components of $f_{i_{-\infty}}^{-1}(w)$ [CSA00]. The *split* tree (Figure 2.7), noted $\mathcal{T}^+(f_i)$, is defined symmetrically and describes the connected components of the super-level set $f_{i_{+\infty}}^{-1}(w)$. Each of these two *directed* trees is called a *merge* tree, noted generically $\mathcal{T}(f_i)$ in the following. Intuitively, these trees track the creation of connected components of the sub (or super) level sets at their leaves, and merge events at their interior nodes.

Definition 2.38

(Merge Tree) Let a PL scalar field $f : \mathcal{M} \to \mathbb{R}$ defined on a PL manifold $\mathcal{M}$. The *join tree* $\mathcal{T}^-(f)$ (respectively the *split tree* $\mathcal{T}^+(f)$) is defined as the quotient space $\mathcal{M}/\sim$ by the equivalence relation $\sim$ stating that two points p_1 and p_2 in $\mathcal{M}$ are equivalent if $f(p_1) = f(p_2)$ and if p_1 and p_2 belong to the same connected component of $f_{-\infty}^{-1}(f(p_1))$ (respectively $f_{+\infty}^{-1}(f(p_1))$).

To mitigate a phenomenon called *saddle swap*, these trees are often post-processed [SMKN20], by merging adjacent saddles in the tree if their relative difference in scalar value is smaller than a threshold $\epsilon_1 \in [0, 1]$.

Merge trees are often visualized according to a persistence-driven

branch decomposition [PCMS04], to make the persistence pairs captured by the tree stand out. In this context, a *persistent branch* is a monotone path on the tree connecting the nodes corresponding to the creation and destruction (according to the Elder rule, Section 2.2.2) of a connected component of sub (or super) level set. Then, the branch decomposition provides a planar layout of the merge tree, where each persistent branch is represented as a vertical segment (center insets in Figure 2.7). Persistent branches can be visually distinguished using a manual persistence threshold of the planar layout algorithm. It allows to show the important branches (having persistence above this threshold) with colored spheres and segments (center insets in Figure 2.7) and the non-important branches (having persistence below the threshold) with small white spheres and segments (center insets in Figure 2.7). Important branches are displayed at the right of their parent branch in the tree hierarchy and non-important branches at the left and close to their parent. In each case, branches are visually connected to their parent branch with a horizontal segment.

The *branch decomposition tree* (BDT), noted $\mathcal{B}(f_i)$, is a directed tree whose nodes are the persistent branches captured by the branch decomposition and whose arcs denote adjacency relations between them in the MT. In Figure 2.7, the BDTs (right insets) can be interpreted as the dual of the branch decompositions (center insets, with matching colors): each vertical segment in the branch decomposition (center) corresponds to a node in the BDT (right) and each horizontal segment (center, denoting an adjacency relation between branches) corresponds to an arc in the BDT. The BDT can be interpreted as a generalization of the extremum persistence diagram: like $\mathcal{D}(f_i)$, $\mathcal{B}(f_i)$ describes the population of (extremum) persistence pairs present in the data. However, unlike the persistence diagram, it additionally captures adjacency relations between them (Figure 2.7). Note that, the birth and death of each persistent branch $b_i \in \mathcal{B}(f_i)$, noted (x_i, y_i), span by construction an interval included in that of its parent $b'_i \in \mathcal{B}(f_i)$: $[x_i, y_i] \subseteq [x'_i, y'_i]$. This *nesting property* of BDTs is a direct consequence of the Elder rule (Section 2.2.2).

2.5.1 Construction

[CSA00] provides the reference algorithm to compute the merge tree in all dimensions. They used a Union-Find structure to track the connected components of the sub-level sets (for the join tree, or super-level sets for the split tree). For the join tree, vertices of the input data are processed

by increasing scalar value order. When a vertex is processed, if it is not connected to vertices of lower scalar values it is therefore considered as a minimum critical point and a node is created in the join tree and is considered as a Union-Find class. If the processed vertex is connected to vertices of lower scalar values but in the same Union-Find class it is therefore a regular point. If the vertices are in different Union-Find classes then it is a saddle critical point and a node is created in the join tree with edges linking this new node and the nodes corresponding to the collected Union-Find classes. These Union-Find classes are then merged into a single one. The process continue until having processed all the vertices. [GFJT17] presents an efficient algorithm for the construction of the merge tree in parallel.

2.5.2 Metric between Merge Trees

Several metrics have been introduced for measuring distances between merge trees. We focus in the remainder on the edit distance introduced by Sridharamurthy et al. [SMKN20], as its balance between computability and acceptable stability in practice seems particularly promising.

The edit distance between two merge trees $\mathcal{T}(f_i)$ and $\mathcal{T}(f_j)$, noted $D_E\big(\mathcal{T}(f_i), \mathcal{T}(f_j)\big)$, is defined as follows. Let N_i be a subset of the nodes of $\mathcal{T}(f_i)$ and $\overline{N_i}$ its complement. Let ϕ''' be a *partial* assignment between N_i and a subset N_j of the nodes of $\mathcal{T}(f_j)$ (with complement $\overline{N_j}$). Then $D_E\big(\mathcal{T}(f_i), \mathcal{T}(f_j)\big)$ is given by:

$$
\begin{aligned}
D_E\big(\mathcal{T}(f_i), \mathcal{T}(f_j)\big) = \min_{(\phi''', \overline{N_i}, \overline{N_j}) \in \Phi'''} \Big(& \sum_{n_i \in N_i} \gamma\big(n_i \to \phi'''(n_i)\big) && (2.5)\\
+ & \sum_{n_i \in \overline{N_i}} \gamma(n_i \to \emptyset) && (2.6)\\
+ & \sum_{n_j \in \overline{N_j}} \gamma(\emptyset \to n_j) \Big) && (2.7)
\end{aligned}
$$

where Φ''' is the space of *constrained* partial assignments (i.e. ϕ''' maps disjoint subtrees of $\mathcal{T}(f_i)$ to disjoint subtrees of $\mathcal{T}(f_j)$) and where γ refers to the cost for: *(i)* mapping a node $n_i \in \mathcal{T}(f_i)$ to a node $\phi''(n_i) = n_j \in \mathcal{T}(f_j)$ (line 2.5), *(ii)* deleting a node $n_i \in \mathcal{T}(f_i)$ (line 2.6) and *(iii)* creating a node $n_j \in \mathcal{T}(f_j)$ (line 2.7), $\emptyset$ being the empty tree.

Zhang [Zha96] introduced a polynomial time algorithm for computing a constrained sequence of edit operations with minimal edit distance (Equation 2.5), and showed that the resulting distance is indeed a metric if each cost γ for the above three edit operations is itself a metric (non-negativity, identity, symmetry, triangle inequality). Sridharamurthy et al.

[SMKN20] exploited this property to introduce their metric, by defining the following distance-based cost model, where p_i and p_j stand for the persistence pairs *containing* the nodes $n_i \in \mathcal{T}(f_i)$ and $n_j \in \mathcal{T}(f_j)$:

$$\begin{aligned}
\gamma(n_i \to n_j) &= \min\left(d_\infty(p_i, p_j), \gamma(n_i \to \emptyset) + \gamma(\emptyset \to n_j)\right) \\
\gamma(n_i \to \emptyset) &= d_\infty\left(p_i, \Delta(p_i)\right) \\
\gamma(\emptyset \to n_j) &= d_\infty\left(\Delta(p_j), p_j\right).
\end{aligned}$$

2.6 Other Topological Abstractions

Other topological abstractions could be used to extract more information and to be more discrimant than the persistence diagrams and the merge trees. Even if these representations are not used in our work, we would like to show the other main tools available in topological data analysis. We briefly introduce the notions of contour trees, Reeb graphs and Morse-Smale complexes and illustrate an example for each of them in Figure 2.8.

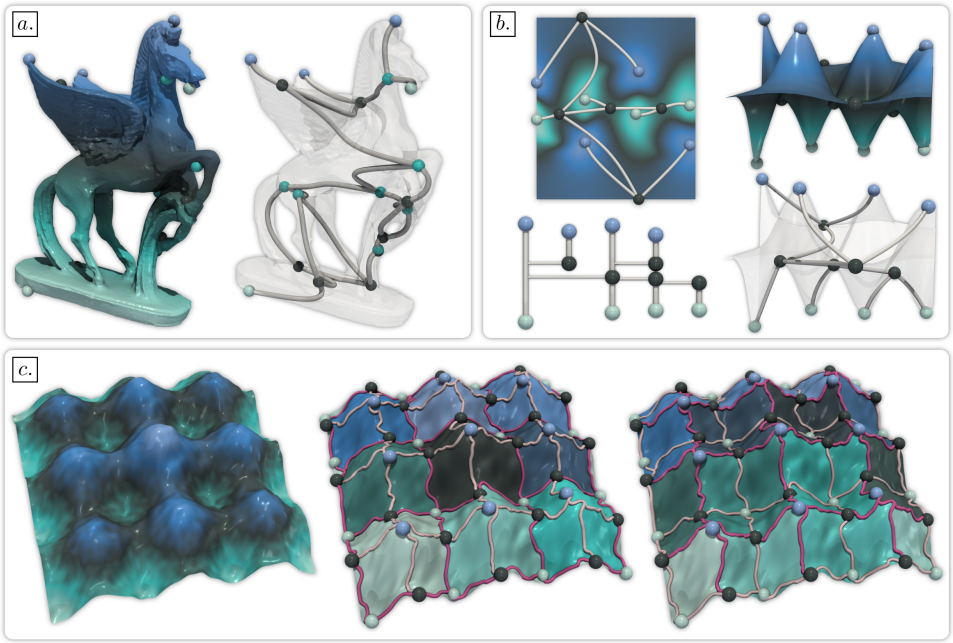


Figure 2.8 – *Examples of other topological abstractions. The Reeb graph (a) tracks the evolution of the connected components of the level set of the function during the sweep of the isovalues from $-\infty$ to ∞. When the graph contains no cycles it is referred to as the contour tree (b). The Morse-Smale complex (c) extracts areas of the domain where the integration of the gradient of the function leads to the same critical point.*

Reeb Graphs and Contour Trees

Reeb graphs (Figure 2.8a) track the evolution of the connected components of the level set. As the isovalue of the level set increases, each contour (the different connected components of the level set) is reduced to a point in the Reeb graph. The Reeb graph intuitively extracts the skeleton of complicated shape given a function. When the domain is simply connected the Reeb graph of a function is acyclic and corresponds to the contour tree (Figure 2.8b), in that case, the join tree and the split tree of a function can be merged together to create its contour tree [CSA00].

More-Smale Complexes

The Morse-Smale complex is another topological abstraction capturing a different aspect of the function compared to the other abstractions. It segments the domain given the gradient of the function, by grouping points together if following the gradient from these points lead to the same critical point. It results in two kinds of objects: the ascending manifolds (Figure 2.8c, middle) and the descending manifolds (Figure 2.8c, right), respectively by using the gradient and, respectively, its opposite.

3 Wasserstein Distances, Geodesics and Barycenters of Merge Trees

Contents

This chapter presents a unified computational framework for the estimation of distances, geodesics and barycenters of merge trees. We extend recent work on the edit distance [SMKN20] and introduce a new metric, called the *Wasserstein* distance between merge trees, which is purposely designed to enable efficient computations of geodesics and barycenters. Specifically, our new distance is strictly equivalent to the L^2-Wasserstein distance between extremum persistence diagrams, but it is restricted to a smaller solution space, namely, the space of rooted partial isomorphisms between branch decomposition trees. This enables a simple extension of existing optimization frameworks [TMMH14] for geodesics and barycenters from persistence diagrams to merge trees. We introduce a task-based algorithm which can be generically applied to distance, geodesic, barycenter or cluster computation. The task-based nature of our approach enables further accelerations with shared-memory parallelism. Extensive experiments on public ensembles and SciVis contest benchmarks demonstrate the efficiency of our approach – with barycenter computations in the orders of minutes for the largest examples – as well as its qualitative ability to generate representative barycenter merge trees, visually summarizing the features of interest found in the ensemble. We show the utility of our contributions with dedicated visualization applications: feature tracking, temporal reduction and ensemble clustering. We provide a lightweight C++ implementation that can be used to reproduce our results.

The work presented in this chapter has been published in the journal IEEE Transactions on Visualization and Computer Graphics as part of the proceedings of the IEEE VIS 2021 conference [PVDT22]. It was certified replicable by the Graphics Replicability Stamp Initiative (http://www.replicabilitystamp.org/). Our implementation is available at https://github.com/MatPont/WassersteinMergeTrees and the data used in this work at https://github.com/MatPont/WassersteinMergeTreesData. It is also integrated in the Topology ToolKit [TFL+17].

Our Contributions in one Image

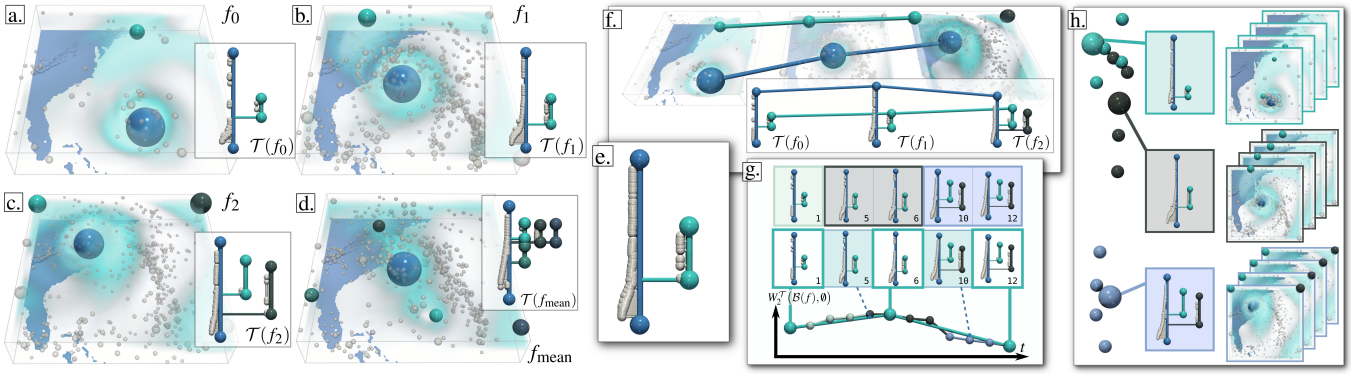


Figure 3.1 – *The merge trees of three members (a-c) of the Isabel ensemble (wind velocity) concisely and visually encode the number and salience of the features of interest found in the data (eyewall and region of high speed wind, blue and cyan). They also describe how these features are globally connected in the data. In these trees, branches with a low persistence (less than* 20% *of the function range) are shown with small white arcs. The pointwise mean for the three members (d) exhibits 5 salient maxima (due to distinct eyewall locations, blue, cyan and black) and its merge tree is not representative of the input trees (containing at most 3 large features). In contrast, the Wasserstein barycenter (e) is representative of the input trees, with a number and persistence of large branches that better match the input trees (a-c). Our framework for distances, geodesics and barycenters enables a variety of merge tree based applications, including (f) feature tracking, (g) temporal reduction –* key frames *are automatically identified (white insets) and deleted merge trees (blue insets) are accurately reconstructed with geodesics – and (h) ensemble clustering and summarization – the clusters and centroids automatically computed by our approach provide a visual summary of the main trends of features found in the ensemble.*

3.1 Context

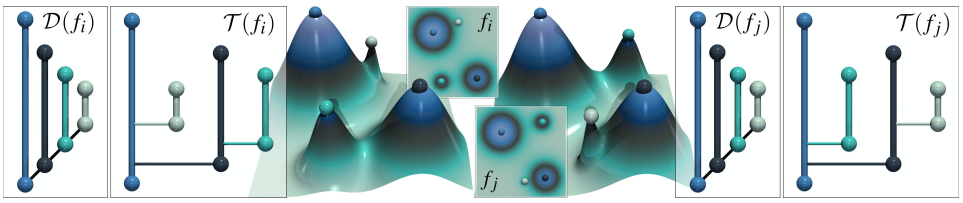


Figure 3.2 – *The persistence diagram, $\mathcal{D}(f_i)$, and the merge tree, $\mathcal{T}(f_i)$, both visually summarize the number, data range and salience of the features of interest present in the data. However, the persistence diagram represents each individual feature independently, while the merge tree additionally describes how they connect together. This results in a lack of specificity for the persistence diagram which can yield identical data representations for significantly distinct datasets (from left to right, the gaussians with white and cyan spheres have been swapped). In contrast, the merge tree captures this nuance and produces two distinct data representations.*

When dealing with an ensemble of datasets, a topological data abstraction such as the persistence diagram can be computed for each ensemble member (possibly in-situ [BAA+16, ABG+15]). Then, a major challenge for end users is the interpretation of the resulting ensemble of persistence diagrams. To address this, several analysis tools were developed for persistence diagrams ([TMMH14, LCO18, VBT20]) such as: distances (to compare persistence diagrams), geodesics (to visualize optimum transitions between them), and barycenters (to visualize one persistence diagram *representative* of a set). However, persistence diagrams suffer from a lack of specificity (Figure 3.2), which can prevent the identification of distinct feature trends within the ensemble.

This chapter addresses this problem by introducing a unified computational framework for the automatic computation of distances, geodesics, barycenters and clusters of merge trees. In particular, we extend recent work on the edit distance [SMKN20] and introduce a new metric, called the *Wasserstein* distance between merge trees, which is purposely designed to enable efficient computations of geodesics (i.e. length minimizing morphings) and barycenters. In that regard, our work can be interpreted as an extension of previous work on the edit distance [SMKN20], to adapt it to the optimization strategy previously developed for the computation of barycenters of persistence diagrams [TMMH14]. We present efficient, task-based algorithms using shared-memory parallelism, resulting in the computation of distances, geodesics and barycenters in practical times for real-life datasets. We illustrate the utility of each of our contributions in dedicated visualization tasks. First, we show that our distance com-

putation algorithm can be used for a merge-tree based tracking of features through time. Second, we show that our framework for computing geodesics between merge trees can be used for the reliable sub-sampling of temporal sequences of merge trees. Third, we illustrate the utility of our barycenters for clustering ensemble members based on their merge trees, while providing cluster centroids which visually summarize the main features of interest present in each cluster.

3.1.1 Related Work

The literature related to this chapter can be classified into three main groups, reviewed in the following: *(i)* uncertainty visualization, *(ii)* ensemble visualization, and *(iii)* topological methods for ensembles.

Uncertainty Visualization

Variability in data can be modeled and encoded in several ways. In particular, *uncertain* datasets capture variability by modeling each point of the domain as a random variable, whose variability is explicitly modeled by an estimator of an a priori probability density function (PDF). The analysis of uncertain data is a notoriously challenging problem in visualization, described in several surveys [PRJ12, BHJ+14, unc08, JS03, MAH+05, PWL97]. Early techniques focused on estimating the entropy of the random variables [PGA13], their correlations [PW12] or their gradient variations [PMW13]. The positional uncertainty of level sets has been studied for several interpolation schemes and PDF models [AE13, ASE16, PRW11, PWH11, PH11, PH13, PPH13, SKS12, AJ19]. Similarly, the positional uncertainty of critical points has been studied for Gaussian [LS16, OGHT10, OGT11, PPH12] or uniform distributions [GST14, BJB+12, Szy13]. A general limitation of existing methods for uncertain data is their dependence on the specific PDF model for which they have been designed. This reduces their usability for ensemble data, where the PDF estimated from the ensemble members can follow an arbitrary, unknown model. Also, most existing techniques for uncertain data do not consider multi-modal PDF models, which is however necessary when several, distinct trends are present in the ensemble data.

Ensemble Visualization

Another way to model and encode variability in data consists in considering ensemble datasets. In this setting, the variability is directly encoded

by an ensemble of empirical observations (i.e. the *members* of the ensemble). Current approaches to ensemble visualization typically compute some geometrical objects describing the features of interest (level sets, streamlines, etc), for each member of the ensemble. Then, an aggregation phase estimates a *representative* object for the resulting ensemble of geometrical objects. For instance, spaghetti plots [DHLZ02] are a typical example for studying level-set variability, especially for weather data [PWB+09, SZD+10]. More specifically, box-plots [WMK13] describe the variability of contours and curves [MWK14]. For flow ensembles, Hummel et al. [HOGJ13] introduce a Lagrangian framework for classification purposes. Clustering techniques have been investigated, to identify the main trends, and their variability, in ensembles of streamlines [FBW16] and isocontours [FKRW16]. However, only few approaches have applied this overall aggregation strategy to topological objects. Favelier et al. [FFST18] and Athawale et al. [AMJ+19] introduced approaches for analyzing the variability of critical points and gradient separatrices respectively. Several techniques attempted to generate an aggregated contour tree from an ensemble based on overlap-driven heuristics [WZ13, Kra10]. Recently, Lohfink et al. [LWL+20] introduced an approach for the consistent layout of multiple contour trees, to support effective visual comparisons between the contour trees of the distinct members of an ensemble. Although the above techniques addressed the visualization of ensembles of topological objects, they did not focus explicitly on the computation of a *representative* of multiple topological objects, such as barycenters.

Topological Methods

Concepts and algorithms from computational topology [EH09] have been investigated, adapted and extended by the visualization community for more than twenty years [HLH+16, YMS+21]. Popular topological representations include the persistence diagram [ELZ02, EH09] (Section 2.4), which represents the population of features of interest in function of their salience, and which can be computed via matrix reduction [EH09, BKR14]. The Reeb graph [BGSF08], which describes the connectivity evolution of level sets, has also been widely studied and several efficient algorithms have been documented [PSBM07, TGSP09, Par12, DN13], including parallel algorithms [GFJT19b]. Efficient algorithms have also been documented for its variants, the merge and contour trees [TV98, CSA00] (Section 2.5), and parallel algorithms have also been

described [MDN12, AN15, CWSA16, GFJT19a]. The Morse-Smale complex [EHZ01, EHNP03, BEHP03], which depicts the global behaviour of integral lines, is another popular topological data abstraction in visualization [DFFIM15]. Robust and efficient algorithms have been introduced for its computation [RWS11, SN12, GBP19] based on Discrete Morse Theory [For98].

Distance metrics, which are necessary ingredients for the computation of barycenters, have been studied for most of the above objects. Inspired by the literature in optimal transport [Kan42, Mon81], the Wasserstein distance between persistence diagrams [EH09] (Section 2.4.1) and its variant the Bottleneck distance [ELZ02] have been extensively studied. They are based on a bipartite assignment problem, for which exact [Mun57] and approximate [Ber81, KMN17] implementations are publicly available [TFL+17]. Several similarity measures have been introduced for Reeb graphs [HSKK01] and their variants [SSW14]. However, since these measures are not distance metrics (the preservation of the triangle inequality is not specifically enforced), they do not seem conducive to barycenter computation. Stable distance metrics between Reeb graphs [BGW14] and merge trees [MBW14] have been studied from a theoretical point of view but their computation, following an exponential time complexity, is not tractable for practical datasets in general, except if reliable correspondence labels between the nodes of the trees are provided on the input [GMO+19], which is not practical either for large ensembles. Distances with polynomial time computation algorithms have also been investigated. Similarly to our overall strategy, Beketayev et al. [BYM+14] focus on a dual representation, the *branch decomposition tree* (BDT, Section 2.5), but in contrast to our approach, they estimate their distances by iteratively reducing a target mismatch term, in particular, over a search space significantly larger than ours. Sridharamurthy et al. [SMKN20] specialize efficient algorithms for computing constrained edit distances between trees [Zha96] to the special case of merge trees (see Section 2.5.2), resulting in a distance which is computable for real-life datasets and with acceptable practical stability. However, it is not conducive to simple barycenter computations.

Indeed, the linear interpolation of the optimal node assignments induced by this metric (Figure 3.3) does not result in a shortest path, and hence generates inaccurate midpoints (i.e. inaccurate barycenters given two trees). This further implies that there is no clear or simple strategy for the general computation of barycenters according to that metric.

A key technical reason for this is that D_E involves assignments be-

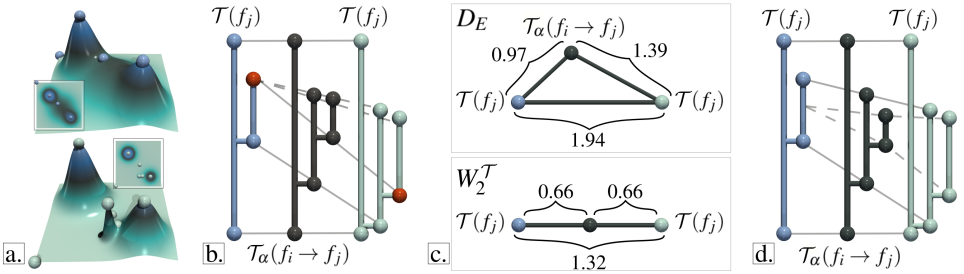


Figure 3.3 – *In this example with two datasets f_i and f_j (a), the optimal matchings (gray) with regard to the edit distance D_E [SMKN20] (b) map a maximum to a saddle (red spheres). The resulting linear interpolation $\mathcal{T}_\alpha(f_i \to f_j)$ (b) does not describe a shortest path between the input trees (c, top): $D_E\big(\mathcal{T}(f_i), \mathcal{T}_\alpha(f_i \to f_j)\big) + D_E\big(\mathcal{T}_\alpha(f_i \to f_j), \mathcal{T}(f_j)\big) > D_E\big(\mathcal{T}(f_i), \mathcal{T}(f_j)\big)$. In contrast, our new metric $W_2^{\mathcal{T}}$ enables linear interpolations (d) which exactly coincide with shortest paths (c, bottom). The numbers included in (c) are the actual values for D_E (top) and $W_2^{\mathcal{T}}$ (bottom) for this example.*

tween *nodes* (of the input merge trees) and not *persistence pairs*. This has several consequences. First, given two input trees $\mathcal{T}(f_i)$ and $\mathcal{T}(f_j)$, D_E's matchings may assign a saddle node in $\mathcal{T}(f_i)$ to an extremum node in $\mathcal{T}(f_j)$, resulting in inconsistent interpolations in the data (from a valley to a peak). Second, D_E's matchings can possibly assign two nodes in $\mathcal{T}(f_i)$ belonging to a *single* persistence pair of f_i to nodes in $\mathcal{T}(f_j)$ belonging to *distinct* persistence pairs in f_j. This second phenomenon further challenges interpolation-based geodesics.

Regarding the estimation of a *representative* object from a set of topological representations, several approaches emerged recently. A recent line of work [GMO+19, YWM+19b] introduced a framework for computing a *1-center* of a set of merge trees (i.e. minimizing its maximum distance to the set), according to an interleaving distance. However, as documented by its authors, this approach requires pre-existing, reliable correspondence labels between the nodes of all the input trees, which is not practical with real-life datasets (heuristics need to be considered). Also, since they minimize their *maximum* distance to a set, 1-centers are typically sensitive to outliers, which prevents their usage for estimating trends or supporting clustering tasks (which typically focus on densities rather than maximum distances). This is further evaluated in Section 3.7.2. In contrast, our approach focuses on the estimation of *barycenters* (instead of 1-centers) and computes a tree which minimizes its *average* distance to an ensemble of merge trees (instead of its maximum distance), which is less sensitive to outliers, which better captures trends and which supports clustering tasks. Moreover, the node correspondences between the barycenter and the input trees are automatically estimated by our approach via an assignment opti-

mization present at the core of our distance estimation. Thus our method does not require input correspondences, which makes it readily applicable to real-life ensembles. Several methods [TMMH14, LCO18, VBT20] have been introduced for the automatic estimation of barycenters (Section 2.4.2) of persistence diagrams (or vectorized variants [ACE+17, Bub15]). However, the persistence diagram can lack specificity in its data characterization (Figure 3.2). This limitation is addressed by our work which focuses instead on merge trees.

3.1.2 Contributions

This chapter makes the following new contributions:

1. *A practical distance metric between merge trees:* We extend recent work on the edit distance [SMKN20] and introduce a new distance between merge trees, which, in contrast to previous work, is purposely designed to enable efficient computations of geodesics and barycenters. It can be computed efficiently, it has acceptable practical stability and it has a strong connection to established metrics, which eases its interpretation. Specifically, it can be interpreted as a variant of the L^2-Wasserstein distance for persistence diagrams, for which we constrain the underlying search space to account for the additional structural information provided by the merge tree.

2. *A simple approach for computing geodesics between merge trees:* Given our new metric, we present a simple approach for computing geodesics between merge trees. It uses a simple linear interpolation of the assignments resulting from our new metric, enabling the exact computation of geodesics in linear time. This follows from previous work on persistence diagram geodesics [TMMH14] and it is made possible thanks to a new, local normalization procedure, guaranteeing the topological consistency of the interpolated trees.

3. *An approach for computing barycenters between merge trees:* Our method for geodesics between merge trees enables a straightforward adaptation of previous optimization strategies for persistence diagram barycenters [TMMH14], resulting, to our knowledge, in the first approach for the computation of barycenters of merge trees.

4. *Unified computational framework:* We present a unified computational framework for the estimation of distances, geodesics, barycenters, and clusters of merge trees. In particular, we introduce an efficient,

task-based algorithm adapted from previous work on edit distances [Zha96, SMKN20], which is generically applicable to any of the above tasks. Our algorithm supports shared-memory parallelism, allowing for further accelerations in practice.

5. *Applications:* We illustrate the utility of each of our contributions with dedicated visualization tasks, including feature tracking, temporal reduction and ensemble clustering and summarization.

6. *Implementation:* We provide a lightweight C++ implementation of our algorithms that can be used for reproduction purposes.

3.2 Preliminaries

The necessary theoretical background for this chapter has already been introduced in Chapter 2. However, we would like here to redefine the Wasserstein distance between persistence diagrams to ease the transition to the distance between merge trees proposed in this chapter.

Let P_i be a subset of the off-diagonal points of $\mathcal{D}(f_i)$ and $\overline{P_i}$ its complement (i.e. the other off-diagonal points of $\mathcal{D}(f_i)$ not in P_i). Let $(\phi, \overline{P_i}, \overline{P_j})$ be a *partial assignment* between $\mathcal{D}(f_i)$ and $\mathcal{D}(f_j)$, i.e. a bijective map between P_i and a subset of off-diagonal points P_j of $\mathcal{D}(f_j)$, with complement $\overline{P_j}$. Then, the L^q-Wasserstein distance, noted $W_q^{\mathcal{D}}$, can be redefined as:

$$W_q^{\mathcal{D}}\big(\mathcal{D}(f_i), \mathcal{D}(f_j)\big) = \min_{(\phi, \overline{P_i}, \overline{P_j}) \in \Phi} \Big(\sum_{p_i \in P_i} d_q\big(p_i, \phi(p_i)\big)^q \tag{3.1}$$

$$+ \sum_{p_i \in \overline{P_i}} d_q\big(p_i, \Delta(p_i)\big)^q \tag{3.2}$$

$$+ \sum_{p_j \in \overline{P_j}} d_q\big(\Delta(p_j), p_j\big)^q \Big)^{1/q} \tag{3.3}$$

where Φ is the set of all possible partial assignments mapping each point $p_i \in \mathcal{D}(f_i)$ to a point $\phi(p_i) = p_j \in \mathcal{D}(f_j)$ (line 3.1), or to its diagonal projection, $\Delta(p_i) = (\frac{x_i+y_i}{2}, \frac{x_i+y_i}{2})$, denoting the removal of the corresponding feature from $\mathcal{D}(f_i)$ or $\mathcal{D}(f_j)$ (lines 3.2 and 3.3).

3.3 Wasserstein Distances Between Merge Trees

This section introduces our new distance metric between merge trees, which is specifically designed for the subsequent computation of geodesics (Section 3.4) and barycenters (Section 3.5). For this, we bridge

the gap between the edit distance between merge trees [SMKN20] and existing work addressing the computation of geodesics and barycenters for persistence diagrams according to the L^2-Wasserstein distance [TMMH14].

3.3.1 Overview

The end goal of our work is the computation of barycenters of merge trees. For this, we extend the edit distance D_E [SMKN20] (formalized in Section 2.5.2), to make it fit the optimization strategy used for barycenters of persistence diagrams [TMMH14]. Our key idea consists in transforming D_E such that it becomes strictly equivalent to the L^2-Wasserstein distance of persistence diagrams, but given a *restricted* space of possible assignments, constrained by the structure of the input trees $\mathcal{T}(f_i)$ and $\mathcal{T}(f_j)$, hence its name Wasserstein distance between merge trees. Then, thanks to this compatibility with the L^2-Wasserstein distance, the assignments resulting from our metric can be directly used for interpolation-based geodesic and barycenter computations (Section 3.4 and Section 3.5). Overall, our strategy involves four major modifications to the edit distance D_E [SMKN20], detailed in the remainder of this section:

1. To consider assignments between persistence pairs instead of merge tree nodes, we consider an edit distance between the BDTs $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$ (Section 2.5) instead of the input merges trees $\mathcal{T}(f_i)$ and $\mathcal{T}(f_j)$ (as done with D_E). This is described in Section 3.3.2.

2. We constrain the assignment search space to the space of *rooted partial isomorphisms*. Specifically, similarly to D_E, we enforce the assignment of disjoint subtrees of $\mathcal{B}(f_i)$ to disjoint subtrees of $\mathcal{B}(f_j)$. Moreover, in contrast to D_E, we additionally extend this constraint by enforcing the destruction of entire subtrees upon the destruction of their root. These two constraints together enforce assignments describing isomorphisms between rooted subtrees of $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$. Such isomorphisms pave the way for interpolation-based geodesics. This is described in Section 3.3.2, Section 3.3.3 and Section 3.4.1.

3. We introduce a cost model based on the Euclidean distance d_2 to enable geodesic computation by linear interpolation of the assignments in the 2D birth/death space. This is described in Secs. 3.3.2 and 3.4.1.

4. We finally extend our metric with a local normalization term, which enforces nested birth-death values, along the interpolation of the assignments, for nested branches. This is described in Section 3.4.2.

3.3.2 Definition and Properties

Given two input merge trees, $\mathcal{T}(f_i)$ and $\mathcal{T}(f_j)$, we first consider their BDTs $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$ (Section 2.5). Let B_i be a subset of the nodes of $\mathcal{B}(f_i)$ and $\overline{B_i}$ its complement. Note that each node in B_i corresponds to a persistence pair of $\mathcal{D}(f_i)$. Let $(\phi', \overline{B_i}, \overline{B_j})$ be a partial assignment between B_i and a subset B_j of the nodes of $\mathcal{B}(f_j)$ (with complement $\overline{B_j}$). Then we introduce the L^2-Wasserstein distance $W_2^{\mathcal{T}}$ between the BDTs $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$ of the merge trees $\mathcal{T}(f_i)$ and $\mathcal{T}(f_j)$ as:

$$W_2^{\mathcal{T}}\big(\mathcal{B}(f_i), \mathcal{B}(f_j)\big) = \min_{(\phi', \overline{B_i}, \overline{B_j}) \in \Phi'} \Big(\sum_{b_i \in B_i} \gamma\big(b_i \to \phi'(b_i)\big)^2 \tag{3.4}$$

$$+ \sum_{b_i \in \overline{B_i}} \gamma(b_i \to \emptyset)^2 \tag{3.5}$$

$$+ \sum_{b_j \in \overline{B_j}} \gamma(\emptyset \to b_j)^2 \Big)^{1/2} \tag{3.6}$$

where Φ' is the space of constrained partial assignments mapping disjoints subtrees of $\mathcal{B}(f_i)$ to disjoint subtrees of $\mathcal{B}(f_j)$, *and* mapping entire subtrees to the empty tree $\emptyset$ if their root is itself mapped to $\emptyset$. Then, given the k^{th} direct child of b_i, noted b_i^k, it follows that b_i^k either maps through ϕ' to a direct child of $\phi'(b_i) \in \mathcal{B}(f_j)$ (then $b_i, b_i^k \in B_i$) or to the empty tree $\emptyset$ (then the subtree rooted in b_i^k, noted $\mathcal{B}(f_i, b_i^k)$, belongs to $\overline{B_i}$). This further implies that the rooted subtrees $B_i \subseteq \mathcal{B}(f_i)$ and $B_j = \phi'(B_i) \subseteq \mathcal{B}(f_j)$ are isomorphic and we call $(\phi', \overline{B_i}, \overline{B_j})$ a *rooted partial isomorphism*. Unlike D_E (see Section 2.5.2) but similarly to $W_2^{\mathcal{D}}$ (Equation 3.1), the cost of each operation (mapping, line 3.4, destruction, line 3.5, and creation, line 3.6) is squared, and the square root of the sum of the squared costs is considered as the overall distance.

Next, we define the edit costs as follows (we recall that each branch $b_i \in \mathcal{B}(f_i)$ exactly coincides with a persistence pair $p_i \in \mathcal{D}(f_i)$):

$$\begin{aligned} \gamma\big(b_i \to \phi'(b_i)\big) &= d_2\big(b_i, \phi'(b_i)\big) \\ \gamma(b_i \to \emptyset) &= d_2\big(b_i, \Delta(b_i)\big) \\ \gamma(\emptyset \to b_j) &= d_2\big(\Delta(b_j), b_j\big). \end{aligned} \tag{3.7}$$

Note that the expression of the L^2-Wasserstein distance $W_2^{\mathcal{T}}$ between merge trees (Equation 3.4) is therefore identical to the expression of the

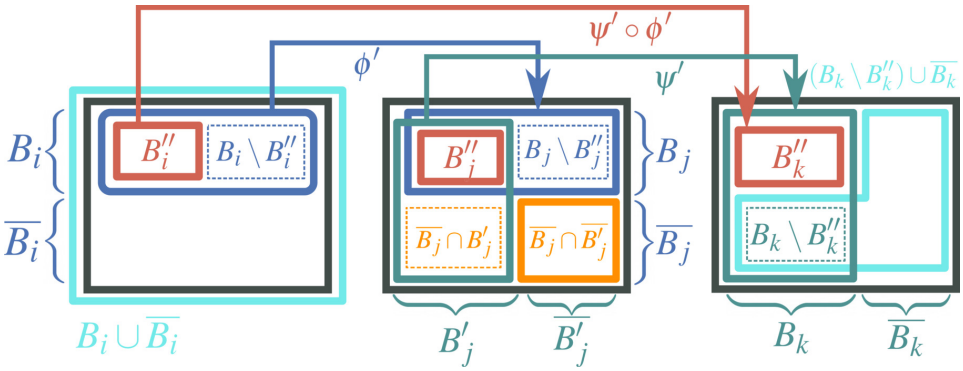


Figure 3.4 – *The composition $\psi' \circ \phi'$ of two rooted isomorphisms ϕ' (blue) and ψ' (green) is itself a rooted isomorphism (red). In this schematic view, the involved subtrees are represented as squares.*

Wasserstein distance between persistence diagrams (Equation 3.1) for $q = 2$, at the notable exception of the search space of the partial assignments $\Phi' \subset \Phi$, which is constrained to rooted partial isomorphisms. $W_2^{\mathcal{T}}$ is indeed a distance metric.

$W_2^{\mathcal{T}}\left(\mathcal{B}(f_i), \mathcal{B}(f_j)\right)$ is always non-negative (the costs γ are squared).

$W_2^{\mathcal{T}}\left(\mathcal{B}(f_i), \mathcal{B}(f_j)\right)$ is symmetric (destruction and creation costs are symmetric, lines 3.5 and 3.6).

$W_2^{\mathcal{T}}\left(\mathcal{B}(f_i), \mathcal{B}(f_j)\right) = 0$ if and only if all costs $\gamma = 0$, which only happens if $\mathcal{B}(f_i) = \mathcal{B}(f_j)$ (the identity is included in Φ').

We now argue that $W_2^{\mathcal{T}}$ preserves the triangle inequality, given three trees $\mathcal{B}(f_i)$, $\mathcal{B}(f_j)$ and $\mathcal{B}(f_k)$. For this, we follow a classical approach which we detail here for the sake of completeness. First, we argue that a composition of (optimal) partial rooted isomorphisms (from $\mathcal{B}(f_i)$ to $\mathcal{B}(f_j)$, then from $\mathcal{B}(f_j)$ to $\mathcal{B}(f_k)$) is itself a valid partial rooted isomorphism (and hence belong to our solution space Φ') and that its associated cost consequently bounds by above $W_2^{\mathcal{T}}\left(\mathcal{B}(f_i), \mathcal{B}(f_k)\right)$ (Equation 3.8). Second, we argue that this associated cost is itself bounded by above by $W_2^{\mathcal{T}}\left(\mathcal{B}(f_i), \mathcal{B}(f_j)\right) + W_2^{\mathcal{T}}\left(\mathcal{B}(f_j), \mathcal{B}(f_k)\right)$.

Let $(\phi', \overline{B_i}, \overline{B_j})$ be the optimal solution of the partial assignment problem between $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$. ϕ' is a *rooted* isomorphism (i.e. an isomorphism between rooted subtrees) between a subtree B_i of $\mathcal{B}(f_i)$ and a subtree B_j of $\mathcal{B}(f_j)$ (blue, Figure 3.4). Equivalently, ϕ' can also be interpreted as a bijection between the *arcs* of B_i and those of B_j.

Let $(\psi', \overline{B_j'}, \overline{B_k})$ be the optimal solution of the partial assignment problem between $\mathcal{B}(f_j)$ and $\mathcal{B}(f_k)$. ψ' is a rooted isomorphism between a subtree B_j' of $\mathcal{B}(f_j)$ and a subtree B_k of $\mathcal{B}(f_k)$ (green, Figure 3.4).

Let B_j'' be the set of nodes of $\mathcal{B}(f_j)$ involved in *both* ϕ' and ψ' ($B_j'' =$

$B_j \cap {B_j}'$, in red in Figure 3.4, center). Let ${B_i}''$ be their pre-image by ϕ' (${B_i}'' = {\phi'}^{-1}({B_j}'')$, in red in Figure 3.4, left) and ${B_k}''$ their image by ψ' (${B_k}'' = \psi'({B_j}'')$, in red in Figure 3.4, right).

Since both ϕ' and ψ' are rooted isomorphisms, their composition $\psi' \circ \phi'$ is also a (rooted) isomorphism between the subtrees ${B_i}''$ of $\mathcal{B}(f_i)$ and ${B_k}''$ of $\mathcal{B}(f_k)$ (equivalently, it is a bijection between the arcs of ${B_i}''$ and the arcs of ${B_k}''$). Then $(\psi' \circ \phi', \overline{{B_i}''}, \overline{{B_j}''})$ is itself a rooted partial isomorphism and belongs to Φ'.

Then, it follows that:

$$\begin{aligned} W_2^{\mathcal{T}}\big(\mathcal{B}(f_i), \mathcal{B}(f_k)\big) \leq \quad & \Big(\sum_{b \in {B_i}''} \gamma\big(b \to \psi' \circ \phi'(b)\big)^2 \\ & + \sum_{b \in \overline{{B_i}''}} \gamma(b \to \emptyset)^2 \\ & + \sum_{b \in \overline{{B_k}''}} \gamma(\emptyset \to b)^2 \Big)^{1/2}. \end{aligned} \tag{3.8}$$

Now, let U, V, W be scalar functions on the nodes of the set $B_{ik} = B_i \cup \overline{B_i} \cup (B_k \setminus B_k'') \cup \overline{B_k}$ (cyan subset, Figure 3.4) such that:

$$U(b) = \begin{cases} \gamma\big(b \to \psi' \circ \phi'(b)\big) & \text{for } b \in B_i'' = {\phi'}^{-1}(B_j'') \\ \gamma(b \to \emptyset) & \text{for } b \in B_i \setminus B_i'' \\ \gamma(b \to \emptyset) & \text{for } b \in \overline{B_i} \\ \gamma(\emptyset \to b) & \text{for } b \in \overline{B_k} \\ \gamma(\emptyset \to b) & \text{for } b \in B_k \setminus B_k'' = \psi'(\overline{B_j} \cap B_j'). \end{cases} \tag{3.9}$$

U describes all the possible individual costs involved in the composition $\psi' \circ \phi'$. In particular, we can re-write Equation 3.8 as:

$$W_2^{\mathcal{T}}\big(\mathcal{B}(f_i), \mathcal{B}(f_k)\big) \leq \quad \|U\|_2 \quad = \Big(\sum_{b \in B_{ik}} U(b)^2 \Big)^{1/2}. \tag{3.10}$$

$$V(b) = \begin{cases} \gamma\big(b \to \phi'(b)\big) & \text{for } b \in B_i'' = {\phi'}^{-1}(B_j'') \\ \gamma\big(b \to \phi'(b)\big) & \text{for } b \in B_i \setminus {B_i}'' \\ \gamma(b \to \emptyset) & \text{for } b \in \overline{B_i} \\ 0 & \text{for } b \in \overline{B_k} \\ \gamma\big({\psi'}^{-1}(b) \to \emptyset\big) & \text{for } b \in B_k \setminus B_k'' = \psi'(\overline{B_j} \cap B_j') \end{cases} \tag{3.11}$$

V describes a *subset* of the individual costs involved in the optimal rooted partial isomorphism ϕ'. In particular, only the costs involving $\overline{B_j} \cap \overline{B_j'}$ (orange square, Figure 3.4, middle) are excluded. Thus, we have:

$$W_2^{\mathcal{T}}\big(\mathcal{B}(f_i), \mathcal{B}(f_j)\big) \geq \quad \|V\|_2 \quad = \big(\sum_{b \in B_{ik}} V(b)^2\big)^{1/2}. \tag{3.12}$$

$$W(b) = \begin{cases} \gamma\big(\phi'(b) \to \psi' \circ \phi'(b)\big) & \text{for } b \in B_i'' = \phi'^{-1}(B_j'') \\ \gamma\big(\phi'(b) \to \emptyset\big) & \text{for } b \in B_i \setminus B_i'' \\ 0 & \text{for } b \in \overline{B_i} \\ \gamma(b \to \emptyset) & \text{for } b \in \overline{B_k} \\ \gamma\big(\psi'^{-1}(b) \to b\big) & \text{for } b \in B_k \setminus B_k'' = \psi'(\overline{B_j} \cap B_j') \end{cases} \tag{3.13}$$

Similarly to V, W describes a *subset* of the individual costs involved in the optimal rooted partial isomorphism ψ'. In particular, only the costs involving B_k'' (red square, Figure 3.4, right) are excluded. Thus:

$$W_2^{\mathcal{T}}\big(\mathcal{B}(f_j), \mathcal{B}(f_k)\big) \geq \quad \|W\|_2 \quad = \big(\sum_{b \in B_{ik}} W(b)^2\big)^{1/2}. \tag{3.14}$$

Now, since γ is defined by the Euclidean distance (Equation 3.7), we have for each node $b \in B_{ik}$:

$$0 \leq U(b) \leq V(b) + W(b).$$

This can be verified by comparing the i^{th} line of Equation 3.9 to the sum of the i^{th} lines of Equation 3.11 and Equation 3.13. Then, we have:

$$\|U\|_2 \leq \|V + W\|_2. \tag{3.15}$$

Now, since the L^2 norm between vectors respects itself the triangle inequality, we have the following inequality:

$$\|V + W\|_2 \leq \|V\|_2 + \|W\|_2. \tag{3.16}$$

Then, by combining equations 3.10, 3.15, 3.16, 3.12, and 3.14, it follows that:

$$\begin{aligned} W_2^{\mathcal{T}}\big(\mathcal{B}(f_i), \mathcal{B}(f_k)\big) \quad &\leq \quad \|U\|_2 \leq \|V + W\|_2 \leq \|V\|_2 + \|W\|_2 \\ &\leq W_2^{\mathcal{T}}\big(\mathcal{B}(f_i), \mathcal{B}(f_j)\big) + W_2^{\mathcal{T}}\big(\mathcal{B}(f_j), \mathcal{B}(f_k)\big) \end{aligned}$$

which concludes the proof.

Moreover, since $\Phi' \subset \Phi$, it follows that $W_2^{\mathcal{T}}\big(\mathcal{B}(f_i), \mathcal{B}(f_j)\big) \geq W_2^{\mathcal{D}}\big(\mathcal{D}(f_i), \mathcal{D}(f_j)\big)$, which was one of the main motivations of our work

(i.e. to exploit the merge tree to define a more discriminative metric, Figure 3.2). Similarly to Sridharamurthy et al. [SMKN20], we mitigate saddle swap instabilities in a preprocessing step, by merging adjacent saddles in the input trees if their difference in scalar value is smaller than a threshold $\epsilon_1 \in [0,1]$ (relative to the largest difference between adjacent saddles). Then, when $\epsilon_1 = 1$, it follows that $W_2^{\mathcal{T}}\left(\mathcal{B}(f_i), \mathcal{B}(f_j)\right) = W_2^{\mathcal{D}}\left(\mathcal{D}(f_i), \mathcal{D}(f_j)\right)$. This simple merging strategy significantly improves the practical stability of $W_2^{\mathcal{T}}$, as empirically studied in Section 3.7.2 (Figure 3.13).

3.3.3 Computation

This section describes our algorithm for the recursive exploration of the search space Φ' (Equation 3.4). It is based on the same recursive traversal as Zhang's algorithm [Zha96], which we simplify as our search space is significantly more constrained. Specifically our distance evaluation between subtrees (Equation 3.18) involves fewer solutions and it is restricted to subtrees rooted at identical depth only.

Given the subtree $\mathcal{B}(f_i, b)$ of $\mathcal{B}(f_i)$ (rooted in b) and b^k the k^{th} direct child of b in $\mathcal{B}(f_i, b)$, the distance between the subtree $\mathcal{B}(f_i, b)$ and the empty tree $\emptyset$ is then obtained recursively by:

$$W_2^{\mathcal{T}}\left(\mathcal{B}(f_i, b), \emptyset\right) = \left(\gamma(b \to \emptyset)^2 + \sum_k W_2^{\mathcal{T}}\left(\mathcal{B}(f_i, b^k), \emptyset\right)^2\right)^{1/2}. \tag{3.17}$$

The first step of our algorithm consists in evaluating $W_2^{\mathcal{T}}\left(\mathcal{B}(f_i, b), \emptyset\right)$ with Equation 3.17 for all branches $b \in \mathcal{B}(f_i)$ (and similarly for $\mathcal{B}(f_j)$).

Next, let $\mathcal{F}(f_i, b)$ be the *forest* of b in $\mathcal{B}(f_i)$: $\mathcal{F}(f_i, b)$ is the set of all the subtrees rooted at the k direct children of b: $\mathcal{F}(f_i, b) = \{\mathcal{B}(f_i, b^1), \mathcal{B}(f_i, b^2), \ldots, \mathcal{B}(f_i, b^k)\}$. Then the distance between two subtrees

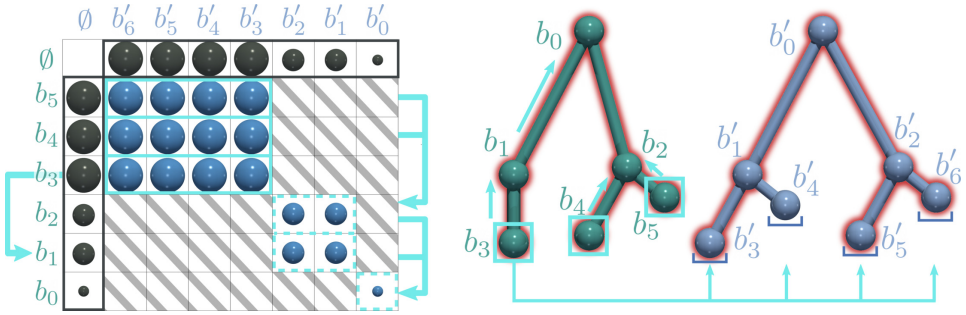


Figure 3.5 – *The exploration of the space Φ' (a candidate is highlighted in red, right) relies on the evaluation of the sparse matrix $\mathbb{T}$ of subtree distances (left). Our task-based algorithm optimizes the parallel computation of independent terms. Spheres of equal radius in $\mathbb{T}$ denote independent terms and arrows between the lines of $\mathbb{T}$ indicate task dependence (equivalently illustrated with arrows in the input BDTs, right).*

$\mathcal{B}(f_i, b_i)$ and $\mathcal{B}(f_j, b_j)$ is set to $+\infty$ when b_i and b_j have distinct depths (gray crosshatching lines in Figure 3.5, left). Otherwise (spheres in Figure 3.5, left), it is obtained recursively by:

$$\begin{aligned} W_2^{\mathcal{T}}\left(\mathcal{B}(f_i, b_i), \mathcal{B}(f_j, b_j)\right) &= \Big(\gamma(b_i \to b_j)^2 \\ &+ W_2^{\mathcal{T}}\left(\mathcal{F}(f_i, b_i), \mathcal{F}(f_j, b_j)\right)^2\Big)^{1/2}. \end{aligned} \tag{3.18}$$

Let F_i be a subset of the forest $\mathcal{F}(f_i, b_i)$ and $\overline{F_i}$ its complement. The distance between two forests is then given recursively by:

$$\begin{aligned} W_2^{\mathcal{T}}\left(\mathcal{F}(f_i, b_i), \mathcal{F}(f_j, b_j)\right) = \min_{(\phi'', \overline{F_i}, \overline{F_j}) \in \Phi''} \Big(& \sum_{f_i \in F_i} W_2^{\mathcal{T}}\left(f_i, \phi''(f_i)\right)^2 \\ + & \sum_{f_i \in \overline{F_i}} W_2^{\mathcal{T}}\left(f_i, \emptyset\right)^2 \\ + & \sum_{f_j \in \overline{F_j}} W_2^{\mathcal{T}}\left(f_j, \emptyset\right)^2\Big)^{1/2} \end{aligned}$$

where $(\phi'', \overline{F_i}, \overline{F_j})$ becomes the solution of a local, partial assignment problem between forests, mapping F_i to a subset $F_j \in \mathcal{F}(f_j, b_j)$ (with complement $\overline{F_j}$) or to the empty tree $\emptyset$, and which can be solved with traditional assignment algorithms [Mun57, Ber81] (see Section 2.4.1). Then, the overall distance $W_2^{\mathcal{T}}$ between the two input merge trees is obtained by estimating Equation 3.18 at the roots of $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$, and solving recursively the local assignment problems between forests (the recursion returns are illustrated with arrows in Figure 3.5). Note that if $\epsilon_1 = 1$ (Section 3.3.2), all the branches of $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$ get attached to the roots and the recursive local assignment problems between forests (above) become only one, large, assignment problem between all branches. Thus, when $\epsilon_1 = 1$, this algorithm indeed becomes strictly equivalent to the resolution of the assignment problem involved in $W_2^{\mathcal{D}}$ (Section 2.4.1).

In addition to considering squared costs in our edit distance (Equation 3.4) , our algorithm for the exploration of the search space indeed simplifies the approach by Zhang [Zha96] (used by Sridharamurthy et al. [SMKN20]), as our search space is significantly more constrained.

First, since our solution space only considers partial isomorphisms between rooted subtrees, this implies that the destruction of a node (a branch) $b_j \in \mathcal{B}(f_j)$ necessarily implies the destruction of its subtrees, i.e. of its forest $\mathcal{F}(f_j, b_j)$. Thus, the admissible solutions in [Zha96, SMKN20] consisting in deleting b_j and mapping a subtree $\mathcal{B}(f_i, b_i)$ to one of the subtrees of b_j in the forest $\mathcal{F}(f_j, b_j)$ are no longer admissible given our overall solution space Φ'. The removal of such solutions drastically simplifies

the evaluation of the distance between subtrees (being the minimum of three solutions in [SMKN20], Eq. 12) to Equation 3.18 (containing only one expression to evaluate).

Second, our solution space (rooted partial isomorphisms) also implies that the nodes of $\mathcal{B}(f_i)$ can only be assigned to nodes with the same *depth* in $\mathcal{B}(f_j)$. This further implies that the distance between subtrees (Equation 3.18) only needs to be evaluated for subtrees rooted at nodes of identical depth (Figure 3.5).

Together, these two simplifications (*(i)* simpler subtree distance and *(ii)* distance evaluation restricted to subtrees of identical depth from the root) are the key adaptations of Zhang's algorithm [Zha96] that are required for the exploration of our (more constrained) solution space.

3.3.4 Parallelism

Similarly to Zhang [Zha96], Equation 3.17 and Equation 3.18 can be estimated recursively. To avoid redundant computations, the distances between the forests $\mathcal{F}(f_i, b_i)$ and $\mathcal{F}(f_j, b_j)$ are stored at the entry (b_i, b_j) of a matrix $\mathbb{F}$ (of size $|\mathcal{B}(f_i)| \times |\mathcal{B}(f_j)|$), while the distances between the subtrees $\mathcal{B}(f_i, b_i)$ and $\mathcal{B}(f_j, b_j)$ (used within the assignment problems between higher forests) are stored in a matrix $\mathbb{T}$ (of the same size, see Figure 3.5).

In our work, we additionally express this computation in terms of *tasks*, to leverage task-based shared memory parallelism. In particular, we initiate a task for each independent term of Eqs. 3.17 and 3.18, which is ready for computation (see Figure 3.5).

First, Equation 3.17 is evaluated. For this, we initiate a task at each leaf of $\mathcal{B}(f_i)$. If a task is the last one to compute among all the direct children of a node $b \in \mathcal{B}(f_i)$, it is then authorized to continue and estimate Equation 3.17 in b. Atomic counters in b are implemented (and atomically incremented by the task of each child) to determine which child task is the last one to complete, which enables an efficient lightweight synchronization (Figure 3.5). Overall, Equation 3.17 is completely estimated with this strategy in a bottom-up fashion. Second, Equation 3.18 is evaluated similarly, by initiating a task at each leaf b_i of $\mathcal{B}(f_i)$. In particular, this task will evaluate Equation 3.18 given b_i against all subtrees of $\mathcal{B}(f_j)$ of identical depth (again using independent tasks initiated at the leaves of $\mathcal{B}(f_j)$, see Figure 3.5). Similarly to Equation 3.17, we employ the same lightweight synchronization mechanism based on atomic counters to continue a task over to its parent only when it is the last child task reaching it.

Then, the number of parallel tasks is initially bounded by the number of leaves in the input BDTs (which is typically much larger than the number of cores) and progressively decreases during the computation.

3.4 Wasserstein Geodesics Between Merge Trees

This section introduces our approach for the efficient computation of geodesics between merge trees, according to the metric $W_2^{\mathcal{T}}$ (Section 3.3). For this, we leverage the rooted partial isomorphism resulting from the distance computation, as well as linear interpolations of the matchings, as introduced for persistence diagrams [TMMH14].

3.4.1 Definition and Properties

Given two input merge trees $\mathcal{T}(f_i)$ and $\mathcal{T}(f_j)$, our approach to geodesic computation (Figure 3.6) simply consists in linearly interpolating the rooted partial isomorphism $(\phi', \overline{B_i}, \overline{B_j})$ resulting from the optimization involved in the computation of $W_2^{\mathcal{T}}(\mathcal{B}(f_i), \mathcal{B}(f_j))$ (Equation 3.4). In particular, given the two BDTs $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$, the interpolated BDT, noted $\mathcal{B}_\alpha(f_i \to f_j)$ with $\alpha \in [0,1]$ such that $\mathcal{B}_0(f_i \to f_j) = \mathcal{B}(f_i)$ and $\mathcal{B}_1(f_i \to f_j) = \mathcal{B}(f_j)$, is obtained by considering the union of:

1. the linear interpolation $B_\alpha \subseteq \mathcal{B}_\alpha(f_i \to f_j)$, between the nodes $B_i \subseteq \mathcal{B}(f_i)$ and these of $B_j \subseteq \mathcal{B}(f_j)$, given the isomorphism ϕ' (the trees B_i, B_j and B_α are then isomorphic, Figure 3.6):

$$b(\alpha) = (1-\alpha)b + \alpha\phi'(b) \quad \forall b \in B_i \tag{3.19}$$

2. the linear interpolation of the destruction of the subtrees $\overline{B_i}$, noted

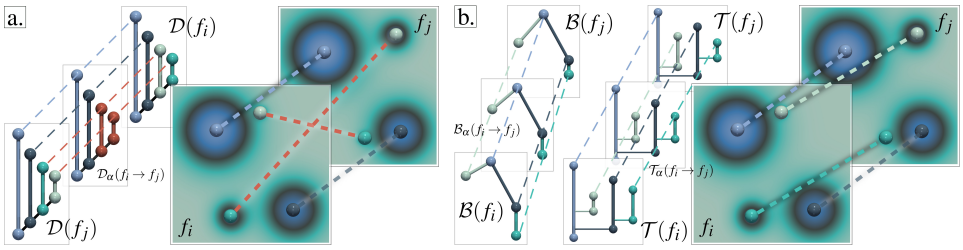


Figure 3.6 – *Our geodesic computation extends interpolation-based geodesics from persistence diagrams (a) to merge trees (b). The interpolated BDT $\mathcal{B}_\alpha(f_i \to f_j)$ is obtained by linear interpolation (with local normalization) of the partial isomorphism ϕ' in the birth/death space. In the data, the feature matching (dashed lines) induced by ϕ' with $W_2^{\mathcal{T}}$ (b) better preserves the global structure of the data than ϕ with $W_2^{\mathcal{D}}$ (a, red crossing).*

$\overline{B_i^\alpha} \subseteq \mathcal{B}_\alpha(f_i \to f_j)$ ($\overline{B_i}$ and $\overline{B_i^\alpha}$ are also isomorphic):

$$b(\alpha) = (1-\alpha)b + \alpha\Delta(b) \quad \forall b \in \overline{B_i} \tag{3.20}$$

3. the linear interpolation of the creation of the subtrees $\overline{B_j}$, noted $\overline{B_j^\alpha} \subseteq \mathcal{B}_\alpha(f_i \to f_j)$ ($\overline{B_j}$ and $\overline{B_j^\alpha}$ are also isomorphic):

$$b(\alpha) = (1-\alpha)\Delta(b) + \alpha b \quad \forall b \in \overline{B_j}. \tag{3.21}$$

Similarly to the distance $W_2^{\mathcal{D}}$ between persistence diagrams, since the edit costs involved in the distance $W_2^{\mathcal{T}}$ are Euclidean distances in the birth/death space (Equation 3.7), the interpolated branches $b(\alpha)$ of $\mathcal{B}_\alpha(f_i \to f_j)$ can be efficiently computed with the simple linear interpolations described above. The resulting interpolated tree $\mathcal{B}_\alpha(f_i \to f_j)$ is indeed on a geodesic given $W_2^{\mathcal{T}}$.

In order to show this, for any two BDTs $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$, we describe the existence of a path between them whose length is equal to $W_2^{\mathcal{T}}\left(\mathcal{B}(f_i), \mathcal{B}(f_j)\right)$ (and thus minimal).

Let $P = \left(\mathcal{B}_t\right)_{t\in[0,1]}$ be a path of BDTs parameterized by t.

We recall that the length $\mathcal{L}(P)$ of P is given by:

$$\mathcal{L}(P) = \sup_{n;0=t_0\leq t_1\leq\cdots\leq t_n=1} \sum_{k=0}^{n-1} W_2^{\mathcal{T}}\left(\mathcal{B}_{t_k}, \mathcal{B}_{t_k+1}\right).$$

Now, let P_α be the path corresponding to the interpolation between $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$, as defined in Equation 3.19, Equation 3.21 and Equation 3.20. We now argue that $\mathcal{L}(P_\alpha) = W_2^{\mathcal{T}}\left(\mathcal{B}(f_i), \mathcal{B}(f_j)\right)$.

Let $(\phi', \overline{B_i}, \overline{B_j})$ be the optimal rooted partial isomorphism between $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$. Moreover, let $\mathcal{B}_s(f_i \to f_j)$ and $\mathcal{B}_t(f_i \to f_j)$ be two interpolated trees obtained respectively with $\alpha = s$ and $\alpha = t$, given $0 \leq s \leq t \leq 1$. We will note ϕ'_s the application on $B_i \cup \overline{B_i} \cup \overline{B_j}$ defined by interpolation (Equation 3.19, Equation 3.21 and Equation 3.20):

$$\phi'_s(b) = \begin{cases} (1-s)b + s\phi'(b) & \text{for } b \in B_i \\ (1-s)b + s\Delta(b) & \text{for } b \in \overline{B_i} \\ sb + (1-s)\Delta(b) & \text{for } b \in \overline{B_j}. \end{cases} \tag{3.22}$$

ϕ'_t is defined similarly for t. Then, as discussed in Section 3.3, since the composition of partial rooted isomorphisms is itself a partial rooted isomorphism, the composition $\phi'_t \circ {\phi'_s}^{-1}$ (which goes from $\mathcal{B}_s(f_i \to f_j)$ to $\mathcal{B}(f_i)$ and then from $\mathcal{B}(f_i)$ to $\mathcal{B}_t(f_i \to f_j)$) does define a valid partial rooted isomorphism between $\mathcal{B}_s(f_i \to f_j)$ and $\mathcal{B}_t(f_i \to f_j)$ and we have:

$$W_2^{\mathcal{T}}\big(\mathcal{B}_s(f_i \to f_j), \mathcal{B}_t(f_i \to f_j)\big) \;\leq\; \Big(\sum_{b \in B_i \cup \overline{B_i} \cup \overline{B_j}} \gamma\big(\phi'_s(b) \to \phi'_t(b)\big)^2\Big)^{1/2}$$

and we also have by definition of ϕ'_s and ϕ'_t (Equation 3.22):

$$\Big(\sum_{b \in B_i \cup \overline{B_i} \cup \overline{B_j}} \gamma\big(\phi'_s(b) \to \phi'_t(b)\big)^2\Big)^{1/2} = (t-s) W_2^{\mathcal{T}}\big(\mathcal{B}(f_i), \mathcal{B}(f_j)\big).$$

Now, given the triangle inequality on the path P_α, we have:

$$\begin{aligned} W_2^{\mathcal{T}}\big(\mathcal{B}(f_i), \mathcal{B}(f_j)\big) \;\leq\; & \\ & W_2^{\mathcal{T}}\big(\mathcal{B}_0(f_i \to f_j), \mathcal{B}_s(f_i \to f_j)\big) \\ +\; & W_2^{\mathcal{T}}\big(\mathcal{B}_s(f_i \to f_j), \mathcal{B}_t(f_i \to f_j)\big) \\ +\; & W_2^{\mathcal{T}}\big(\mathcal{B}_t(f_i \to f_j), \mathcal{B}_1(f_i \to f_j)\big) \\ & \leq \big(s + (t-s) + (1-t)\big) W_2^{\mathcal{T}}\big(\mathcal{B}(f_i), \mathcal{B}(f_j)\big) \\ & = W_2^{\mathcal{T}}\big(\mathcal{B}(f_i), \mathcal{B}(f_j)\big). \end{aligned}$$

If follows that the above inequalities are in fact equalities and we have:

$$W_2^{\mathcal{T}}\big(\mathcal{B}_s(f_i \to f_j), \mathcal{B}_t(f_i \to f_j)\big) = (t-s) W_2^{\mathcal{T}}\big(\mathcal{B}(f_i), \mathcal{B}(f_j)\big).$$

Then, for *any* subdivision $0 = t_0 \leq t_1 \leq \cdots \leq t_n = 1$ of P_α, we have:

$$\begin{aligned} & \sum_{k=0}^{n-1} W_2^{\mathcal{T}}\big(\mathcal{B}_{t_k}(f_i \to f_j), \mathcal{B}_{t_{k+1}}(f_i \to f_j)\big) \\ & \quad = \sum_{k=0}^{n-1} (t_{k+1} - t_k) W_2^{\mathcal{T}}\big(\mathcal{B}(f_i), \mathcal{B}(f_j)\big) \\ & \quad\quad = (t_n - t_0) W_2^{\mathcal{T}}\big(\mathcal{B}(f_i), \mathcal{B}(f_j)\big) \\ & \quad\quad\quad = W_2^{\mathcal{T}}\big(\mathcal{B}(f_i), \mathcal{B}(f_j)\big). \end{aligned}$$

Thus $\mathcal{L}(P_\alpha) = W_2^{\mathcal{T}}\big(\mathcal{B}(f_i), \mathcal{B}(f_j)\big)$.

Hence the space of merge trees equipped with $W_2^{\mathcal{T}}$ is a geodesic space, and $\mathcal{B}_\alpha(f_i \to f_j)$ constructs paths of minimal length on it.

3.4.2 From Branch Decomposition Trees to Merge Trees

The previous section described the computation of geodesics between BDTs, given $W_2^{\mathcal{T}}$. In this section, given an interpolated BDT $\mathcal{B}_\alpha(f_i \to f_j)$, we describe how to retrieve the corresponding merge tree $\mathcal{T}_\alpha(f_i \to f_j)$ (i.e. a merge tree whose BDT is indeed equal to $\mathcal{B}_\alpha(f_i \to f_j)$).

A requirement for an arbitrary BDT $\mathcal{B}$ to be the valid BDT of a merge tree $\mathcal{T}$ is that subtrees of $\mathcal{B}$ need to respect a *nesting condition* on their

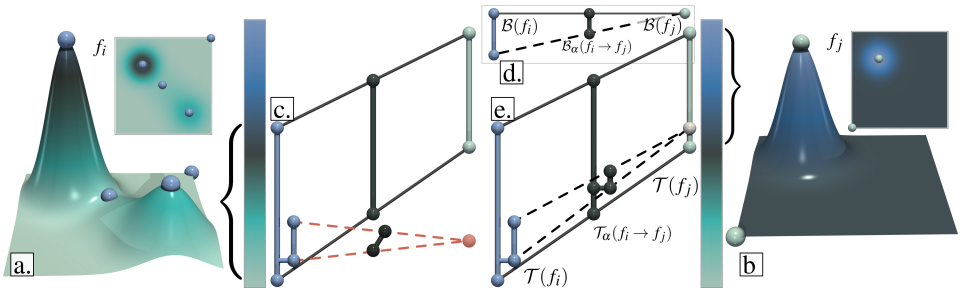


Figure 3.7 – *Given two scalar fields f_i (a) and f_j (b), a simple interpolation of the birth/death values of the branches of their BDTs may result in inconsistencies upon branch destruction (red): the interpolated merge tree (black) in (c) is disconnected, unlike the interpolated BDT (d). Our local normalization (Section 3.4.2) addresses this issue by enforcing nested birth/death values for nested branches. This results in a valid interpolated merge tree (e) whose BDT is indeed equal to the interpolated BDT (e).*

birth/death (i.e. x, y) values (to respect the Elder rule, Section 2.4). In particular, given a direct child b_α^k of a branch $b_\alpha \in \mathcal{B}_\alpha(f_i \to f_j)$, we need to guarantee that $[x_{b_\alpha}^k, y_{b_\alpha}^k] \subseteq [x_{b_\alpha}, y_{b_\alpha}]$. While this is guaranteed by construction for the subset $B_\alpha \subseteq \mathcal{B}_\alpha(f_i \to f_j)$ (B_α is isomorphic to B_i and B_j), this is not necessarily the case for the subsets of $\mathcal{B}_\alpha(f_i \to f_j)$ involved in subtree creation or destruction ($\overline{B_i^\alpha}$ and $\overline{B_j^\alpha}$, Section 3.4.1). In particular, since the branches involved in destructions map independently to the diagonal (Equation 3.20), it is possible that the above nesting condition is not respected along their interpolation. This is shown in Figure 3.7c (red interpolation), where the resulting merge tree, $\mathcal{T}_\alpha(f_i \to f_j)$, becomes disconnected and hence invalid (i.e. $\mathcal{B}_\alpha(f_i \to f_j)$, Figure 3.7d, *is* connected and not equal to the BDT of $\mathcal{T}_\alpha(f_i \to f_j)$ from Figure 3.7c).

In the following, we introduce a pre-processing step for the trees $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$ (together with its inverse post-processing step), which we call *local normalization*, which addresses this issue and guarantees the above nesting condition, even in case of destruction/creation.

Given a direct child ${b_i}^k$ of a branch $b_i \in \mathcal{B}(f_i)$, we consider the following local, birth/death normalization $\mathcal{N}({b_i}^k) = \big(\mathcal{N}_x({b_i}^k), \mathcal{N}_y({b_i}^k)\big)$:

$$\begin{aligned} \mathcal{N}_x(b_i) &= (x_i - x_i')/(y_i' - x_i') \\ \mathcal{N}_y(b_i) &= (y_i - x_i')/(y_i' - x_i'). \end{aligned} \tag{3.23}$$

Once this pre-process is recursively completed, the Wasserstein distance $W_2^{\mathcal{T}}$ between the locally normalized BDTs, noted $\mathcal{N}\big(\mathcal{B}(f_i)\big)$ and $\mathcal{N}\big(\mathcal{B}(f_j)\big)$ is computed as described in Section 3.3.3. Then, the interpolation of the locally normalized BDTs, noted $\mathcal{N}\big(\mathcal{B}_\alpha(f_i \to f_j)\big)$, is evaluated as described in Section 3.4.1. Next, the local normalization is recursively reverted to turn $\mathcal{N}\big(\mathcal{B}_\alpha(f_i \to f_j)\big)$ back into $\mathcal{B}_\alpha(f_i \to f_j)$, by explicitly evalu-

ating $\mathcal{N}^{-1}(b_\alpha^{\mathcal{N}})$ for each branch $b_\alpha^{\mathcal{N}} \in \mathcal{N}\big(\mathcal{B}_\alpha(f_i \to f_j)\big)$. Now, even in case of branch destruction, by construction, the birth/death interval of each interpolated branch $\mathcal{N}(b_\alpha)$, noted $[\mathcal{N}_x(b_\alpha), \mathcal{N}_y(b_\alpha)]$, is included in $[0,1]$ (since $\Delta\big(\mathcal{N}(b_i)\big) \subseteq [\mathcal{N}_x(b_i), \mathcal{N}_y(b_i)] \subseteq [0,1]$). Therefore, after reverting the local normalization, we have the guarantee that $[x^k_{b_\alpha}, y^k_{b_\alpha}] \subseteq [x_{b_\alpha}, y_{b_\alpha}]$ for all the branches b_α of $\mathcal{B}_\alpha(f_i \to f_j)$.

At this stage, $\mathcal{B}_\alpha(f_i \to f_j)$ indeed respects the nesting condition on the birth/death values of all its subtrees. Then, given the dual relation between merge trees and BDTs, the merge tree $\mathcal{T}_\alpha(f_i \to f_j)$ can be simply obtained by creating a vertical branch for each node b_α of $\mathcal{B}_\alpha(f_i \to f_j)$ and connecting them according the arcs of $\mathcal{B}_\alpha(f_i \to f_j)$, as illustrated in Figure 3.7 (right). The distance $W_2^{\mathcal{T}}$ between $\mathcal{N}\big(\mathcal{B}(f_i)\big)$ and $\mathcal{N}\big(\mathcal{B}(f_j)\big)$ then still describes a metric between $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$, such that $\mathcal{B}_\alpha(f_i \to f_j)$ is indeed on a geodesic.

Let $W_2^{\mathcal{N}}\big(\ \mathcal{B}(f_i), \mathcal{B}(f_j)\big)$ be a similarity measure between $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$, defined as:

$$W_2^{\mathcal{N}}\big(\ \mathcal{B}(f_i), \mathcal{B}(f_j)\big) = W_2^{\mathcal{T}}\Big(\mathcal{N}\big(\mathcal{B}(f_i)\big), \mathcal{N}\big(\mathcal{B}(f_j)\big)\Big)$$

where $\mathcal{N}$ is the local normalization described with Equation 3.23. Since $\mathcal{N}$ is invertible, $W_2^{\mathcal{N}}$ inherits all the properties of $W_2^{\mathcal{T}}$ and is also a distance metric.

The *normalized* interpolation $\mathcal{B}_s(f_i \to f_j)$, $s \in [0,1]$ between $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$ is defined as the image by $\mathcal{N}^{-1}$ of the interpolation between the normalized trees $\mathcal{N}(\mathcal{B}(f_i))$ and $\mathcal{N}(\mathcal{B}(f_j))$. Then, given s and t such that $0 \leq s \leq t \leq 1$, it follows that:

$$\begin{aligned} W_2^{\mathcal{N}}\big(\mathcal{B}_s(f_i \to f_j), \mathcal{B}_t(f_i \to f_j)\big) &= (t-s) W_2^{\mathcal{T}}\Big(\mathcal{N}\big(\mathcal{B}(f_i)\big), \mathcal{N}\big(\mathcal{B}(f_j)\big)\Big) \\ &= (t-s) W_2^{\mathcal{N}}\big(\mathcal{B}(f_i), \mathcal{B}(f_j)\big) \end{aligned}$$

which proves that the space of merge trees equipped with $W_2^{\mathcal{N}}$ is a geodesic space, and that the above normalized interpolation constructs paths of minimal length on it.

Note that the local normalization shrinks all the input branches to the interval $[0,1]$, irrespective of their original persistence. To mitigate this effect, we introduce a pre-processing step on the input BDTs, which moves, up the trees, subtrees rooted at branches with a relative persistence smaller than ϵ_3, until their persistence relative to their parent becomes smaller than a threshold ϵ_2. This has the practical effect of reducing the normalized persistence of small branches corresponding to small features. Overall, ϵ_1, ϵ_2 and ϵ_3 are the only parameters of our approach and we use a

unique, default set of values ($\epsilon_1 = 0.05$, $\epsilon_2 = 0.95$ and $\epsilon_3 = 0.9$) in our experiments (Section 3.7). In the remainder, we will consider that all the input BDTs are normalized this way.

3.5 Wasserstein Barycenters of Merge Trees

This section introduces our approach for the computation of barycenters of merge trees, for the metric $W_2^{\mathcal{T}}$ (Section 3.3). The resulting barycenters will serve as core tools for clustering ensembles of merge trees (Section 3.6).

3.5.1 Definition

Let $\mathcal{S}_{\mathcal{B}} = \{\mathcal{B}(f_1), \mathcal{B}(f_2), \ldots, \mathcal{B}(f_N)\}$ be a set of N BDTs. Similarly to Equation 2.4, the Fréchet energy, under the metric $W_2^{\mathcal{T}}$, is given by:

$$F(\mathcal{B}) = \sum_{\mathcal{B}(f_i) \in \mathcal{S}_{\mathcal{B}}} W_2^{\mathcal{T}}\left(\mathcal{B}, \mathcal{B}(f_i)\right)^2. \tag{3.24}$$

We call a *Wasserstein barycenter* of $\mathcal{S}_{\mathcal{B}}$, a BDT $\mathcal{B}^* \in \mathbb{B}$ (where $\mathbb{B}$ is the space of BDTs) which minimizes $F(\mathcal{B})$. It is a centroid of the set, i.e. a tree which minimizes the sum of its distances to the set.

3.5.2 Computation

Our distance $W_2^{\mathcal{T}}$ (Section 3.3) is identical to $W_2^{\mathcal{D}}$, but with a smaller search space, restricted to rooted partial isomorphisms. This enabled an extension of interpolation-based geodesics from persistence diagrams to merge trees (Section 3.4). Given these two components, the strategy presented by Turner et al. [TMMH14] for minimizing the Fréchet energy over the space of persistence diagrams can be directly extended to our framework. For this, we consider an algorithm that resembles a Lloyd relaxation [Llo82], and which alternates an *(i) assignment* and an *(ii) update* procedure. First, the candidate $\mathcal{B}$ is initialized at an arbitrary tree of $\mathcal{S}_{\mathcal{B}}$. Then the assignment step *(i)* computes an optimal assignment $(\phi_i', \overline{B_{\mathcal{B}}}, \overline{B_i})$ between $\mathcal{B}$ and each tree $\mathcal{B}(f_i) \in \mathcal{S}_{\mathcal{B}}$. Next, the update step *(ii)* updates the candidate $\mathcal{B}$ to a position in $\mathbb{B}$ which minimizes $F(\mathcal{B})$ under the current set of assignments $(\phi_i', \overline{B_{\mathcal{B}}}, \overline{B_i})_{i=1,\ldots,N}$. This is achieved by moving each branch $b \in \mathcal{B}$ (in the birth/death space) to the arithmetic mean of the assignments (by generalizing the interpolation defined in Eqs. 3.19, 3.20, and 3.21, to more

than two trees):

$$b \leftarrow \frac{1}{N} \sum_{i=1,\dots,N} \begin{cases} \phi_i'(b) & \text{if } b \in B_{\mathcal{B}} \\ \Delta(b) & \text{if } b \in \overline{B_{\mathcal{B}}} \\ b & \text{if } b \in \overline{B_i}. \end{cases}$$

This overall assignment/update sequence is then iterated , each iteration of this sequence decreases the Fréchet energy constructively).

In particular, once a local minimizer of the Fréchet energy is obtained for a fixed assignment with the *update* step *(ii)*, the subsequent *assignment* step *(i)* does further improve the assignments hence iteratively decreasing the Fréchet energy constructively.

Let F' be a function of an arbitrary BDT $\mathcal{B}$ and of an arbitrary (i.e. not necessarily optimal) set of N rooted partial isomorphisms $({\phi'}_i, \overline{B_{\mathcal{B}}}, \overline{B_i})_{i=1,\dots,N}$ between $\mathcal{B}$ and the N BDTs of $\mathcal{S}_{\mathcal{B}}$:

$$\begin{aligned} F'\big(\mathcal{B}, ({\phi'}_i, \overline{B_{\mathcal{B}}}, \overline{B_i})_{i=1,\dots,N}\big) := \sum_{\mathcal{B}(f_i) \in \mathcal{S}_{\mathcal{B}}} \Big(& \textstyle\sum_{b_i \in B_i} \gamma\big(b_i \to \phi_i'(b_i)\big)^2 \\ & + \textstyle\sum_{b_i \in \overline{B_i}} \gamma(b_i \to \emptyset)^2 \\ & + \textstyle\sum_{b_{\mathcal{B}} \in \overline{B_{\mathcal{B}}}} \gamma(\emptyset \to b_{\mathcal{B}})^2 \Big)^{1/2}. \end{aligned}$$

Now, let $\mathcal{B}_k$ be the candidate barycenter at the iteration k of the algorithm and let $({\phi_i'}^k, \overline{B_{\mathcal{B}}^k}, \overline{B_i^k})$ be the optimal rooted partial isomorphism between $\mathcal{B}_k$ and $\mathcal{B}(f_i)$, computed by the *assignment* step of the iteration. Then, we have:

$$F'\big(\mathcal{B}_k, ({\phi_i'}^k, \overline{B_{\mathcal{B}^k}}, \overline{B_i^k})_{i=1,\dots,N}\big) = \sum_{\mathcal{B}(f_i) \in \mathcal{S}_{\mathcal{B}}} W_2^{\mathcal{T}}\big(\mathcal{B}_k, \mathcal{B}(f_i)\big)^2.$$

Next, the *update* step of the iteration k consists in moving $\mathcal{B}_k$ to $\mathcal{B}_{k+1}$ by placing (in the 2D birth/death space) each branch $b \in \mathcal{B}_k$ at the arithmetic mean of the assignments. Since the arithmetic mean generally minimizes sums of Euclidean distances, we have:

$$F'(\mathcal{B}_{k+1}, ({\phi_i'}^k, \overline{B_{\mathcal{B}^k}}, \overline{B_i^k})_{i=1,\dots,N}) \leq F'(\mathcal{B}_k, ({\phi_i'}^k, \overline{B_{\mathcal{B}^k}}, \overline{B_i^k})_{i=1,\dots,N}).$$

Now, observe that since the previous rooted partial isomorphisms are not optimal anymore for $\mathcal{B}_{k+1}$, we also have:

$$\sum_{\mathcal{B}(f_i) \in \mathcal{S}_{\mathcal{B}}} W_2^{\mathcal{T}}\big(\mathcal{B}_{k+1}, \mathcal{B}(f_i)\big)^2 \leq F'(\mathcal{B}_{k+1}, ({\phi_i'}^k, \overline{B_{\mathcal{B}^k}}, \overline{B_i^k})_{i=1,\dots,N}).$$

Once $\mathcal{B}_{k+1}$ is fixed, all the rooted partial isomorphisms are then optimized again with the *assignment* step of the iteration $k+1$ to attain:

$$F'\big(\mathcal{B}_{k+1}, ({\phi'}_i^{k+1}, \overline{B_{\mathcal{B}^{k+1}}}, \overline{B_i^{k+1}})_{i=1,\dots,N}\big) = \sum_{\mathcal{B}(f_i) \in \mathcal{S}_{\mathcal{B}}} W_2^{\mathcal{T}}\big(\mathcal{B}_{k+1}, \mathcal{B}(f_i)\big)^2.$$

The result of these two steps is that:

$$\begin{aligned} F'\big(\mathcal{B}_{k+1}, (\phi_i'^{k+1}, \overline{B_{\mathcal{B}}{}^{k+1}}, \overline{B_i{}^{k+1}})_{i=1,\dots,N}\big) &\leq F'(\mathcal{B}_k, (\phi_i'^{k}, \overline{B_{\mathcal{B}}{}^{k}}, \overline{B_i{}^{k}})_{i=1,\dots,N}) \\ \sum_{\mathcal{B}(f_i)\in\mathcal{S}_{\mathcal{B}}} W_2^{\mathcal{T}}\big(\mathcal{B}_{k+1}, \mathcal{B}(f_i)\big)^2 &\leq \sum_{\mathcal{B}(f_i)\in\mathcal{S}_{\mathcal{B}}} W_2^{\mathcal{T}}\big(\mathcal{B}_k, \mathcal{B}(f_i)\big)^2. \end{aligned}$$

Then, each iteration of our algorithm indeed decreases the Fréchet energy. Since there is a finite number of combinations of rooted partial isomorphisms between the barycenter and the N input trees $\mathcal{B}(f_i)$, it follows that the algorithm converges, in a finite number of steps, to a local minimum $\mathcal{B}^*$ of the Fréchet energy (if multiple, equally valued, optimal sets of assignments exist between $\mathcal{B}^*$ and $\mathcal{S}_{\mathcal{B}}$, each one needs to be explored with the update step of our algorithm). In practice, we stop our algorithm when the Fréchet energy has decreased by less than 1% between consecutive iterations.

In our implementation, the algorithm stops and returns the barycenter estimation $\mathcal{B}^*$ when the Fréchet energy decreased by less than 1% between two consecutive iterations. Given $\mathcal{B}^*$, we obtain its dual merge tree $\mathcal{T}^*$ as described in Section 3.4.2. Figure 3.15 illustrates a barycenter computed with this strategy for a toy example.

3.5.3 Parallelism

The N assignment problems (between the candidate $\mathcal{B}$ and the trees of the set $\mathcal{S}_{\mathcal{B}}$, Section 3.5.2) are independent and can be computed in parallel. However, this naive strategy is subject to load imbalance, as the input trees can have different sizes. Hence, each iteration would be bounded by the sequential execution of the largest of the N assignment problems.

We address this issue by leveraging the task-based parallelization of our distance computation algorithm (Section 3.3.4). In particular, we use a single task pool for all of the N assignment problems. Then, the task environment picks up at runtime the tasks to compute irrespective of their tree of origin, and place them on different threads. This fine scheduling granularity has the beneficial effect of triggering the execution of the tasks of a new assignment problem while a first problem is reaching completion (and thus exploiting less threads, Section 3.3.4). This improves thread load imbalance and thus increases the overall parallel efficiency.

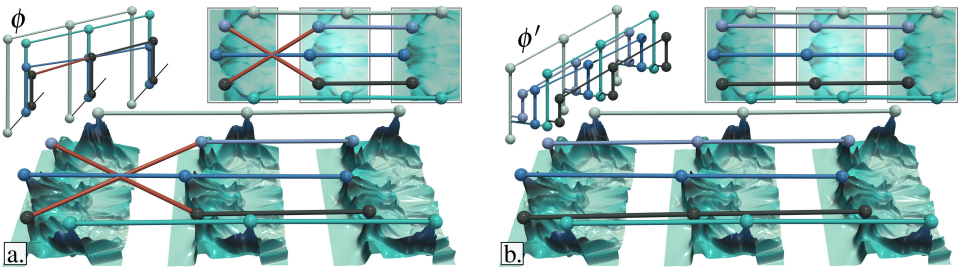


Figure 3.8 – *Tracking features (the five most persistent maxima, spheres) in time-varying 2D data (ion density during universe formation [TCWN08]): optimal assignment ϕ (a) of $W_2^{\mathcal{D}}$ (Section 2.4), optimal isomorphism ϕ' (b) of $W_2^{\mathcal{T}}$ (Section 3.3.2). Since $W_2^{\mathcal{D}}$ considers persistence pairs individually, it can generate incorrect matchings resulting in a characteristic* crossing *(a, red). Our distance improves this aspect (b) thanks to its more constrained search space, which better preserves the global structure of the data.*

3.6 Applications

The section illustrates the utility of our contributions (distances, geodesics, and barycenters) in concrete visualization tasks (Figure 3.1).

3.6.1 Branch Matching for Feature Tracking

Our distance (Section 3.3.3) relies on the optimization of a partial isomorphism between the input BDTs. Then, the resulting matchings can be used to track features in time-varying data, as studied for persistence diagrams [SPCT18]. Figure 3.8 illustrates this on a temporal sequence (SciVis contest 2008 [TCWN08]). Since $W_2^{\mathcal{D}}$ considers persistence pairs individually, it can generate inconsistent matchings with a typical incorrect *crossing* in the feature tracking (already visible on synthetic data, Figure 3.6). Our distance $W_2^{\mathcal{T}}$ improves this aspect by better preserving the global structure of the data, thanks to our more constrained, merge-tree driven, assignment search space. Overall, our matchings provide visual hints to the users, to help them relate features from distinct time steps.

3.6.2 Geodesics for Temporal Reduction

The topological analysis of time-varying data typically requires the computation of a topological representation, for instance a merge tree, for each time step. Although merge trees are usually orders of magnitude smaller than the original data, the resulting sequence of merge trees can still represent considerable amounts of data. To address this issue, we exploit our geodesic computation (Section 3.4) for the reduction of temporal sequences of merge trees. In particular, we greedily remove from the se-

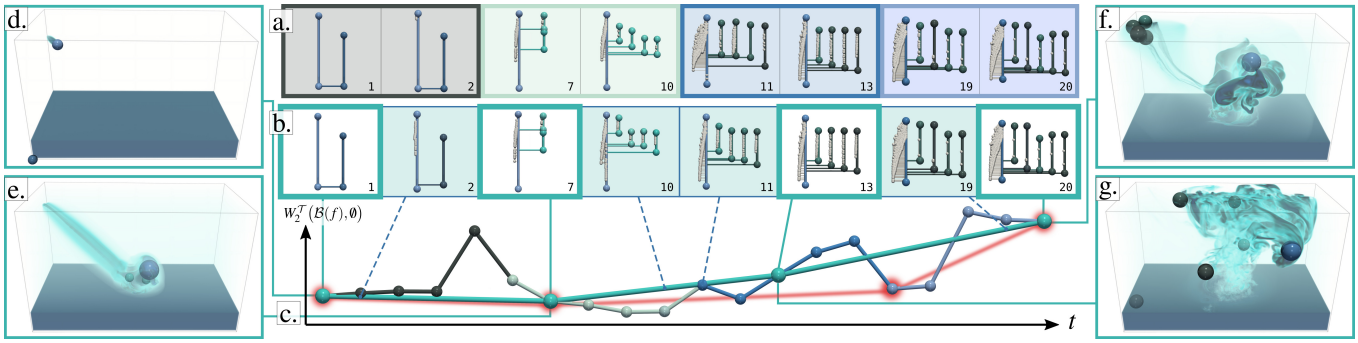


Figure 3.9 – *Geodesic computation for the reduction (b) of temporal sequences of merge trees (a). Our algorithm greedily removes from the sequence the trees that it can accurately estimate by geodesic computation (trees with blue background (b)). This reduction is also visualized with the three curves in (c), plotting the distance $W_2^{\mathcal{T}}$ to the empty tree $\emptyset$ over time (multiple colors: original sequence, cyan: reduction by $W_2^{\mathcal{T}}$, red: reduction by $W_2^{\mathcal{D}}$). This iterated removal of trees highlights* key frames *in the sequence (d-g) corresponding to key phases of an asteroid impact simulation [PG18]: initial state (black, time steps 1-5), approach (light green, 6-10), impact (blue, 11-15), aftermath (light blue, 16-20). In contrast, a similar greedy optimization based on the distance $W_2^{\mathcal{D}}$ between persistence diagrams (red curve) fails at capturing the* impact *phase (blue) of this sequence.*

quence, one by one, the merge trees which can be accurately reconstructed by simple geodesic computation, until the sequence only contains a target number of merge trees.

Let $\mathcal{S} = \{\mathcal{B}(f_1), \mathcal{B}(f_2), \ldots, \mathcal{B}(f_N)\}$ be the input temporal sequence of BDTs (we assume a regular temporal sampling). Let $\mathcal{K} \subseteq \mathcal{S}$ be a set of key frames. Let $\mathcal{S}' = \{\mathcal{B}'(f_1), \mathcal{B}'(f_2), \ldots, \mathcal{B}'(f_N)\}$ be a *reduced* temporal sequence, where:

$$\mathcal{B}'(f_i) = (1-\alpha_i)\mathcal{B}(f_j) + \alpha_i \mathcal{B}(f_k) \tag{3.25}$$

where $\mathcal{B}(f_j)$ and $\mathcal{B}(f_k)$ are two consecutive trees in $\mathcal{K}$, such that $j \leq i \leq k$ and $\alpha_i = (i-j)/(k-j)$. $\mathcal{B}'(f_i)$ is then on a geodesic between $\mathcal{B}(f_j)$ and $\mathcal{B}(f_k)$. We introduce the following distance between the temporal sequences $\mathcal{S}$ and $\mathcal{S}'$:

$$d_S(\mathcal{S}, \mathcal{S}') = \Big(\sum_{i=0}^{N} W_2^{\mathcal{T}}\big(\mathcal{B}(f_i), \mathcal{B}'(f_i)\big)^2 \Big)^{1/2}. \tag{3.26}$$

d_S is indeed a metric since it is a composition of metrics (being the L^2 norm between vectors of BDTs under the metric $W_2^{\mathcal{T}}$).

Our algorithm for temporal reduction consists in initializing $\mathcal{K}$ with the entire input sequence ($\mathcal{K} \leftarrow \mathcal{S}$) and then removing greedily, at each iteration, the tree $\mathcal{B}^*$ from $\mathcal{K}$ ($\mathcal{K} \leftarrow \mathcal{K} - \{\mathcal{B}^*\}$) which minimizes $d_S(\mathcal{S}, \mathcal{S}')$, and which, hence, better preserves the input sequence, until $\mathcal{K}$ reaches a target size.

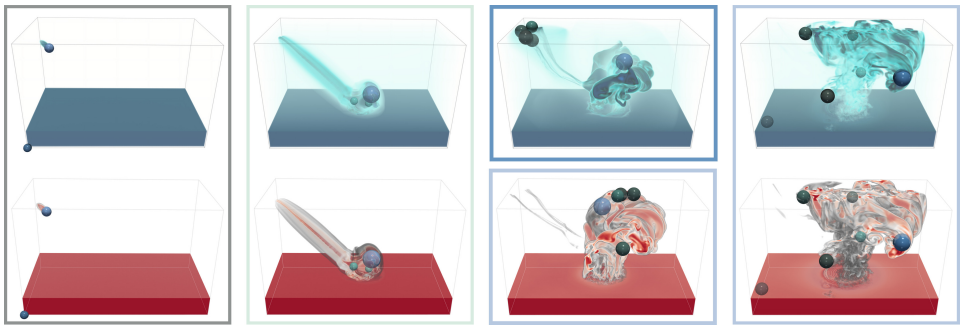


Figure 3.10 – *Comparison between the* key frames *identified by our temporal reduction algorithm, with regard to* $W_2^{\mathcal{T}}$ *(blue, top) and with regard to* $W_2^{\mathcal{D}}$ *(red, bottom). By construction, the reduction algorithm identifies as key frames the first and last time steps, irrespective of the employed metric.*

This enables the reliable visualization of time-varying sequences of merge trees at greatly reduced storage costs. The remaining merge trees (white background, Figure 3.9) correspond to *key frames* of the sequence, i.e. time steps of particular significance in terms of the features of interest. In contrast, a similar strategy based on persistence diagram interpolation (red curve) fails at identifying a key frame in one of the key phases of the sequence (impact, in blue). Also, note that the reduced merge trees (reconstructed with geodesics, blue background) are visually highly similar to the trees from the input sequence.

Figure 3.10 shows the temporal reduction performed by this algorithm on the *Asteroid impact* sequence (see Appendix A). This figure illustrates *key frames*, which correspond to time steps for which $\mathcal{B}(f_i) = \mathcal{B}'(f_i)$: these are the time steps which have *not* been removed from the sequence through the reduction. In particular, this figure compares the usage of two metrics in the reduction algorithm: $W_2^{\mathcal{T}}$ (blue, top) and $W_2^{\mathcal{D}}$ (red, bottom, obtained with $\epsilon_1 = 1$). By construction, since our reduction algorithm is based on interpolation only, the first and last BDTs in the sequence $\mathcal{S}$ are always kept in the reduced sequence $\mathcal{S}'$. In other words, the first (leftmost, Figure 3.10) and last (rightmost, Figure 3.10) time steps are always identified as key frames, irrespective of the employed metric. For this specific example, the second key frame (second from left, Figure 3.10) also happens to be identical for both metrics. In contrast to the sequence extremities, the common identification of this time step as a key frame by $W_2^{\mathcal{T}}$ and $W_2^{\mathcal{D}}$ is not obtained by construction: the reduction algorithm did select this key frame in both configurations. Then, only the third key frame (third from left, Figure 3.10) is different in this example. In particular, when using $W_2^{\mathcal{T}}$, the reduction algorithm identifies one key frame

per key phase of the simulation (see Appendix A, each key phase is represented in Figure 3.10 with a frame of distinct color). In contrast, when using $W_2^{\mathcal{D}}$, the third key frame belongs to the same key phase as the last key frame (*"Aftermath"*, light blue frame). Then the reduction driven by $W_2^{\mathcal{D}}$ fails at identifying a key frame for the third key phase (*"Impact"*, dark blue frame). This is confirmed visually in Figure 3.10, as the third key frame identified with $W_2^{\mathcal{T}}$ (in blue) seems to represent an intermediate step in the simulation between the second and fourth key frames. In contrast, the third key frame identified by the reduction with $W_2^{\mathcal{D}}$ (in red) is more visually similar to the fourth key frame, and hence possibly more redundant.

3.6.3 Barycenters for Topological Clustering

To understand the main trends within an ensemble, in terms of features of interest, it may be desirable to cluster the ensemble by grouping members with a similar *topological profile*. For this, we adapt the k-means algorithm [Elko3, CKV13] to the problem of clustering merge trees. In particular, this can be easily achieved by using our merge tree barycenter computation algorithm (Section 3.5) as the centroid estimation routine of k-means, and by using $W_2^{\mathcal{T}}$ (Section 3.3) to measure the distance between merge trees. Note that in practice, our entire computational framework is implemented in this single clustering algorithm (with a unique task pool), as the above clustering generalizes the barycenter problem ($k = 1$) as well as the geodesic and distance problems ($N = 2$).

Figure 3.11 and Figure 3.12 present clustering examples obtained with this strategy on an acquired [FHG+14] and cosmology ensemble [HGTS15]. In both cases, our approach correctly assigns the members to each cluster. Moreover, the centroids computed by our algorithm pro-

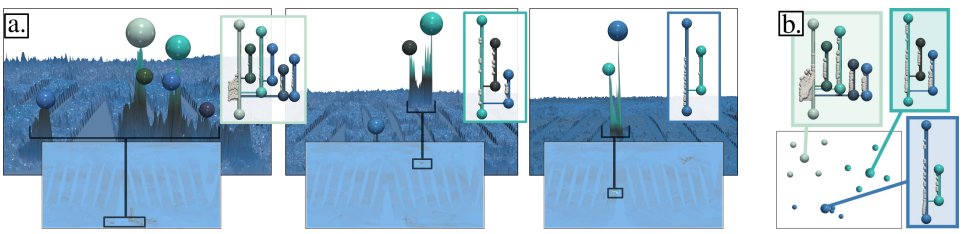


Figure 3.11 – *Three members (a) of an acquired ensemble, corresponding to distinct volcanic eruptions [FHG+14]. Our clustering approach correctly assigns the members to each cluster (b, distinct colors in the planar view, generated in a post-process by multi-dimensional scaling of $W_2^{\mathcal{T}}$). Our centroids (larger spheres in the planar view) provide a visual summary of the features of interest (matching colors) for each cluster.*

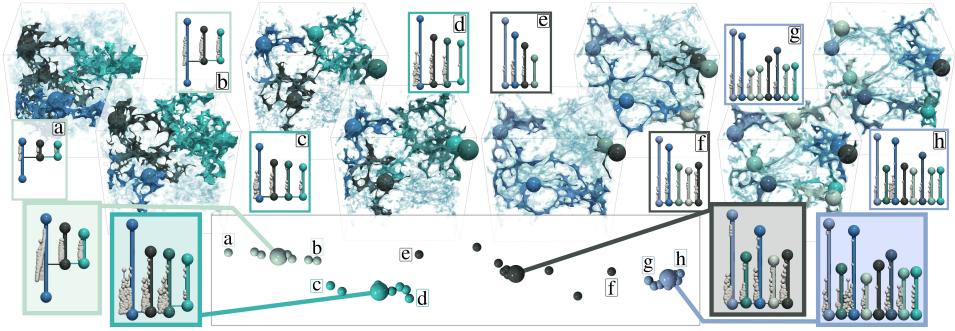


Figure 3.12 – *Eight members (top) of a cosmology ensemble [HGTS15], and their merge trees (persistent maxima are displayed with matching colors in the data). Our clustering approach correctly assigns the members to each cluster (distinct colors in the bottom planar view, generated in post-process by multi-dimensional scaling of $W_2^{\mathcal{T}}$). Our centroids (large spheres, bottom) provide a visual summary which is well representative of the trees in the cluster (same number and persistence of large branches, automatically color-coded based on their matching to their centroid).*

vide a visual summary of the features of interest found in each cluster, enabling global overviews (Figure 3.11, right, and Figure 3.12, bottom) summarizing the topological profile of each of the main trends found in the ensemble. In both figures, the tree branches of the ensemble members are automatically colored with the color of their matched centroid branch. This matching visualization enables users to visually relate the centroid to concrete features in the data (Figure 3.11) and to compare matching features across multiple members (i.e. which have been matched to the same centroid branch, Figure 3.12). Then, the centroid, in addition to being a visual summary, also acts as a reference point for the visual comparison of ensemble members.

3.7 Results

This section presents experimental results obtained on a computer with two Xeon CPUs (3.2 GHz, 2x10 cores, 96GB of RAM). The input merge trees were computed with FTM [GFJT19a] and pre-processed to discard noisy features (persistence simplification threshold: 0.25% of the data range). We implemented our approach in C++ (with the OpenMP task runtime), as modules for TTK [TFL+17, BMBF+19].

Our experiments were performed on a variety of simulated and acquired 2D and 3D ensembles used in previous work [FFST18] (vorticity and sea surface height) or extracted from past SciVis contests: 2004 (wind velocity magnitude [WBKS04]), 2006 (wavefront velocity magni-

tude [ODM+06]), 2008 (ion concentration [TCWN08]), 2014 (sulfur dioxide concentration [FHG+14]), 2015 (dark matter density [HGTS15]), 2016 (salt concentration [GGH+16]), 2017 (pressure [WCG+17]), 2018 (matter density [PG18]). A detailed specification of these ensembles is provided in Appendix A.

3.7.1 Time Performance

The time complexity of our algorithm for exploring the search space of $W_2^{\mathcal{T}}$ (Section 3.3.3) is similar to that of the edit distance [Zha96, SMKN20]. It takes $\mathcal{O}(|\mathcal{B}|^2)$ steps in practice, with $|\mathcal{B}|$ the number of nodes in the input BDTs (in our implementation, each local forest assignment problem is solved with the efficient *Auction* approximation [Ber81] with default parameters). Once $W_2^{\mathcal{T}}\left(\mathcal{B}(f_i), \mathcal{B}(f_j)\right)$ is computed, the computation of a point on the geodesic (Section 3.4) between $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$ is obtained in $\mathcal{O}(|\mathcal{B}|)$ steps. Regarding our barycenter computation algorithm (Section 3.5), each of its iterations takes $\mathcal{O}(N|\mathcal{B}|^2)$ steps. Table 3.1 evaluates the practical time performance of our computational framework for the barycenter computation (which includes itself distance and geodesic computations). In sequential mode, we observe that the running time is indeed a function of the number of ensemble members (N) and the average size of the trees ($\mathcal{B}$). It is slightly slower for $W_2^{\mathcal{T}}$ than for $W_2^{\mathcal{D}}$, but runtimes remain comparable overall. In parallel, speedups are the most important for the largest examples. However, the iterative nature of our barycenter optimization algorithm seems to limit parallel efficiency globally (the end of each iteration still constitutes a strong synchronization). For the smaller

Table 3.1 – *Running times (in seconds, 10 run average) of our approach for the barycenter computation, with respect to $W_2^{\mathcal{D}}$ ($\epsilon_1 = 1$, Section 3.3.3, sequential) and to our new metric $W_2^{\mathcal{T}}$ (sequential, then with 20 cores).*

Dataset	N	$\lvert\mathcal{B}\rvert$	$W_2^{\mathcal{D}}$ (1 c.)	$W_2^{\mathcal{T}}$ (1 c.)	$W_2^{\mathcal{T}}$ (20 c.)	Speedup
Asteroid Impact [PG18] (3D)	7	1,295	514.71	450.91	93.11	4.84
Cloud processes [WCG+17] (2D)	12	1,209	54.90	124.99	35.14	3.55
Viscous fingering [GGH+16] (3D)	15	118	5.68	5.12	3.89	1.31
Dark matter [HGTS15] (3D)	40	2,592	3,172.37	3,083.24	471.45	6.53
Volcanic eruptions [FHG+14] (2D)	12	811	171.13	140.02	48.52	2.88
Ionization front [TCWN08] (2D)	16	135	10.40	12.10	8.20	1.47
Ionization front [TCWN08] (3D)	16	763	682.76	1,277.72	219.61	5.81
Earthquake [ODM+06] (3D)	12	1,203	191.54	509.59	117.31	4.34
Isabel [WBKS04] (3D)	12	1,338	330.88	284.19	62.70	4.53
Starting Vortex [FFST18] (2D)	12	124	7.72	5.58	6.11	0.91
Sea Surface Height [FFST18] (2D)	48	1,787	4,509.78	10,557.07	881.49	11.97
Vortex Street [FFST18] (2D)	45	23	1.71	1.90	1.44	1.31

examples, the cost of the task runtime seems to become non-negligible in comparison to the actual computation, resulting in moderate speedups. Still, our parallelization significantly reduces runtimes overall, with less than 3 minutes of computation on average and at most 15 minutes for the largest examples.

3.7.2 Framework Quality

$W_2^{\mathcal{T}}$ is indeed a distance metric (Section 3.3.2). It is more discriminative than $W_2^{\mathcal{D}}$ (i.e. $W_2^{\mathcal{T}} \geq W_2^{\mathcal{D}}$, Section 3.3, Figure 3.2). Figure 3.13 evaluates empirically its stability. For this, given a scalar field f_i, a noisy version f_j is created such that $\|f_i - f_j\|_\infty \leq \epsilon$, for increasing values of ϵ. Then, we observe the evolution of $W_2^{\mathcal{T}}\left(\mathcal{B}(f_i), \mathcal{B}(f_j)\right)$, as a function of ϵ (Figure 3.13, right), to estimate how $W_2^{\mathcal{T}}$ varies under input perturbations. For $\epsilon_1 = 1$, we have $W_2^{\mathcal{T}} = W_2^{\mathcal{D}}$ (Section 3.3) and the curve evolves nearly linearly ($W_2^{\mathcal{D}}$ is stable [TMMH14]). For other ϵ_1 values, the curves indicate clear *transition* points (colored dots) before which $W_2^{\mathcal{T}}$ evolves nearly linearly too. This indicates that for reasonable noise levels (smaller than the ϵ value of each transition point, vertical lines), $W_2^{\mathcal{T}}$ is also stable and that only mild increases of ϵ_1 result in fast shifts of these transition points (to an accepted noise level of 64% at $\epsilon_1 = 0.15$). This illustrates overall that the stability of $W_2^{\mathcal{T}}$ can indeed be controlled with ϵ_1 and that small values already lead to stable results for reasonable noise levels. A detailed empirical analysis of the other two parameters of our approach (ϵ_2, ϵ_3, Section 3.4.2) is provided in Appendix B.

Next, we study the practical relevance of $W_2^{\mathcal{T}}$ by evaluating our clustering performance. For this, each ensemble of Table 3.1 is associated with a ground truth classification (distinct phases of a time-varying phenomenon, distinct input parameters, etc), by following the com-

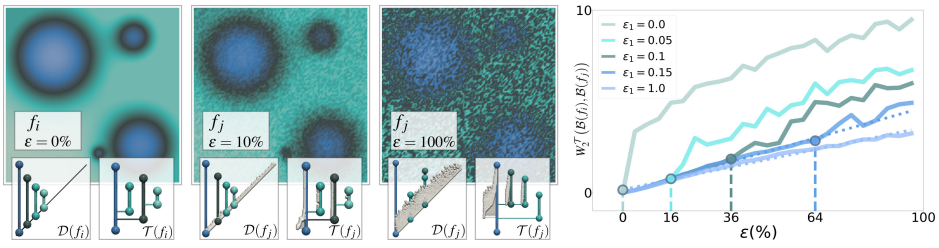


Figure 3.13 – *Empirical stability evaluation. Given an input scalar field f_i, a noisy version f_j is created by inserting a random noise of increasing amplitude ϵ (left). The evolution of $W_2^{\mathcal{T}}\left(\mathcal{B}(f_i), \mathcal{B}(f_j)\right)$ with ϵ (right), for varying values of ϵ_1 (Section 3.3.2), indicates clear transition points (colored dots) before which $W_2^{\mathcal{T}}$ evolves nearly linearly. Before these transition points (i.e. before these noise levels, vertical lines), $W_2^{\mathcal{T}}$ is stable.*

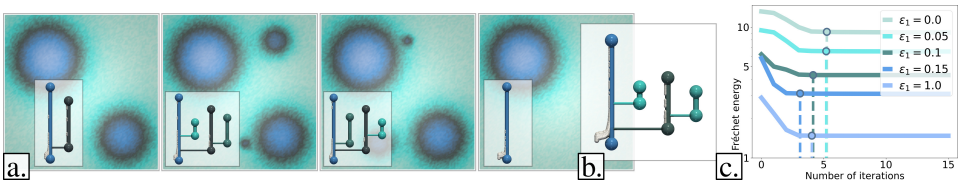


Figure 3.14 – *Evolution of the Fréchet energy for estimating the barycenter (b) for an ensemble of 100 noisy variants of four fields (a). The energy (c) is shown for several ϵ_1 values (Section 3.3.2). In practice, we stop the algorithm when the energy decreases by less than 1% (vertical lines).*

panion specifications [WBKS04, ODM+06, TCWN08, FHG+14, HGTS15, GGH+16, WCG+17, PG18]. Clustering performance is evaluated with accepted scores, namely the normalized mutual information and adjusted rand index (NMI, ARI). When using our barycenters (Section 3.5), our clustering approach (Section 3.6.3) achieves a perfect classification for all ensembles ($NMI = ARI = 1$). These scores decrease to $NMI = 0.78$ and $ARI = 0.69$ on average when using, within k-means, a barycenter of persistence diagrams [TMMH14] ($\epsilon_1 = 1$), and to $NMI = 0.73$ and $ARI = 0.56$ when using the 1-center of Yan et al. [YWM+19b] (obtained with the authors' implementation [YWM+19a], using leaf labels generated by our distance computation, Section 3.3). This simply confirms experimentally that 1-centers in general are not suited for clustering tasks. A standard clustering approach (multi-dimensional scaling to kD followed by k-means) using the distance D_E [SMKN20] achieves lower average scores than our approach, with $NMI = 0.89$ and $ARI = 0.85$ on average. Overall, this confirms that D_E induces more discriminative classifiers than $W_2^{\mathcal{D}}$, and that our metric $W_2^{\mathcal{T}}$ further improves that.

Figure 3.14 shows the evolution of the Fréchet energy for our barycenter algorithm (Section 3.5) for various ϵ_1 values. In practice, the algorithm stops when the Fréchet energy decreases by less than 1% between consecutive iterations, which occurs early in the process.

Figure 3.15 provides a visual comparison between our barycenter and the 1-center of Yan et al. [YWM+19b] (obtained with the authors' implementation [YWM+19a], using leaf labels generated by our distance computation, Section 3.3). This figure confirms the general sensitivity in practice of 1-centers to outliers, and the ability of barycenters to better capture the main trends in the ensemble. From a qualitative perspective, our framework enables the computation of faithful interpolations of merge trees: the reconstructed trees, blue background (Figure 3.9), are visually very similar to the input trees. Moreover, our framework produces

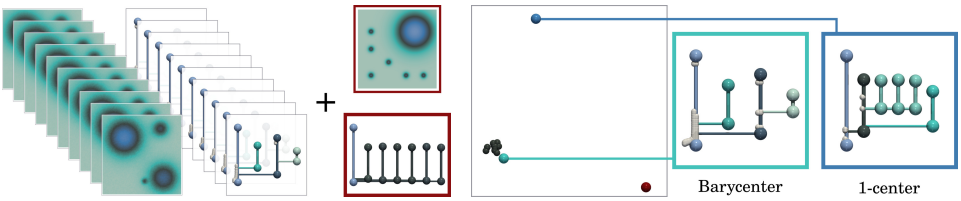


Figure 3.15 – *Visual comparison between our barycenter (Section 3.5) and the 1-center of Yan et al. [YWM+19b, YWM+19a]. Left: an ensemble is created with an outlier member f_j (red, 7 persistent branches) and 10 noisy versions of a field f_i (4 persistent branches). Right: planar view of the ensemble computed by multi-dimensional scaling of $W_2^{\mathcal{T}}$. The barycenter computed with our approach (cyan) is more similar to the merge trees of f_i (same number and persistence of large branches) and hence better captures the overall trend of the ensemble, despite the presence of the outlier f_j (red sphere).*

barycenters (Figure 3.1, Figure 3.11, and Figure 3.12) which capture well the main features of the input ensemble: for each cluster, the resulting centroid is visually similar to the input trees of the cluster (same number and persistence of large branches). Then, our clustering framework, coupled with our centroids, provides a faithful visual summary of the features of interest, for each of the main trends (i.e. for each cluster) found in the ensemble.

3.7.3 Limitations

The search space associated with our metric $W_2^{\mathcal{T}}$ is constrained to rooted partial isomorphisms. Then, if a matching exists between two BDTs (i.e. if they are not both destructed when optimizing $W_2^{\mathcal{T}}$), it has to match their roots together. In other words, $W_2^{\mathcal{T}}$ nearly always matches the most persistent branch of the two trees together, which might be too restrictive (in particular for feature tracking applications). Note however, that $W_2^{\mathcal{D}}$ behaves equivalently: the most persistent branch of $\mathcal{B}(f_i)$ corresponds to the component of $f_{i_{-\infty}}^{-1}(w)$ created in the global minimum of f_i, which in principle has infinite persistence and which is typically treated separately when evaluating $W_2^{\mathcal{D}}$. Similarly to Sridharamurthy et al. [SMKN20], saddle swap instabilities are handled in our approach by a pre-processing step which merges adjacent saddles (controlled by ϵ_1). An alternative would consist in exploring the space of all possible branch decompositions (not necessarily persistence-driven), as studied by Beketayev et al. [BYM+14]. However, the search space would then become significantly larger. Moreover, the nesting of birth/death values within the BDTs would no longer be guaranteed, which is however a key property which we exploit in our framework (Section 3.4). When computing barycenters of persistence dia-

grams, Vidal et al. [VBT20] showed that the optimization could be drastically accelerated by introducing persistence pairs progressively along the iterations, while implicitly maintaining previous assignments at each initialization. We leave the study of such a progressive strategy to future work, although the fact that $W_2^{\mathcal{T}}$ handles many small assignment problems (unlike $W_2^{\mathcal{D}}$) indicates that such a strategy may result in only modest gains for merge trees. Figure 3.13 provides an empirical evaluation of the stability of $W_2^{\mathcal{T}}$. Similarly to Sridharamurthy et al. [SMKN20], we believe that the theoretical investigation of the stability of $W_2^{\mathcal{T}}$ goes beyond the scope of this work and we leave it for future work.

3.8 Summary

In this chapter, we presented a computational framework for the estimation of distances, geodesics and barycenters of merge trees, with applications to feature tracking, temporal reduction and ensemble clustering and summarization. Our approach filled the gap between the edit distance [SMKN20] and existing optimization frameworks for persistence diagrams [TMMH14]. Our work enables faithful interpolations of merge trees (Figure 3.9) and the generation of merge trees representative of a set (Figure 3.1, Figure 3.11, and Figure 3.12). Moreover, our task-based algorithm enables automatic barycenter computations within minutes for real-life ensembles.

4 Principal Geodesic Analysis of Merge Trees and Persistence Diagrams

Contents

This chapter presents a computational framework for the Principal Geodesic Analysis of merge trees (MT-PGA), a novel adaptation of the celebrated Principal Component Analysis (PCA) framework [Pea01] to the Wasserstein metric space of merge trees introduced in Chapter 3. We formulate MT-PGA computation as a constrained optimization problem, aiming at adjusting a basis of orthogonal geodesic axes, while minimizing a fitting energy. We introduce an efficient, iterative algorithm which exploits shared-memory parallelism, as well as an analytic expression of the fitting energy gradient, to ensure fast iterations. Our approach also trivially extends to extremum persistence diagrams. Extensive experiments on public ensembles demonstrate the efficiency of our approach – with MT-PGA computations in the orders of minutes for the largest examples. We show the utility of our contributions by extending to merge trees two typical PCA applications. First, we apply MT-PGA to *data reduction* and reliably compress merge trees by concisely representing them by their first coordinates in the MT-PGA basis. Second, we present a *dimensionality reduction* framework exploiting the first two directions of the MT-PGA basis to generate two-dimensional layouts of the ensemble. We augment these layouts with persistence correlation views, enabling global and local visual inspections of the feature variability in the ensemble. In both applications, quantitative experiments assess the relevance of our framework. Finally, we provide a C++ implementation that can be used to reproduce our results.

The work presented in this chapter has been published in the journal IEEE Transactions on Visualization and Computer Graphics in 2022 [PVT23]. It was certified replicable by the Graphics Replicability Stamp Initiative (http://www.replicabilitystamp.org/). Our implementation is available at https://github.com/MatPont/MT-PGA and the data used in this work at https://github.com/MatPont/WassersteinMergeTreesData. It is also integrated in the Topology ToolKit [TFL+17].

Our Contributions in one Image

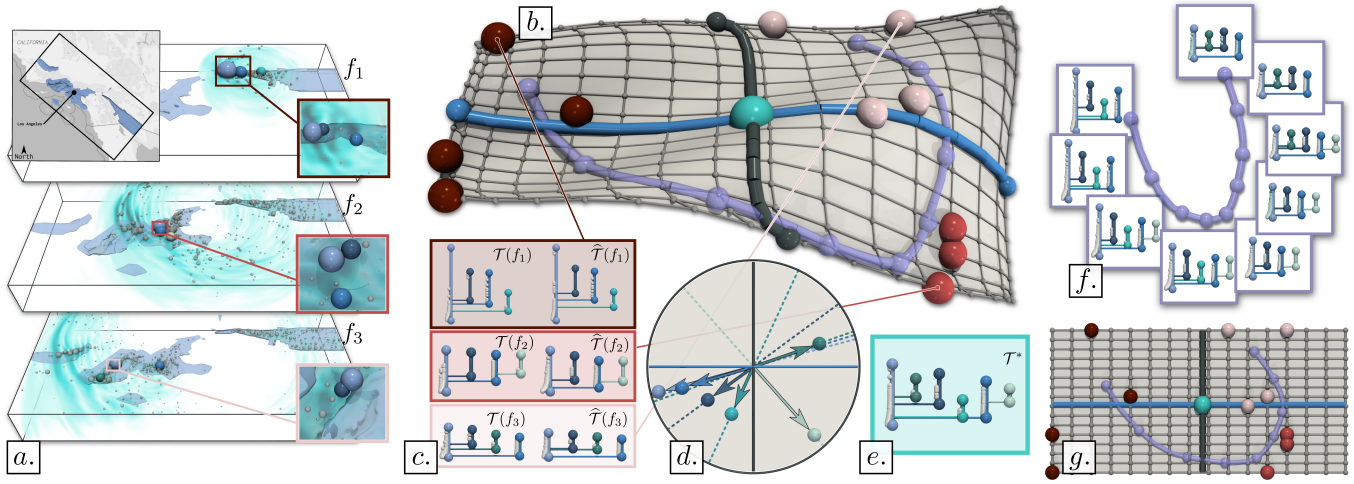


Figure 4.1 – *Visual analysis of the* Earthquake *ensemble with Merge Tree Principal Geodesic Analysis (MT-PGA,* (a)*: one member per ground-truth class). Our framework computes a coordinate system* (b) *for the Wasserstein metric space of merge trees* $\mathbb{B}$ *by adjusting geodesic axes (blue and black,* (b)*) to optimize a fitting energy. This enables the adaptation to merge trees of typical applications of Principal Component Analysis, such as data reduction, where the input trees are accurately reconstructed (*(c)*, right), by simply storing their MT-PGA coordinates, or dimensionality reduction. MT-PGA enables the computation of a* Principal Geodesic Surface (b)*, which complements its planar layout* (g) *by better conveying visually the curved nature of* $\mathbb{B}$*. MT-PGA supports the efficient reconstruction of user-defined locations, for the interactive exploration of* $\mathbb{B}$*: the reconstruction of the purple curve* (f) *enables the navigation from the trees of the first cluster (dark red,* (b)*) to the second (orange,* (b)*) and third (pink,* (b)*) clusters. MT-PGA also introduces* Persistence Correlation Views (d) *which enable the visual identification of the features which are the most responsible for the variability in the ensemble (high correlation, near the disk boundary,* (d)*) as well as their direct inspection in the data (matching colors* (a)*).*

4.1 Context

A series of recent works focused on the notion of *average* for persistence diagrams [TMMH14, LCO18, VBT20, YWM+19b] and we explored this notion for merge trees in Chapter 3, with applications to ensemble summarization and clustering. However, while such averages synthesize a topological descriptor which is well *representative* of the ensemble, they do not describe the topological variability of the ensemble.

This chapter addresses this issue and goes beyond simple averages by adapting the celebrated framework of Principal Component Analysis (PCA) [Pea01] to ensembles of merge trees. For that, we introduce the novel notion of *"Merge-Tree Principal Geodesic Analysis"* (MT-PGA), which captures the most informative geodesics (i.e. analogs of straight lines on the abstract space of merge trees) given the input ensemble, hence facilitating variability analysis and visualization. In particular, we formalize the computation of an orthogonal basis of principal geodesics in the Wasserstein metric space of merge trees (Chapter 3) as a constrained optimization problem (Section 4.2), inspired by previous work on the optimal transport of histograms [SC15, CSB+18], which we extend and specialize to merge trees. We introduce an efficient iterative algorithm (Section 4.3), which exploits an analytic expression of the energy gradient to ensure fast iterations. Moreover, we document accelerations with shared-memory parallelism. Extensive experiments (Section 4.5) indicate that our algorithm produces bases of acceptable reconstruction quality within minutes, for real-life ensembles extracted from public benchmarks. We illustrate the utility of our contribution in two applications. First, we show that the principal geodesic bases computed by our algorithm can result in an important compression of ensembles of merge trees, while still enabling a successful post-processing for typical visualization tasks such as feature tracking or ensemble clustering. Second, we present an extended application of our work to dimensionality reduction, for the visual inspection of the ensemble variability via two-dimensional embeddings, where we show that the views generated by our approach preserve well the intrinsic metric between merge trees, as well as the global structure of the input ensembles. Since our framework is based on the Wasserstein distance between merge trees (Chapter 3), which generalizes the Wasserstein distance between persistence diagrams [TMMH14], it trivially extends to persistence diagrams by simply adjusting a parameter.

4.1.1 Related Work

We refer to Section 3.1.1 for an overview of the literature regarding uncertainty visualization, ensemble visualization and topological methods. Although the topology-based methods presented in Section 3.1.1 addressed the visualization of ensembles of topological objects (e.g. by presenting a representative descriptor, such as a barycenter), they did not focus explicitly on the statistical analysis of the feature variability among the ensemble of descriptors.

Principal Component Analysis (PCA) [Pea01] is a classical approach for the analysis of variability in vectorized data (i.e. point clouds in Euclidean spaces). Extensions have been investigated for metric spaces [GKWZ08, GZ09, FLPJ04], including transport based distances [Cut13] between histograms [SC15, CSB+18]. However, these methods are not directly applicable to merge trees. They focus on fundamentally different objects (histograms). Thus, their distances, geodesics and barycenters are defined differently (in particular in an entropic form [Cut13, CD14]) and the algorithms for their computations are drastically different (based on Sinkhorn matrix scaling [Sin67]). Our global strategy (also based, at a high level, on an alternation of fitting and constraint enforcement, Section 4.2) can be interpreted as an extension of this line of work, but we revisit it completely, to specialize it to merge trees.

As detailed in Section 4.2, the development of a computational framework for Principal Geodesic Analysis (PGA) over an ensemble of topological objects requires several key, low-level, geometrical ingredients, namely *(i)* a distance metric (to measure distances between objects), *(ii)* a geodesic estimation routine (to model the axes of the PGA basis) and *(iii)* a barycenter estimation routine (to compute the origin of the PGA basis). We review now the previous work related to these aspects.

Distance metrics have been studied for most of the above topological objects. Inspired by the literature in optimal transport [Kan42, Mon81], the Wasserstein distance between persistence diagrams [EH09] (Section 2.4.1) has been extensively studied. It is based on a bipartite assignment problem, for which exact [Mun57] and approximate [Ber81, KMN17] implementations are publicly available [TFL+17]. However, the persistence diagram can lack specificity in its data characterization and more advanced topological descriptors, such as merge trees (Section 2.5), are often reported to differentiate datasets better ([MBW14, BYM+14, SMKN20] and Chapter 3). Several similarity measures have been introduced for

Reeb graphs [HSKK01] and their variants [SSW14]. However, since these measures are not metrics (the preservation of the triangle inequality is not specifically enforced), they are not conducive to the computation of geodesics. Stable distance metrics between Reeb graphs [BGW14] and merge trees [MBW14] have been studied from a theoretical point of view [BCLM21] but their computation, following an exponential time complexity, is not tractable for practical datasets in general. Distances with polynomial time computation algorithms have also been investigated. Beketayev et al. [BYM+14] focus on the *branch decomposition tree* (BDT, Section 2.5), and estimate their distances by iteratively reducing a target mismatch term over a significantly large search space. Sridharamurthy et al. [SMKN20] specialize efficient algorithms for computing constrained edit distances between trees [Zha96] to the special case of merge trees, resulting in a distance which is computable for real-life datasets and with acceptable practical stability. Based on this edit distance, we have introduced a generalization of the L_2-Wasserstein distance between persistence diagrams [TMMH14] for merge trees in Chapter 3, thereby enabling the efficient computation of distances, geodesics and barycenters of merge trees.

Regarding the estimation of a *representative* object from a set of topological representations, multiple approaches emerged recently. Several methods [TMMH14, LCO18, VBT20] have been introduced for the estimation of barycenters of persistence diagrams (or vectorized variants [ACE+17, Bub15]). A recent work [YWM+19b] introduced a framework for computing a *1-center* of a set of merge trees (i.e. minimizing its maximum distance to the set), for an interleaving distance [GMO+19]. However, this approach requires pre-existing, reliable correspondence labels between the nodes of the input trees, which is not practical for real-life datasets (heuristics need to be considered). Also, the resulting representative merge tree is not a barycenter (it does not minimize a Fréchet energy, i.e. a sum of distances to the set). Thus, it cannot be used directly for PGA. In contrast, the barycentric framework of Chapter 3 automatically minimizes a Fréchet energy explicitly. Thus, the resulting barycenter merge tree can be directly used for PGA. Another line of approaches aimed at directly applying the classical PCA (or variants from the matrix sketching literature [Woo14]) to *vectorizations* of topological descriptors [RT16, AVRT16, LPW21], i.e. by first converting each topological descriptor into a (high-dimensional) Euclidean vector and then leveraging traditional tools from linear algebra (e.g. classical PCA) on these vectors. However, vectorizations are in general subject to a number of limitations. They

are prone to approximation errors (due to quantization and linearization artifacts), they can be difficult to revert (which challenges their usage for visualization applications) and their stability is not always established. In contrast, our approach *directly* manipulates merge trees in the Wasserstein metric space, and not linear approximations in a Euclidean space. This results in a faithful formulation of merge tree PGA, which ensures improved accuracy and interpretability (cf. experiments, Section 4.5.2).

4.1.2 Contributions

This chapter makes the following new contributions:

1. *An approach to Principal Geodesic Analysis of Merge Trees (MT-PGA):* We formulate the definition of an orthogonal basis of principal geodesics in the Wasserstein metric space of merge trees (Chapter 3) as a constrained optimization problem. Our formulation (Section 4.2) extends previous work on histograms [SC15, CSB+18] and specializes it to merge trees.

2. *An optimization algorithm for MT-PGA:* We introduce an efficient optimization algorithm (Section 4.3) for MT-PGA. Each iteration alternates between *(i)* a minimization of the fitting energy of the MT-PGA basis and *(ii)* a constraint enforcement. Our algorithm exploits an analytic expression of the fitting energy gradient, to ensure fast iterations. We document accelerations based on shared-memory parallelism and report running times in the orders of minutes for real-life ensembles.

3. *An application to data reduction:* We present an application to data reduction (Section 4.4.1), where the merge trees of the input ensemble are significantly compressed, by solely storing the MT-PGA basis and the coordinates of the input merge trees in the basis. We illustrate the utility of this reduction with applications to feature tracking and ensemble clustering.

4. *An application to dimensionality reduction:* We present an application to dimensionality reduction (Section 4.4.2), by embedding each merge tree as a point in a planar view, based on its first two coordinates in the MT-PGA basis. We also contribute derived visualizations – the *Principal Geodesic Surface* and the *Persistence Correlation View* – which enable the visual inspection of the individual features which are the most responsible for the variability in the ensemble.

5. *Implementation:* We provide a C++ implementation of our algorithms that can be used for reproduction purposes: https://github.com/MatPont/MT-PGA

4.2 Formulation

This section describes our novel extension of the Principal Component Analysis (PCA) framework to the Wasserstein metric space of merge trees, leading to the new notion of merge tree Principal Geodesic Analysis (MT-PGA). This section is structured as follows. First, we describe a geometric interpretation of PCA (Section 4.2.1), which our approach extends. Next, we describe how to generalize each low-level geometrical tool used in PCA (Section 4.2.1) to the Wasserstein metric space of merge trees $\mathbb{B}$ (Section 4.2.2). Finally, once these tools are available, we formalize our notion of MT-PGA as a constrained optimization problem (Section 4.2.4), for which we provide an algorithm in Section 4.3 (an overview of the algorithm is given in Section 4.3.1).

4.2.1 Geometric Interpretation of PCA

Principal Component Analysis (PCA) [Pea01] can be described with several, different, yet equivalent, formulations. In the following, we focus on a geometric interpretation (Figure 4.2) which simplifies the transition to merge trees. Given a set of points $P = \{p_1, p_2, \ldots, p_N\}$ in a Euclidean space $\mathbb{R}^d$, PCA defines an orthogonal basis $B_{\mathbb{R}^d} = \{\vec{b_1}, \vec{b_2}, \ldots, \vec{b_d}\}$ of vectors $\vec{b_i} \in \mathbb{R}^d$ ($i \in \{1, \ldots, d\}$), with origin $o_b \in \mathbb{R}^d$, such that:

1. the origin o_b coincides with the arithmetic mean of P,
2. the line l_i (defined by o_b and $\vec{b_i}$) is orthogonal to all previous lines $l_{i'}$ ($i' \in \{1, \ldots, i-1\}$) and minimizes its average squared distance to P.

Let $\Delta_i(p_j)$ be the orthogonal projection of the point p_j on l_i (i.e. $\Delta_i(p_j)$ is the closest point to p_j on l_i, Figure 4.2). It can be expressed as a displacement from o_b on l_i: $\Delta_i(p_j) = o_b + \alpha_i^j \vec{b_i}$, with $\alpha_i^j \in [-1, 1]$. Then, $B_{\mathbb{R}^d}$ can be formulated as an orthogonal basis which minimizes the following data fitting energy (with $d' = d$ in general):

$$E_{L_2}(B_{\mathbb{R}^d}) = \sum_{j=1}^{N} ||p_j - \left(o_b + \sum_{i=1}^{d'} \alpha_i^j \vec{b_i}\right)||_2^2. \tag{4.1}$$

In practice, $B_{\mathbb{R}^d}$ can be optimized iteratively, one dimension at a time (starting with $d' = 1$ and finishing with $d' = d$), by finding at each iteration

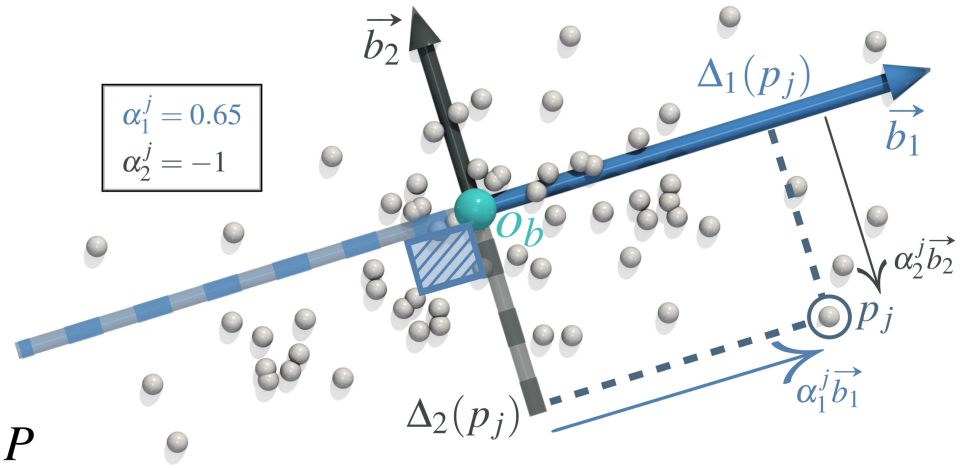


Figure 4.2 – *The classical Principal Component Analysis (PCA) of a point cloud P in $\mathbb{R}^d$ (white spheres) can be computed by iteratively optimizing the fitting to P of orthogonal directions (blue axis first, then black axis).*

a vector $\vec{b_{d'}}$ which is orthogonal to all the previous vectors in $B_{\mathbb{R}^d}$ ($\vec{b_{d'}} \cdot \vec{b_{d''}} = 0, \forall d'' \in \{1, \ldots, d'-1\}$) and which minimizes E_{L_2} (Equation 4.1). After all dimensions have been processed ($d' = d$), $B_{\mathbb{R}^d}$ provides a new coordinate system for P (composed of the vectors $\vec{b_i}$ and the coordinates α_i^j for each point p_j), such that each direction of $B_{\mathbb{R}^d}$ successively provides an optimal fit to P (Equation 4.1). Note that, by construction, the variance of the projected data (i.e. the variance of $\Delta_{d'}(p_j)$ along $l_{d'}$) will be maximized for the first direction ($d' = 1$) and it will keep on decreasing for increasing d'. This motivates early terminations of the optimization (for $d' < d$), as the most *informative* directions are identified in the first iterations of the algorithm.

4.2.2 From PCA to MT-PGA

When the input data is not given as a point cloud in a Euclidean space (Section 4.2.1) but as an abstract set equipped with a metric, the above formulation of Principal Component Analysis (PCA) needs to be extended. Such an extension can be done with the more general notion of Principal Geodesic Analysis (PGA), which needs itself to be specifically instantiated given the specific metric space under study [FLPJ04, SC15, CSB+18]. This instantiation consists in redefining the low-level geometrical tools used in PCA , but within the considered metric space. For instance, *geodesics* (length-minimizing paths) between merge trees (Section 3.4) extend to $\mathbb{B}$ the notion of straight lines, and the Fréchet mean (Section 3.5) extends the arithmetic mean. In this section, we specifically formulate such an extension for merge trees (MTs). In particular, we formalize the following

low-level geometrical notions for the Wasserstein metric space of merge trees $\mathbb{B}$ (see Section 4.2.3 for detailed illustrations):

i. BDT geodesic dot product;

ii. Orthogonal BDT geodesics;

iii. Collinear BDT geodesics;

iv. BDT geodesic axis;

v. BDT geodesic axis arc-length parametrization;

vi. BDT geodesic axis projection;

vii. BDT geodesic axis translation;

viii. BDT geodesic orthogonal basis.

The above notions derive sequentially from one another, to eventually result in the concept of *orthogonal bases of BDT geodesic axes*, which is precisely the main variable of the constrained optimization problem for MT-PGA formulation (Section 4.2.4).

***(i)* BDT geodesic dot product.** A *geodesic* on the Wasserstein metric space of merge trees $\mathbb{B}$ is the shortest path $\vec{\mathcal{G}}(\mathcal{E}, \mathcal{E}')$ between its two extremity BDTs $\mathcal{E}$ and $\mathcal{E}'$. It is an optimal assignment with regard to Equation 3.4, which can be represented as a vector in $\mathbb{R}^{2\times|\mathcal{E}|}$ obtained by concatenating the $|\mathcal{E}|$ 2D vectors (Figure 4.4b) representing the optimal assignment $\phi' \in \Phi'$ in the 2D birth/death space. Then, given two geodesics $\vec{\mathcal{G}}(\mathcal{E}, \mathcal{E}') \in \mathbb{R}^{2\times|\mathcal{E}|}$ and $\vec{\mathcal{G}}(\mathcal{E}, \mathcal{E}'') \in \mathbb{R}^{2\times|\mathcal{E}|}$ sharing an extremity $\mathcal{E}$, their *dot product*, noted $\vec{\mathcal{G}}(\mathcal{E}, \mathcal{E}') \cdot \vec{\mathcal{G}}(\mathcal{E}, \mathcal{E}'')$ can be naturally introduced by considering their Cartesian dot product (i.e. sum of component-wise products between two vectors in $\mathbb{R}^{2\times|\mathcal{E}|}$). Note that these two vectors must be consistently parametrized with regard to their common extremity $\mathcal{E}$: for both vectors, the i^{th} entry represents a 2D vector in the birth/death space modeling the optimal assignment (with regard to Equation 3.4) of the i^{th} point of $\mathcal{E}$ to points of $\mathcal{E}'$ and $\mathcal{E}''$ (small arrows, Figure 4.4d).

***(ii)* Orthogonal BDT geodesics.** Two geodesics $\vec{\mathcal{G}}(\mathcal{E}, \mathcal{E}')$ and $\vec{\mathcal{G}}(\mathcal{E}, \mathcal{E}'')$ are *orthogonal* if $\vec{\mathcal{G}}(\mathcal{E}, \mathcal{E}') \cdot \vec{\mathcal{G}}(\mathcal{E}, \mathcal{E}'') = 0$.

***(iii)* Collinear BDT geodesics.** Two geodesics $\vec{\mathcal{G}}(\mathcal{E}, \mathcal{E}')$ and $\vec{\mathcal{G}}(\mathcal{E}, \mathcal{E}'')$ are

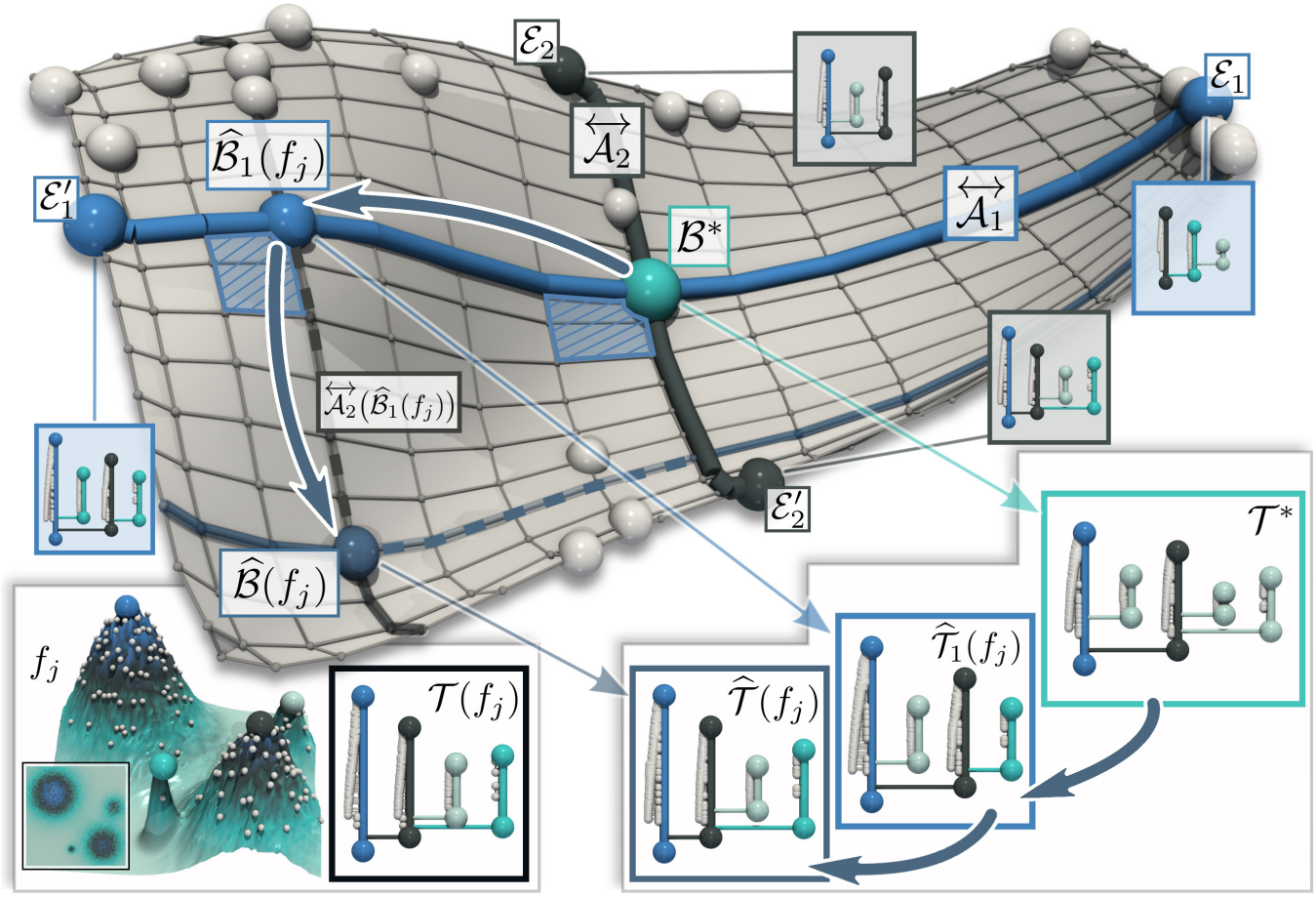


Figure 4.3 – *Approach overview: Our merge tree Principal Geodesic Analysis (MT-PGA) defines a basis $B_{\mathbb{B}}$ on the Wasserstein metric space of BDTs $\mathbb{B}$ (grey surface). It is computed by iteratively optimizing a fitting energy (Equation 4.5) to an input ensemble of BDTs (white spheres) for orthogonal* geodesic axes *(blue: $\overleftrightarrow{\mathcal{A}_1}$, black: $\overleftrightarrow{\mathcal{A}_2}$). Each axis (e.g. $\overleftrightarrow{\mathcal{A}_1}$) is defined as a pair of* geodesics *between its extremities ($\mathcal{E}_1$ and $\mathcal{E}'_1$) and the Fréchet mean of the ensemble (cyan: $\mathcal{B}^*$). After the optimization of the MT-PGA basis, any input BDT $\mathcal{B}(f_j)$ can be reconstructed into $\widehat{\mathcal{B}}(f_j)$ (dark blue sphere) by successive geodesic displacements (dark blue arrows, Equation 4.13) along* translations *of the axes (e.g. $\overleftrightarrow{\mathcal{A}_2}(\widehat{\mathcal{B}}_1(f_j))$, black dashed curve), given the coordinates α^j of $\mathcal{B}(f_j)$ in the basis. While our framework manipulates BDTs, each of them can be directly inverted (transparent arrows) back to MTs (insets), resulting in reconstructions ($\widehat{\mathcal{T}}(f_j)$) that are visually similar to the input ($\mathcal{T}(f_j)$).*

collinear if $\vec{\mathcal{G}}(\mathcal{E},\mathcal{E}') = \lambda\vec{\mathcal{G}}(\mathcal{E},\mathcal{E}'')$, with $\lambda \in \mathbb{R}$. In particular, they are *positively collinear* if $\lambda > 0$, and *negatively collinear* if $\lambda < 0$.

***(iv)* BDT geodesic axis.** Now that the notion of collinear geodesics is available, we proceed to the introduction of the concept of *geodesic axis*. Given an *origin* BDT $\mathcal{E}_O$ and two extremities $\mathcal{E}_i$ and $\mathcal{E}'_i$, a *geodesic axis*, noted $\overleftrightarrow{\mathcal{A}_i}$, is defined as a pair of negatively collinear geodesics $\vec{\mathcal{G}}_i = \vec{\mathcal{G}}(\mathcal{E}_O,\mathcal{E}_i)$, and $\vec{\mathcal{G}}'_i = \vec{\mathcal{G}}(\mathcal{E}_O,\mathcal{E}'_i)$. It follows that a geodesic axis $\overleftrightarrow{\mathcal{A}_i}$ is itself a geodesic between $\mathcal{E}'_i$ and $\mathcal{E}_i$, which is guaranteed to pass through a given origin $\mathcal{E}_O$. An example of geodesic axis is given in Figure 4.3 with the axis $\overleftrightarrow{\mathcal{A}_1}$

(represented as a blue curve) between the BDTs $\mathcal{E}'_1$ and $\mathcal{E}_1$, with origin $\mathcal{B}^*$. This notion will be instrumental in the definition of an orthogonal basis in $\mathbb{B}$, to constrain multiple geodesics to indeed pass through the origin of the basis, while still allowing for an optimization of their extremities (Section 4.3). Each geodesic axis $\overleftrightarrow{\mathcal{A}_i}$ is associated to a *direction vector*: $\vec{\mathcal{V}_i} = \vec{\mathcal{G}_i} - \vec{\mathcal{G}'_i} = \vec{\mathcal{G}}(\mathcal{E}'_i, \mathcal{E}_i)$.

***(v)* BDT geodesic axis arc-length parametrization.** By construction, any BDT $\mathcal{B}$ located on an axis $\overleftrightarrow{\mathcal{A}_i}$ can be expressed with the following arc-length parametrization $\vec{\mathcal{A}_i}(\mathcal{B})$ (with $\alpha_i \in [0,1]$):

$$\mathcal{B} = \mathcal{E}_O + \vec{\mathcal{A}_i}(\mathcal{B}) = \mathcal{E}_O + \alpha_i \times \vec{\mathcal{G}_i} + (1-\alpha_i) \times \vec{\mathcal{G}'_i}. \tag{4.2}$$

This arc-length parametrization is the analog for the metric space $\mathbb{B}$ of the Euclidean line parametrization used in PCA (Equation 4.1).

***(vi)* BDT geodesic axis projection.** The *projection* $\mathcal{B}_{\overleftrightarrow{\mathcal{A}_i}}$ of an arbitrary BDT $\mathcal{B}$ on the axis $\overleftrightarrow{\mathcal{A}_i}$ is its closest BDT on $\overleftrightarrow{\mathcal{A}_i}$. It minimizes the following projection energy:

$$\begin{aligned} \mathcal{B}_{\overleftrightarrow{\mathcal{A}_i}} &= \underset{\mathcal{B}' \in \overleftrightarrow{\mathcal{A}_i}}{\arg\min} \left(W_2^{\mathcal{T}}(\mathcal{B}, \mathcal{B}') \right) \\ &= \underset{\mathcal{B}' \in \overleftrightarrow{\mathcal{A}_i}}{\arg\min} \left(W_2^{\mathcal{T}}(\mathcal{B}, \mathcal{E}_O + \vec{\mathcal{A}_i}(\mathcal{B}')) \right). \end{aligned} \tag{4.3}$$

An example of projection is given in Figure 4.3, with the blue sphere noted $\widehat{\mathcal{B}}_1(f_j)$, which is the projection on the axis $\overleftrightarrow{\mathcal{A}_1}$ (blue curve) of an input BDT $\mathcal{B}(f_j)$. The projection $\mathcal{B}_{\overleftrightarrow{\mathcal{A}_i}}$ of $\mathcal{B}$ on $\mathcal{A}_i$ will act as the analog, for the metric space $\mathbb{B}$, of the projection $\Delta_i(p_j)$ of a point p_j along a line l_i in the Euclidean PCA (Section 4.2.1).

***(vii)* BDT geodesic axis translation.** Let $\overleftrightarrow{\mathcal{A}_i}$ and $\overleftrightarrow{\mathcal{A}_j}$ be two axes sharing the same origin $\mathcal{E}_O$. The axis $\overleftrightarrow{\mathcal{A}_j}(\mathcal{B})$ is called the *translation* of $\overleftrightarrow{\mathcal{A}_j}$ along $\overleftrightarrow{\mathcal{A}_i}$, with origin $\mathcal{B} \in \overleftrightarrow{\mathcal{A}_i}$ if: $\overleftrightarrow{\mathcal{A}_j}(\mathcal{B}) = \big((\mathcal{B}, \mathcal{B} + \vec{\mathcal{G}_j}), (\mathcal{B}, \mathcal{B} + \vec{\mathcal{G}'_j})\big)$. Intuitively, a translated axis $\overleftrightarrow{\mathcal{A}_j}(\mathcal{B})$ is a BDT axis, which is parallel to the axis $\overleftrightarrow{\mathcal{A}_j}$, and which passes through a specific BDT $\mathcal{B}$. In Figure 4.3, an example is given with the axis $\overleftrightarrow{\mathcal{A}_2}(\widehat{\mathcal{B}}_1(f_j))$ (black dashed curve), which is the translation of the axis $\overleftrightarrow{\mathcal{A}_2}$ (black curve) at the BDT $\widehat{\mathcal{B}}_1(f_j)$ (blue sphere). This notion of translated axis is needed when reconstructing a BDT given its coordinates in the MT-PGA basis, as detailed next.

***(viii)* BDT geodesic orthogonal basis.** We now introduce the notion of

orthogonal basis in the metric space of merge trees $\mathbb{B}$, which will be the variable at the core of the constrained optimization problem for MT-PGA formulation (Section 4.2.4). Let $B_{\mathbb{B}} = \{\overleftrightarrow{\mathcal{A}_1}, \overleftrightarrow{\mathcal{A}_2}, \ldots, \overleftrightarrow{\mathcal{A}_{d'}}\}$ be an *orthogonal basis* of d' geodesic axes (i.e. with pairwise orthogonal direction vectors $\{\overrightarrow{\mathcal{V}_1}, \overrightarrow{\mathcal{V}_2}, \ldots, \overrightarrow{\mathcal{V}_{d'}}\}$), with origin $\mathcal{E}_O$. Let $\widehat{\mathcal{B}}_1$ be the projection of an arbitrary BDT $\mathcal{B}$ on $\overleftrightarrow{\mathcal{A}_1}$ ($\widehat{\mathcal{B}}_1 = \mathcal{B}_{\overleftrightarrow{\mathcal{A}_1}} = \mathcal{B}_{\overleftrightarrow{\mathcal{A}_1(\mathcal{E}_O)}}$, e.g. blue sphere on $\overleftrightarrow{\mathcal{A}_1}$, Figure 4.3, noted $\widehat{\mathcal{B}}_1(f_j)$). Now, let $\widehat{\mathcal{B}}_2$ be the projection of $\mathcal{B}$ on the *translation* of $\overleftrightarrow{\mathcal{A}_2}$ along $\overleftrightarrow{\mathcal{A}_1}$, with origin $\widehat{\mathcal{B}}_1$: $\widehat{\mathcal{B}}_2$ is the BDT of the *translated* axis $\overleftrightarrow{\mathcal{A}_2}(\widehat{\mathcal{B}}_1)$ which is the closest to $\mathcal{B}$ (i.e. $\widehat{\mathcal{B}}_2 = \mathcal{B}_{\overleftrightarrow{\mathcal{A}_2(\widehat{\mathcal{B}}_1)}}$, e.g. dark blue sphere on $\overleftrightarrow{\mathcal{A}_2}(\widehat{\mathcal{B}}_1)$ in Figure 4.3, noted $\widehat{\mathcal{B}}(f_j)$).

By recursion, we can then define the *translated projection* of $\mathcal{B}$ for the axis $\overleftrightarrow{\mathcal{A}_{d'}}$, noted $\widehat{\mathcal{B}}_{d'}$, as $\widehat{\mathcal{B}}_{d'} = \mathcal{B}_{\overleftrightarrow{\mathcal{A}_{d'}(\widehat{\mathcal{B}}_{d'-1})}}$ (with $\widehat{\mathcal{B}}_0 = \mathcal{E}_O$). Intuitively, the translated projection $\widehat{\mathcal{B}}_{d'}$ of $\mathcal{B}$ is the closest BDT to $\mathcal{B}$ (dark blue sphere in Figure 4.3), which can be obtained by a sequence of d' geodesic displacements (dark blue arrows in Figure 4.3) along the translated axes of the orthogonal basis $B_{\mathbb{B}}$.

By using Equation 4.2, $\widehat{\mathcal{B}}_{d'}$ can be associated with a collection of arc-length parameterizations $\{\overrightarrow{\mathcal{A}_1}(\widehat{\mathcal{B}}_1), \overrightarrow{\mathcal{A}_2}(\widehat{\mathcal{B}}_2), \ldots, \overrightarrow{\mathcal{A}_{d'}}(\widehat{\mathcal{B}}_{d'})\}$, s.t.:

$$\widehat{\mathcal{B}}_{d'} = \mathcal{E}_O + \sum_{i=1}^{d'} \overrightarrow{\mathcal{A}_i}(\widehat{\mathcal{B}}_i). \tag{4.4}$$

$\widehat{\mathcal{B}}_{d'}$ can then be interpreted as the *reconstruction* of $\mathcal{B}$ (up to the dimension d'), given the basis $B_{\mathbb{B}}$: it is obtained by starting from the origin $\mathcal{E}_O$ of the basis, and successively minimizing the projection energy (Equation 5.13) along translated axes of $B_{\mathbb{B}}$, yielding a coordinate vector $\alpha \in [0,1]^{d'}$ (Equation 4.2) which can be interpreted as the coordinates of $\mathcal{B}$ in $B_{\mathbb{B}}$.

4.2.3 Concept Illustrations

We now present an illustration, on a toy example, of the low-level geometrical notions formalized in our approach.

Figure 4.4(a): a BDT geodesic orthogonal basis with its origin $\mathcal{B}^*$ (cyan sphere), its BDT geodesic axis $\overleftrightarrow{\mathcal{A}_1}$ (blue, with its extremities $\mathcal{E}_1$ and $\mathcal{E}'_1$ also in blue) and its axis $\overleftrightarrow{\mathcal{A}_2}$ (black, with its extremities $\mathcal{E}_2$ and $\mathcal{E}'_2$ also in black). Given an input BDT $\mathcal{B}(f_j)$, its *translated projection* $\widehat{\mathcal{B}}_1(f_j)$ along $\overleftrightarrow{\mathcal{A}_1}$ is shown with a light blue sphere (on $\overleftrightarrow{\mathcal{A}_1}$). The translated axis of $\overleftrightarrow{\mathcal{A}_2}$ at $\widehat{\mathcal{B}}_1(f_j)$, noted $\overleftrightarrow{\mathcal{A}_2}(\widehat{\mathcal{B}}_1(f_j))$, is shown with a black transparent curve. Finally, the reconstruction of $\mathcal{B}(f_j)$ by this two-dimensional MT-PGA basis is given by the grey sphere $\widehat{\mathcal{B}}(f_j)$.

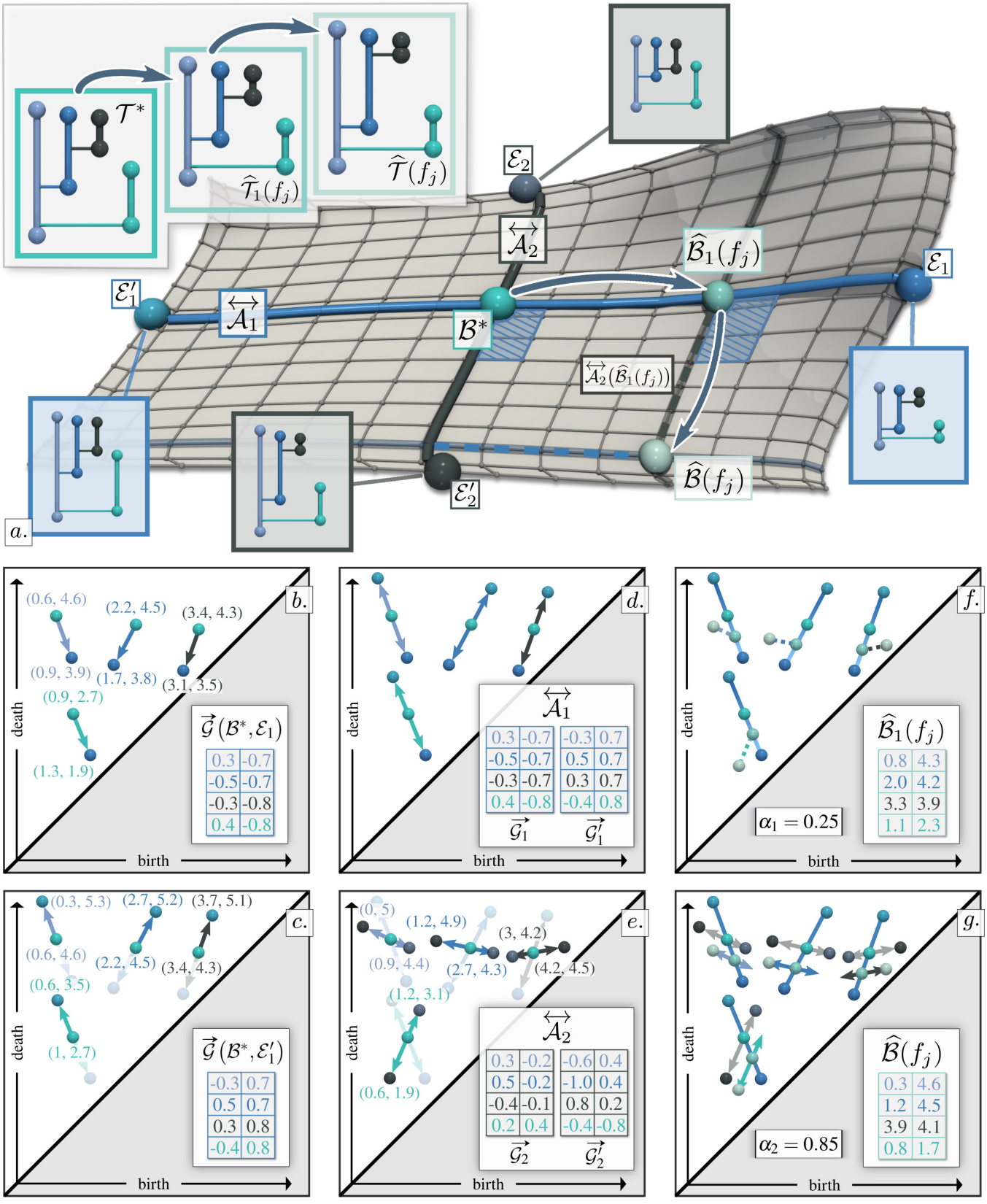


Figure 4.4 – *Illustration on a toy example of the geometrical notions formalized in our approach. We refer the reader to Section 4.2.3 for a detailed description.*

All these concepts are illustrated in the birth-death space in the figures (b) to (g). In these figures, $\mathcal{B}^*$ is represented by the cyan point cloud (origin of the basis, here: $|\mathcal{B}^*| = 4$). Each $(2 \times |\mathcal{B}^*|)$-dimensional geodesic vector is represented in the birth-death space by a set of $|\mathcal{B}^*|$ 2D arrows, modeling the individual branch-wise assignments (colored according to the branch in $\mathcal{B}^*$ they start from, see the inset merge tree, top left corner of (a)). The coordinates of the arrows are reported, with matching colors, in the adjacent array (the numbers report the actual values, but rounded to one decimal after the point).

Figure 4.4(b): The geodesic $\vec{\mathcal{G}}(\mathcal{B}^*, \mathcal{E}_1)$ is reported with colored arrows as described above.

Figure 4.4(c): The geodesic $\vec{\mathcal{G}}(\mathcal{B}^*, \mathcal{E}_1')$ is illustrated similarly. $\vec{\mathcal{G}}(\mathcal{B}^*, \mathcal{E}_1)$ is reported in transparent, to illustrate the fact that the *global* collinearity of two $(2 \times |\mathcal{B}^*|)$-dimensional geodesic vectors translates into a *local* collinearity of the individual 2D assignment vectors in the birth-death space.

Figure 4.4(d): The geodesic axis $\overleftrightarrow{\mathcal{A}_1}$ is reported with colored double arrows, along with their coordinates (arrays).

Figure 4.4(e): The geodesic axis $\overleftrightarrow{\mathcal{A}_2}$ is shown similarly. $\overleftrightarrow{\mathcal{A}_1}$ is reported in transparent, to illustrate the fact that the *global* orthogonality between the $(2 \times |\mathcal{B}^*|)$-dimensional direction vectors $\vec{\mathcal{V}_1} = \vec{\mathcal{G}_1} - \vec{\mathcal{G}_1'}$ and $\vec{\mathcal{V}_2} = \vec{\mathcal{G}_2} - \vec{\mathcal{G}_2'}$ does not necessarily translate into a *local* orthogonality between the individual 2D assignment vectors in the birth-death space. This can be explained by the fact that a 2D space can only capture orthogonality between 2 directions, not $2 \times |\mathcal{B}^*|$. Alternatively, the geodesic axes $\overleftrightarrow{\mathcal{A}_1}$ and $\overleftrightarrow{\mathcal{A}_2}$, when shown in the birth-death space, can be interpreted as 2-dimensional projections of orthogonal $(2 \times |\mathcal{B}^*|)$-dimensional direction vectors.

Figure 4.4(f): The projection $\widehat{\mathcal{B}}_1(f_j)$ (light blue sphere on the segments) is the BDT on $\overleftrightarrow{\mathcal{A}_1}$ which minimizes its distance to $\mathcal{B}(f_j)$.

Figure 4.4(g): The translated axis $\overleftrightarrow{\mathcal{A}_2}(\widehat{\mathcal{B}}_1(f_j))$ is shown with colored double arrows. The axis $\overleftrightarrow{\mathcal{A}_2}$ is shown in transparent (black).

4.2.4 MT-PGA Formulation

Now that the above low-level geometrical tools have been introduced for the Wasserstein metric space $\mathbb{B}$, we can formulate the MT-PGA by direct analogy to the classical PCA (Section 4.2.1).

Given a set $\mathcal{S}_\mathcal{B} = \{\mathcal{B}(f_1), \ldots, \mathcal{B}(f_N)\}$ of input BDTs, let $\mathcal{B}^*$ be their Fréchet mean (Equation 3.24). Then, similarly to PCA, MT-PGA defines

an orthogonal basis $B_{\mathbb{B}} = \{\overleftrightarrow{\mathcal{A}_1}, \overleftrightarrow{\mathcal{A}_2}, \ldots, \overleftrightarrow{\mathcal{A}_{2\times|\mathcal{B}^*|}}\}$ of geodesic axes $\overleftrightarrow{\mathcal{A}_i}$ ($i \in \{1, \ldots, 2 \times |\mathcal{B}^*|\}$) s.t.:

1. the origin $\mathcal{E}_O$ of $B_{\mathbb{B}}$ coincides with the Fréchet mean $\mathcal{B}^*$ of $\mathcal{S}_{\mathcal{B}}$,
2. the axis $\overleftrightarrow{\mathcal{A}_i}$ is orthogonal to all axes $\overleftrightarrow{\mathcal{A}_{i'}}$ ($i' \in \{1, \ldots, i-1\}$) and it minimizes its average squared Wasserstein distance to $\mathcal{S}_{\mathcal{B}}$.

Then, $B_{\mathbb{B}}$ can be formulated as an orthogonal basis which minimizes the following (non-convex) fitting energy (with $d' = 2 \times |\mathcal{B}^*|$ in general):

$$E_{W_2^{\mathcal{T}}}(B_{\mathbb{B}}) = \sum_{j=1}^{N} W_2^{\mathcal{T}}\left(\mathcal{B}(f_j), \mathcal{B}^* + \sum_{i=1}^{d'} \overrightarrow{\mathcal{A}_i}(\widehat{\mathcal{B}}_i(f_j))\right)^2. \tag{4.5}$$

The above equation is a direct analogy to the classical PCA (Equation 4.1): the L_2 norm is replaced by the Wasserstein distance $W_2^{\mathcal{T}}$, the arithmetic mean o_b by $\mathcal{B}^*$ and the term $\alpha_i^j \overrightarrow{b_i}$ by $\overrightarrow{\mathcal{A}_i}(\widehat{\mathcal{B}}_i(f_j))$.

4.3 Algorithm

This section presents our novel algorithm, based on the constrained minimization of Equation 4.5, for the estimation of an orthogonal basis $B_{\mathbb{B}}$ for the Principal Geodesic Analysis of merge trees.

4.3.1 Overview

Algorithm 1 provides an overview of our approach. Our algorithm takes an ensemble $\mathcal{S}_{\mathcal{B}}$ of BDTs as an input and provides three outputs: *(i)* the basis origin $\mathcal{B}^*$, *(ii)* its d_{max} geodesic axes $\{\overleftrightarrow{\mathcal{A}_1}, \overleftrightarrow{\mathcal{A}_2}, \ldots, \overleftrightarrow{\mathcal{A}_{d_{max}}}\}$ (where $1 \leq d_{max} \leq 2 \times |\mathcal{B}^*|$ is an input parameter) and *(iii)* the coordinates $\alpha^j \in [0,1]^{d_{max}}$ of the input BDTs in $B_{\mathbb{B}}$ (Equation 4.2, with $j \in \{1, 2, \ldots, N\}$).

First, the origin of the basis $B_{\mathbb{B}}$ is computed as the Wasserstein barycenter $\mathcal{B}^*$ of $\mathcal{S}_{\mathcal{B}}$ (line 1). This is done with an iterative algorithm (Chapter 3), which initializes $\mathcal{B}^*$ at the BDT in $\mathcal{S}_{\mathcal{B}}$ minimizing Equation 3.24, and which further minimizes Equation 3.24 by iteratively alternating assignment and update phases (see Chapter 3). After this optimization has completed, only the N_1 most persistent branches are kept in $\mathcal{B}^*$, where N_1 is an input parameter controlling the memory footprint of the basis (see Section 5.3.7).

Next, the MT-PGA basis is computed by adapting the iterative strategy described in the case of PCA (Section 4.2.1). In particular, geodesic axes are computed one dimension at a time (starting with $d' = 1$ and finishing with $d' = d_{max}$, *for* loop, lines 2 to 16).

In particular, at each step d':

1. The geodesics $\overrightarrow{\mathcal{G}_{d'}}$ and $\overrightarrow{\mathcal{G}'_{d'}}$ defining the axis $\overleftrightarrow{\mathcal{A}_{d'}}$ are optimized in order to minimize Equation 4.5 (line 8). This optimization (described in Section 4.3.2 and illustrated in Figure 4.5) involves the translated projection $\widehat{\mathcal{B}}_{d'}(f_j)$ of each input BDT $\mathcal{B}(f_j)$ (line 6), providing an estimation of its coordinate $\alpha^j_{d'}$ (Equation 4.2).
2. The axis $\overleftrightarrow{\mathcal{A}_{d'}}$ is updated into a valid solution (as described in Section 4.3.3 and illustrated in Figure 4.6), by enforcing each constraint one after the other (lines 11 to 13).

This alternated procedure optimization/constraint is iterated until $E_{W_2^{\mathcal{T}}}(B_{\mathbb{B}})$ does not evolve significantly anymore (see Section 4.3.2). Once this is achieved, the current axis $\overleftrightarrow{\mathcal{A}_{d'}}$ is considered as finalized and d' is incremented until $\overleftrightarrow{\mathcal{A}_{d_{max}}}$ is finalized.

Algorithm 1 Merge Tree Principal Geodesic Analysis (MT-PGA)

Input: Set of BDTs $\mathcal{S}_{\mathcal{B}} = \{\mathcal{B}(f_1), \ldots, \mathcal{B}(f_N)\}$.
Output1: Basis origin $\mathcal{B}^*$;
Output2: Basis geodesic axes $B_{\mathbb{B}} = \{\overleftrightarrow{\mathcal{A}_1}, \overleftrightarrow{\mathcal{A}_2}, \ldots, \overleftrightarrow{\mathcal{A}_{d_{max}}}\}$;
Output3: Coordinates $\alpha^j \in [0,1]^{d_{max}}$ of the input BDTs in $B_{\mathbb{B}}$ (with $j \in \{1, 2, \ldots, N\}$).

1: $\mathcal{B}^* \leftarrow$WassersteinBarycenter$(\mathcal{S}_{\mathcal{B}})$;
2: **for** $d' \in \{1, 2, \ldots, d_{max}\}$ **do**
3: $\quad\overleftrightarrow{\mathcal{A}_{d'}} \leftarrow$InitializeGeodesicAxis$(\mathcal{S}_{\mathcal{B}}, \mathcal{B}^*, B_{\mathbb{B}})$;
4: $\quad$**while** $E_{W_2^{\mathcal{T}}}(B_{\mathbb{B}})$ decreases **do**
5: $\quad\quad$// Optimize the current geodesic axis $\overleftrightarrow{\mathcal{A}_{d'}}$ (Section 4.3.2)
6: $\quad\quad\alpha^{j\in\{1,2,\ldots N\}}_{d'} \leftarrow$ProjectTrees$(\mathcal{S}_{\mathcal{B}}, \mathcal{B}^*, B_{\mathbb{B}}, \alpha^{j\in\{1,2,\ldots N\}})$;
7: $\quad\quad E_{W_2^{\mathcal{T}}}(B_{\mathbb{B}}) \leftarrow$EvaluateFittingEnergy$(\mathcal{S}_{\mathcal{B}}, \mathcal{B}^*, B_{\mathbb{B}}, \alpha^{j\in\{1,2,\ldots N\}})$;
8: $\quad\quad\overleftrightarrow{\mathcal{A}_{d'}} \leftarrow$OptimizeFittingEnergy$(\mathcal{S}_{\mathcal{B}}, \mathcal{B}^*, B_{\mathbb{B}}, \alpha^{j\in\{1,2,\ldots N\}})$;
9: $\quad\quad$// Enforce the constaints (Section 4.3.3)
10: $\quad\quad$**while** $\overleftrightarrow{\mathcal{A}_{d'}}$ evolves **do**
11: $\quad\quad\quad\overleftrightarrow{\mathcal{A}_{d'}} \leftarrow$EnforceGeodesics$(\mathcal{B}^*, \overleftrightarrow{\mathcal{A}_{d'}})$;
12: $\quad\quad\quad\overleftrightarrow{\mathcal{A}_{d'}} \leftarrow$EnforceNegativeCollinearity$(\overleftrightarrow{\mathcal{A}_{d'}})$;
13: $\quad\quad\quad\overleftrightarrow{\mathcal{A}_{d'}} \leftarrow$EnforceOrthogonality$(B_{\mathbb{B}}, \overleftrightarrow{\mathcal{A}_{d'}})$;
14: $\quad\quad$**end while**
15: $\quad$**end while**
16: **end for**

4.3.2 Geodesic Axis Optimization

This section describes the optimization of a single axis $\overleftrightarrow{\mathcal{A}_{d'}}$.

First, the geodesic $\overrightarrow{\mathcal{G}_{d'}}$ of $\overleftrightarrow{\mathcal{A}_{d'}}$ is initialized by considering as its extremity $\mathcal{E}_{d'}$ the input BDT in $\mathcal{S}_{\mathcal{B}}$ which maximizes its Wasserstein distance to all the previous geodesic axes $\overleftrightarrow{\mathcal{A}_{d''}}$, $\forall d'' \in \{1, \dots d'-1\}$ (to $\mathcal{B}^*$ when $d'=1$). The second geodesic $\overrightarrow{\mathcal{G}'_{d'}}$ is initialized at $-\overrightarrow{\mathcal{G}_{d'}}$.

Next, both $\overrightarrow{\mathcal{G}_{d'}}$ and $\overrightarrow{\mathcal{G}'_{d'}}$ are optimized, to minimize Equation 4.5. This is achieved with an iterative algorithm (Algorithm 1, *while* loop, line 4) which, similarly to the minimization of the Fréchet energy (Chapter 3), alternates an *(i) Assignment* and an *(ii) Update* phase. Intuitively, the *(i) Assignment* phase will compute geodesics (dashed blue curves, Figure 4.5a) between the input BDTs (white spheres, Figure 4.5a) and the axis to optimize (plain blue curve, Figure 4.5a), while the *(ii) Update* phase will optimize freely the axis under these optimal assignments. After the *(ii) Update*, the initial assignments may no longer be optimal (since the axis has changed) and the *Assignment/Update* sequence needs to be iterated again, as detailed next.

The *Assignment* phase *(i)* computes a geodesic between each input BDT $\mathcal{B}(j)$ and its translated projection $\widehat{\mathcal{B}}_{d'}(f_j)$ (Figure 4.5). In particular, these translated projections (Algorithm 1, line 6) are estimated in practice by sampling the translated axis $\overleftrightarrow{\mathcal{A}_{d'}}(\widehat{\mathcal{B}}_{d'-1}(f_j))$ along N_2 evenly spaced samples and by selecting, for each $\mathcal{B}(f_j)$, the sample $\widehat{\mathcal{B}}_{d'}(f_j)$ which minimizes Equation 5.13.

Next, the *Update* phase *(ii)* consists in optimizing $\overrightarrow{\mathcal{G}_{d'}}$ and $\overrightarrow{\mathcal{G}'_{d'}}$ to minimize Equation 4.5 under the assignments computed by the *Assignment* phase *(i)*. As detailed just after , for fixed assignments, the fitting energy

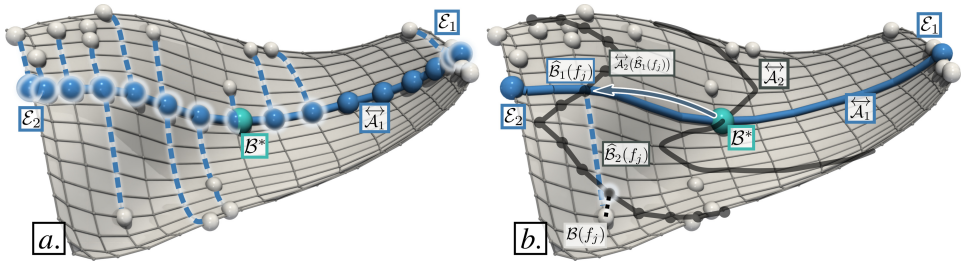


Figure 4.5 – *Computing the* translated projection *of each input BDT* $\mathcal{B}(f_j)$ *(white sphere). For the first axis* (a), N_2 *samples evenly spaced on* $\overleftrightarrow{\mathcal{A}_1}$ *are considered (blue spheres). Then, for each* $\mathcal{B}(f_j)$*, the sample which minimizes its* $W_2^{\mathcal{T}}$ *distance to* $\mathcal{B}(f_j)$ *is selected as* $\widehat{\mathcal{B}}_1(f_j)$ *(dashes). Next* (b)*, the axis* $\overleftrightarrow{\mathcal{A}_2}$ *is* translated *to* $\widehat{\mathcal{B}}_1(f_j)$ *along* $\overleftrightarrow{\mathcal{A}_1}$ *(arrow) yielding the translated axis* $\overleftrightarrow{\mathcal{A}_2}(\widehat{\mathcal{B}}_1(f_j))$*, which is also evenly sampled. The sample of* $\overleftrightarrow{\mathcal{A}_2}(\widehat{\mathcal{B}}_1(f_j))$ *minimizing its* $W_2^{\mathcal{T}}$ *distance to* $\mathcal{B}(f_j)$ *is selected as* $\widehat{\mathcal{B}}_2(f_j)$.

$E_{W_2^{\mathcal{T}}}$ is convex with the variables $\overrightarrow{\mathcal{G}_{d'}}$ and $\overrightarrow{\mathcal{G}'_{d'}}$. We then provide the analytic expression of the gradient of this energy and we derive the analytic expressions of $\overrightarrow{\mathcal{G}_{d'}}$ and $\overrightarrow{\mathcal{G}'_{d'}}$ which minimize Equation 4.5.

Given the Wasserstein barycenter $\mathcal{B}^*$ of the input set of BDTs $\mathcal{S}_{\mathcal{B}} = \{\mathcal{B}(f_1), \ldots, \mathcal{B}(f_N)\}$ as well as an initialization of the d'-th geodesic axis of $B_{\mathbb{B}}$, noted $\overleftrightarrow{\mathcal{A}_{d'}}$, we want to update $\overleftrightarrow{\mathcal{A}_{d'}}$ in order to decrease the fitting energy associated to $B_{\mathbb{B}}$ (Equation 4.5).

Similarly to previous optimization strategies for the Fréchet energy (such as [TMMH14, VBT20] or in Chapter 3), our approach consists in alternating phases of *(i) Assignments* (between the input BDTs and their projections along translations of $\overleftrightarrow{\mathcal{A}_{d'}}$, cf. Equation 5.13) and *(ii) Updates* (of the axis $\overleftrightarrow{\mathcal{A}_{d'}}$).

Thus, at the *Update* phase *(ii)*, the assignment ${\phi'_{d'}}^j$ between each input BDT $\mathcal{B}(f_j)$ and its projection $\widehat{\mathcal{B}}_{d'}(f_j)$ on the the translated axis $\overleftrightarrow{\mathcal{A}_{d'}}\big(\widehat{\mathcal{B}}_{d'-1}(f_j)\big)$ is constant. It follows that each branch $b^* \in \mathcal{B}^*$ can be considered independently for the minimization of Equation 5.13. In particular, let $\big(b^* + \sum_{i=1}^{d'}(1-\alpha_i^j)\overrightarrow{g_i} + \alpha_i^j\overrightarrow{g'_i}\big)$ be the branch assigned to b^* in $\widehat{\mathcal{B}}_{d'}(f_j)$, with $\overrightarrow{g_i}$ and $\overrightarrow{g'_i}$ being two 2D vectors in the birth/death space (i.e. the entries of $\overrightarrow{\mathcal{G}_i}$ and $\overrightarrow{\mathcal{G}'_i}$ corresponding to the branch b^* of $\mathcal{B}^*$). Then, the individual fitting energy associated to b^*, i.e. its contribution to Equation 5.13, can be expressed as:

$$E_{b^*}(\overrightarrow{g_{d'}}, \overrightarrow{g'_{d'}}) = \sum_{j=1}^{N} d_2\big(b_j, b^* + \sum_{i=1}^{d'}(1-\alpha_i^j)\overrightarrow{g_i} + \alpha_i^j\overrightarrow{g'_i}\big)^2, \tag{4.6}$$

where b_j stands for the branch in $\mathcal{B}(f_j)$ assigned to b^* and d_2 stands for the L_2 norm in the 2D birth/death space (i.e. the ground distance involved in $W_2^{\mathcal{T}}$, see Equation 2.2). Note that the usage of the L_2 norm implies that E_{b^*} is convex. Our goal at this stage is to find the two vectors $\overrightarrow{g_{d'}}$ and $\overrightarrow{g'_{d'}}$ which minimize Equation 4.6.

Equation 4.6 can be further detailed as follows, where g_{ix} and g_{iy} stand for the X-Y coordinates of the vector $\overrightarrow{g_i}$ in the birth/death plane:

$$\begin{aligned} E_{b^*}(\overrightarrow{g_{d'}}, \overrightarrow{g'_{d'}}) = \sum_{j=1}^{N} &\Big(b_{jx} - \big(b_x^* + \sum_{i=1}^{d'}(1-\alpha_i^j)\times g_{ix} + \alpha_i^j \times g'_{ix}\big)\Big)^2 \\ &+ \Big(b_{jy} - \big(b_y^* + \sum_{i=1}^{d'}(1-\alpha_i^j)\times g_{iy} + \alpha_i^j \times g'_{iy}\big)\Big)^2. \end{aligned} \tag{4.7}$$

In the following, we derive the gradient of the above convex energy. In particular, we detail the derivation for the X coordinate only (the deriva-

tion along the Y coordinate being identical):

$$\frac{\partial E_{b^*}(\overrightarrow{g_{d'}}, \overrightarrow{g'_{d'}})}{\partial g_{d'x}} = 2\sum_{j=1}^{N}(1-\alpha_{d'}^j)\Big(b_{jx} - \big(b_x^* + \sum_{i=1}^{d'}(1-\alpha_i^j)\times g_{ix} + \alpha_i^j \times g'_{ix}\big)\Big)$$
$$\frac{\partial E_{b^*}(\overrightarrow{g_{d'}}, \overrightarrow{g'_{d'}})}{\partial g'_{d'x}} = 2\sum_{j=1}^{N}\alpha_{d'}^j\Big(b_{jx} - \big(b_x^* + \sum_{i=1}^{d'}(1-\alpha_i^j)\times g_{ix} + \alpha_i^j \times g'_{ix}\big)\Big).$$

To minimize Equation 4.7, we aim to find the values of $g_{d'x}$ and $g'_{d'x}$ for which the above partial derivatives equal zero. This yields the following linear system of two equations (with two unknowns, $g_{d'x}$ and $g'_{d'x}$):

$$\begin{cases} \sum_{j=1}^{N}(1-\alpha_{d'}^j)\Big(b_{jx} - \big(b_x^* + \sum_{i=1}^{d'}(1-\alpha_i^j)\times g_{ix} + \alpha_i^j \times g'_{ix}\big)\Big) = 0 \\ \sum_{j=1}^{N}\alpha_{d'}^j\Big(b_{jx} - \big(b_x^* + \sum_{i=1}^{d'}(1-\alpha_i^j)\times g_{ix} + \alpha_i^j \times g'_{ix}\big)\Big) = 0. \end{cases} \tag{4.8}$$

Given the above system, we aim next at expressing $g_{d'x}$ as a function of $g'_{d'x}$. To simplify notations, we introduce the term $b^*_{d'x}$ as follows:

$$b^*_{d'x} := b_x^* + \sum_{i=1}^{d'-1}(1-\alpha_i^j)\times g_{ix} + \alpha_i^j \times g'_{ix}.$$

Then, the first line of Equation 4.8 can be re-written as:

$$\sum_{j=1}^{N}(1-\alpha_{d'}^j)\Big(b_{jx} - \big(b^*_{d'x} + (1-\alpha_{d'}^j)\times g_{d'x} + \alpha_{d'}^j \times g'_{d'x}\big)\Big) = 0.$$

Then, it follows that:

$$g_{d'x} = \frac{\sum_{j=1}^{N}(1-\alpha_{d'}^j)\Big(b_{jx} - \big(b^*_{d'x} + \alpha_{d'}^j \times g'_{d'x}\big)\Big)}{\sum_{j=1}^{N}(1-\alpha_{d'}^j)^2}. \tag{4.9}$$

Now, we apply the same reasoning with the second line of Equation 4.8, yielding the following expression of $g'_{d'x}$:

$$g'_{d'x} = \frac{\sum_{j=1}^{N}\alpha_{d'}^j\Big(b_{jx} - \big(b^*_{d'x} + (1-\alpha_{d'}^j)\times g_{d'x}\big)\Big)}{\sum_{j=1}^{N}(\alpha_{d'}^j)^2}. \tag{4.10}$$

At this stage, one can notice that the expression of $g'_{d'x}$ is itself a function of $g_{d'x}$. Thus, we insert the expression of $g'_{d'x}$ (Equation 4.10) into that of $g_{d'x}$ (Equation 4.9), which results eventually in the following expression (we omit the detailed, intermediate steps):

$$g_{d'x} = \frac{\sum_{j=1}^{N}(1-\alpha_{d'}^j)\left(b_{jx} - b^*_{d'x} - \alpha_{d'}^j\frac{\sum_{k=1}^{N}\alpha_{d'}^k(b_{kx} - b^*_{d'x})}{\sum_{k=1}^{N}(\alpha_{d'}^k)^2}\right)}{\sum_{j=1}^{N}(1-\alpha_{d'}^j)^2 + \sum_{j=1}^{N}(1-\alpha_{d'}^j)\alpha_{d'}^j\frac{\sum_{k=1}^{N}\alpha_{d'}^k\big(-(1-\alpha_{d'}^k)\big)}{\sum_{k=1}^{N}(\alpha_{d'}^k)^2}}. \tag{4.11}$$

Finally, we can insert the expression of $g_{d'x}$ (Equation 4.9) into the original expression of $g'_{d'x}$ (Equation 4.10), resulting in the following expression (again, we omit the detailed, intermediate steps):

$$g'_{d'x} = \frac{\sum_{j=1}^{N} \alpha_{d'}^{j} \left(b_{jx} - b^{*}_{d'x} - (1 - \alpha_{d'}^{j}) \dfrac{\sum_{k=1}^{N} (1 - \alpha_{d'}^{k})(b_{kx} - b^{*}_{d'x})}{\sum_{k=1}^{N} (1 - \alpha_{d'}^{k})^2} \right)}{\sum_{j=1}^{N} (\alpha_{d'}^{j})^2 + \sum_{j=1}^{N} \alpha_{d'}^{j} (1 - \alpha_{d'}^{j}) \dfrac{\sum_{k=1}^{N} (1 - \alpha_{d'}^{k})(-\alpha_{d'}^{k})}{\sum_{k=1}^{N} (1 - \alpha_{d'}^{k})^2}}. \quad (4.12)$$

Overall, Equation 4.11 and Equation 4.12 provide the expression of the X-coordinate of the vectors $\overrightarrow{g_{d'}}$ and $\overrightarrow{g'_{d'}}$ which minimize the individual fitting energy (Equation 4.7). The same reasoning (not detailed here) can be applied identically to retrieve the Y-coordinate of $\overrightarrow{g_{d'}}$ and $\overrightarrow{g'_{d'}}$.

Then, the entire vectors $\overrightarrow{\mathcal{G}_{d'}}$ and $\overrightarrow{\mathcal{G}'_{d'}}$ (defining $\overrightarrow{\mathcal{A}_{d'}}(\widehat{\mathcal{B}}_{d'}(f_j))$) which minimize Equation 5.13 under the current assignments can be updated similarly, by iterating the above computation for all the branches b^* of $\mathcal{B}^*$.

This overall *Assignment/Update* sequence is then iterated (Algorithm 1, *while* loop, line 4). Each iteration decreases $E_{W_2^{\mathcal{T}}}$ constructively: while the *Update* phase *(ii)* minimizes it under the current assignment, the next *Assignment* phase *(i)* further improves (by construction) the assignments, hence decreasing $E_{W_2^{\mathcal{T}}}$ overall. In our implementation, the algorithm stops when the fitting energy has decreased by less than 1% between two consecutive iterations.

4.3.3 Constraints

The previous section described an algorithm for optimizing the geodesic axis $\overleftrightarrow{\mathcal{A}_{d'}}$, in order to minimize the fitting energy $E_{W_2^{\mathcal{T}}}$ (Equation 4.5). However, this algorithm did not consider yet key constraints present in the definition of MT-PGA (Section 4.2.2), such as axis orthogonality. We describe now our strategy for extending the above algorithm with a succession of constraint enforcements.

Geodesic Enforcement

After the axis $\overleftrightarrow{\mathcal{A}_{d'}}$ has been optimized (Section 4.3.2) to minimize the fitting energy $E_{W_2^{\mathcal{T}}}$ (Equation 4.5), its associated vectors $\overrightarrow{\mathcal{G}_{d'}} \in \mathbb{R}^{2\times|\mathcal{B}^*|}$ and $\overrightarrow{\mathcal{G}'_{d'}} \in \mathbb{R}^{2\times|\mathcal{B}^*|}$ may represent displacements in the 2D birth/death space which are no longer geodesics in $\mathbb{B}$ (as illustrated in Figure 4.6a with the dashed arrows). Concretely, this situation occurs if the assignment described by the (optimized) vector $\overrightarrow{\mathcal{G}_{d'}}$ between the barycenter $\mathcal{B}^*$ and the

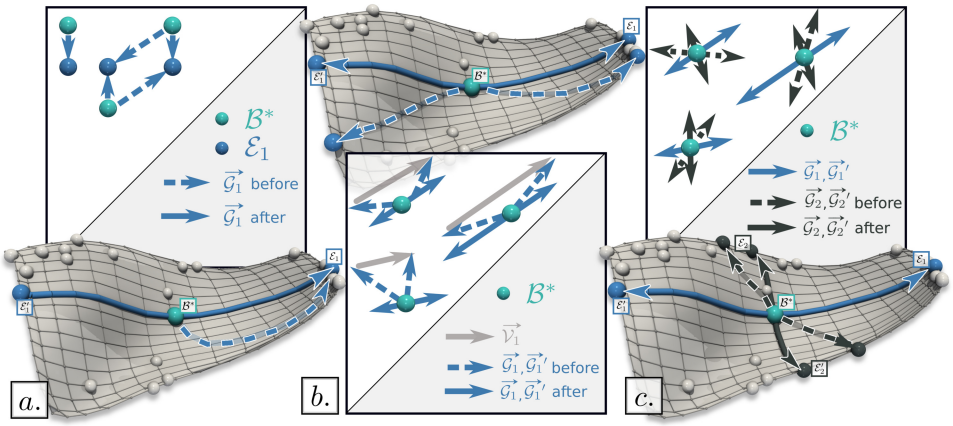


Figure 4.6 – *Constraint enforcements:* (a) *Geodesic enforcement guarantees that the vector* $\vec{\mathcal{G}_1}$ *describes an assignment which minimizes Equation 3.4;* (b) *Collinearity enforcement constructively aligns* $\vec{\mathcal{G}_1}$ *and* $\vec{\mathcal{G}_1'}$*;* (c) *Orthogonality enforcement updates* $\vec{\mathcal{G}_2}$ *and* $\vec{\mathcal{G}_2'}$ *via* Gram-Schmidt *orthogonalization.*

vector extremity $\mathcal{E}_{d'}$ is no longer optimal with regard to $W_2^{\mathcal{T}}$ (Equation 3.4), as $\vec{\mathcal{G}_{d'}}$ has been optimized freely when minimizing $E_{W_2^{\mathcal{T}}}$. This can be easily corrected by re-computing the optimal assignment between $\mathcal{B}^*$ and $\mathcal{E}_{d'}$, yielding an updated vector $\vec{\mathcal{G}_{d'}}$, now describing a valid geodesic between $\mathcal{B}^*$ and $\mathcal{E}_{d'}$. In Figure 4.6a, this is illustrated by the switch from the dashed arrows to the plain blue arrows. The same correction is applied to second vector of the axis, $\vec{\mathcal{G}_{d'}'}$.

Negative Collinearity Enforcement

Up to now, the two geodesics $\vec{\mathcal{G}_{d'}}$ and $\vec{\mathcal{G}_{d'}'}$ of the axis $\overleftrightarrow{\mathcal{A}_{d'}}$ have been optimized independently. Then, they may not be negatively collinear and thus, they may not describe a valid geodesic axis, as defined in Section 4.2.2. Let $\beta' = ||\vec{\mathcal{G}_{d'}}||/(||\vec{\mathcal{G}_{d'}}|| + ||\vec{\mathcal{G}_{d'}'}||)$ be the ratio describing the relative norm of $\vec{\mathcal{G}_{d'}}$ with regard to $\overleftrightarrow{\mathcal{A}_{d'}}$. Negative collinearity is now constructively enforced by updating $\vec{\mathcal{G}_{d'}}$ and $\vec{\mathcal{G}_{d'}'}$ (Figure 4.6b) such that $\vec{\mathcal{G}_{d'}} \leftarrow \beta' \times \vec{\mathcal{V}_{d'}}$ and $\vec{\mathcal{G}_{d'}'} \leftarrow -(1-\beta') \times \vec{\mathcal{V}_{d'}}$, where $\vec{\mathcal{V}_{d'}}$ is the direction vector of $\overleftrightarrow{\mathcal{A}_{d'}}$ (Section 4.2.2, $\vec{\mathcal{V}_{d'}} = \vec{\mathcal{G}_{d'}} - \vec{\mathcal{G}_{d'}'}$).

Orthogonality Enforcement

We now describe the enforcement of the orthogonality of $\overleftrightarrow{\mathcal{A}_{d'}}$ to all previous axes in the basis. In particular, given the direction vector $\vec{\mathcal{V}_{d'}} = \vec{\mathcal{G}_{d'}} - \vec{\mathcal{G}_{d'}'}$, we want to update $\vec{\mathcal{G}_{d'}}$ and $\vec{\mathcal{G}_{d'}'}$, such that $\vec{\mathcal{V}_{d'}} \cdot \vec{\mathcal{V}_{d''}} = 0$, for all $d'' \in \{1, 2, \ldots, d'-1\}$. Let $\mathcal{P}_{\vec{\mathcal{V}_{d''}}}(\vec{\mathcal{V}_{d'}}) = \left((\vec{\mathcal{V}_{d'}} \cdot \vec{\mathcal{V}_{d''}})/(\vec{\mathcal{V}_{d''}} \cdot \vec{\mathcal{V}_{d''}})\right) \times \vec{\mathcal{V}_{d''}}$ be the projection of $\vec{\mathcal{V}_{d'}}$ onto the direction spanned by $\vec{\mathcal{V}_{d''}}$. The orthogonality

of $\overrightarrow{\mathcal{V}_{d'}}$ to all vectors $\overrightarrow{\mathcal{V}_{d''}}$ is enforced (Figure 4.6c) by updating $\overrightarrow{\mathcal{V}_{d'}}$ as follows: $\overrightarrow{\mathcal{V}_{d'}} \leftarrow \overrightarrow{\mathcal{V}_{d'}} - \sum_{d''=1}^{d'-1} \mathcal{P}_{\overrightarrow{\mathcal{V}_{d''}}}(\overrightarrow{\mathcal{V}_{d'}})$.

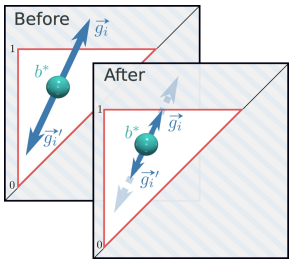
Note that the above constraints may go against each other: e.g. after orthogonality enforcement *(iii)*, $\overrightarrow{\mathcal{G}_{d'}}$ may no longer represent a valid geodesic *(i)*. Thus, we iterate the above succession of constraint enforcements, until $\overleftrightarrow{\mathcal{A}_{d'}}$ no longer evolves significantly (Algorithm 1, line 10). In practice, using a constant number of iterations (specifically 4) is sufficient to obtain satisfactory results (see Section 4.5.3 for further details regarding convergence).

4.3.4 From BDTs to MTs

Given an MT-PGA basis, for an arbitrary coordinate vector $\alpha \in [0,1]^{d_{max}}$, the resulting BDT $\widehat{\mathcal{B}}$ can be reconstructed (Section 4.2.2) by considering a BDT isomorphic to $\mathcal{B}^*$ such that:

$$\widehat{\mathcal{B}} = \widehat{\mathcal{B}}_{d_{max}} = \mathcal{B}^* + \sum_{i=1}^{d_{max}} \alpha_i \times \vec{\mathcal{G}}_i + (1-\alpha_i) \times \vec{\mathcal{G}}'_i. \tag{4.13}$$

Applying the above computation embeds each branch b of $\widehat{\mathcal{B}}$ as a point (x_b, y_b) in the 2D birth/death space. Since we consider that the input BDTs have been pre-normalized (Section 3.4.2), a valid MT $\widehat{\mathcal{T}}$ can in principle be reverted from $\widehat{\mathcal{B}}$, by recursively reverting Equation 3.23, assuming that $[x_b, y_b] \subseteq [0,1]$ (to enforce the nesting property of BDTs, Section 2.5). However, the latter assumption is not explicitly enforced in our constrained optimization (Section 4.3.2 and Section 4.3.3). Thus, in our entire framework (optimization included), when re-constructing a BDT (Equation 4.13), for each branch $b^* \in \mathcal{B}^*$, the corresponding direction vector $\vec{v}_i = \vec{g}_i - \vec{g}'_i$ (with $\vec{g}_i$ and $\vec{g}'_i$ being the entries of $\vec{\mathcal{G}}_i$ and $\vec{\mathcal{G}}'_i$ corresponding to the branch b^*) is temporarily scaled down if needed (inset, right) by locally renormalizing α_i, to guarantee that the extremities e_i ($\alpha_i = 1$) and e'_i ($\alpha_i = 0$) describe valid normalized birth/death locations (i.e. $x_{e_i} < y_{e_i}$ and $[x_{e_i}, y_{e_i}] \subseteq [0,1]$).



4.3.5 Computational Parameters

As described in Chapter 3, the Wasserstein metric $W_2^{\mathcal{T}}$ is subject to three parameters (ϵ_1, ϵ_2 and ϵ_3, described in detail in Section B.1), for which

we use the recommended default values (Section B.1). Otherwise, when switching ϵ_1 to 1, $W_2^{\mathcal{T}}$ becomes equivalent to $W_2^{\mathcal{D}}$ (see Chapter 3 or Section B.1) and our framework computes then a PGA basis of extremum persistence diagrams (PD-PGA for short).

Our algorithm itself (Algorithm 1) is subject to two parameters. *(i)* N_1 controls the size of $\mathcal{B}^*$, and hence the memory footprint of the MT-PGA basis. In practice, we set N_1 to a conservative value: 20% of the total number of branches in the ensemble ($\sum_{j=1}^{N} |\mathcal{B}(f_j)|$). *(ii)* N_2 controls the number of samples on each geodesic axis, to balance accuracy and speed. In practice, we set N_2 to 16.

4.4 Applications

This section illustrates the utility of our framework in concrete visualization tasks: data reduction and dimensionality reduction.

4.4.1 Data Reduction

Like any other data representation, merge trees can benefit from lossy compression, to facilitate their storage or transfer. This can be particularly useful in scenarios where the scalar fields of the ensemble cannot all be stored permanently (because of their size or the induced IO bottleneck) but are represented instead individually by a topological signature (e.g. a persistence diagram or a merge tree), yielding an ensemble of signatures. This can be the case for instance during large data acquisition campaigns,

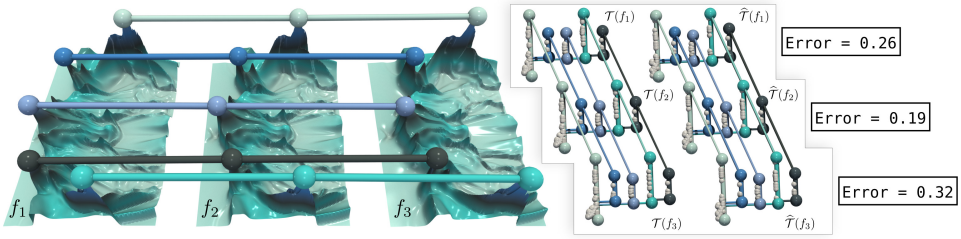


Figure 4.7 – *Tracking features (the five most persistent maxima, spheres) in time-varying (from left to right) 2D data (ion density during universe formation [Org04]). The computed tracking from Chapter 3 is identical, when using the input original merge trees (inset, left) or their versions compressed by MT-PGA (inset, right), which are highly similar visually. Here, since the input ensemble has few, small BDTs, the default reduction parameters ($d_{max} = 3$ and $N_1 = 20\%$) resulted in a modest compression factor (5.12). The reported relative reconstruction error (right) is given by the distance $W_2^{\mathcal{T}}$ between a tree and its reconstruction, divided by the maximum pairwise distance $W_2^{\mathcal{T}}$ observed in the input ensemble.*

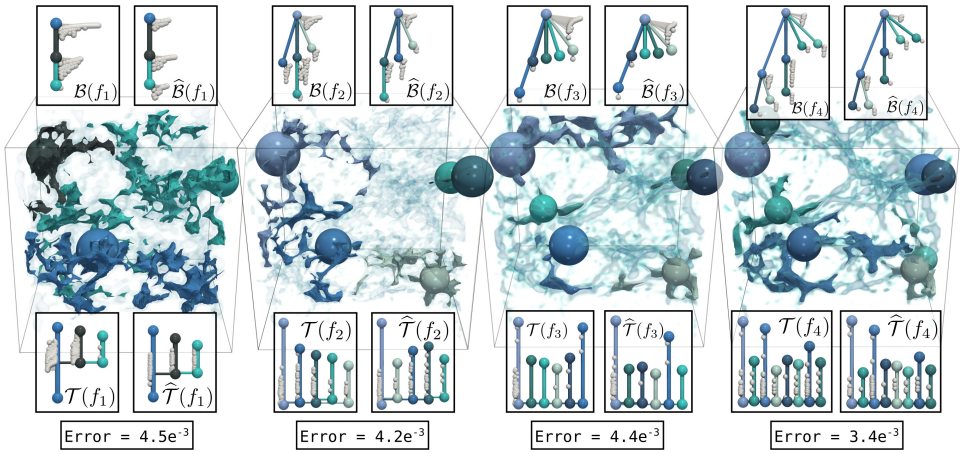


Figure 4.8 – *Topological clustering of the* 40 *members of a cosmology ensemble [Org04] (one member is represented for each cluster). The computed clustering (Chapter 3) is identical, when using the input original merge trees (bottom left inset) or their versions compressed by MT-PGA (bottom right inset). The optimal assignment between the left and right trees is visualized by the matching colors and the corresponding BDTs are reported at the top. Since this ensemble has many members, the default reduction parameters ($d_{max} = 3$ and $N_1 = 20\%$) resulted in an important compression factor (*19.27*). This compression still provided compressed trees (bottom right inset) that are visually similar to the input trees (bottom left inset): the number of prominent features (thick cylinders) and their persistence (heights of the cylinders of matching color) are well preserved. This is confirmed with the reconstructed BDTs (top right inset) which are mostly isomorphic to the input BDTs (thick cylinders of matching color). The reported relative reconstruction error (bottom) is given by the distance $W_2^{\mathcal{T}}$ between a tree and its reconstruction, divided by the maximum pairwise distance $W_2^{\mathcal{T}}$ observed in the input ensemble.*

or large-scale simulations, where a topological signature can be typically computed in-situ [ABG+15] to represent each time step [BNP+21]. In this scenario, lossy compression is useful to facilitate the manipulation (i.e. storage and transfer) of the resulting ensemble of merge trees. In Chapter 3 we have investigated the lossy compression of a temporal sequence of merge trees. In this section, we extend this work to arbitrary ensembles of merge trees thanks to MT-PGA.

Given its coordinates $\alpha^j \in [0,1]^{d_{max}}$, each BDT $\mathcal{B}(f_j)$ of $\mathcal{S}_{\mathcal{B}}$ can be reliably estimated with Equation 4.13. Thus, we present now an application to data reduction where the input ensemble of BDTs is compressed, by only storing to disk: *(i)* the origin $\mathcal{B}^*$ of $B_{\mathbb{B}}$, *(ii)* its axes $\{\overleftrightarrow{\mathcal{A}_1}, \overleftrightarrow{\mathcal{A}_2}, \ldots, \overleftrightarrow{\mathcal{A}_{d_{max}}}\}$ and *(iii)* the N BDT coordinates $\alpha^j \in [0,1]^{d_{max}}$. The compression quality can be controlled with two input parameters (Section 4.3.1). *(i)* d_{max} controls the number of axes (and thus the ability of the basis to capture mild variabilities). *(ii)* N_1 controls the size of $\mathcal{B}^*$ (and thus the ability of the basis to capture small features). The reconstruction error (Equation 4.5) will

be minimized for large values of both parameters, while the compression factor will be maximized for low values. In our reduction experiments, we set d_{max} to 3 and N_1 to its default value (Section 5.3.7). Figure 4.7 and Figure 4.8 show two examples of visualization applications (feature tracking and ensemble clustering, replicated from Chapter 3) where the BDTs compressed with the above strategy ($\widehat{\mathcal{B}}(f_j)$) have been used as an input. In both cases, the output is identical to the outcome obtained with the *original* BDTs. This shows the viability of our reconstructed BDTs (and MTs) and demonstrates the utility of our reduction scheme.

Table 4.1 reports the compression factors and average relative reconstruction error for all our test ensembles. In particular, for each input BDT $\mathcal{B}(f_i)$, we compute its reconstruction error via the $W_2^{\mathcal{T}}$ distance to its reconstruction $\widehat{\mathcal{B}}(f_i)$. In order to become comparable across ensembles, this distance is then divided by the maximum $W_2^{\mathcal{T}}$ distance observed among two input BDTs in the ensemble. Finally, this relative reconstruction error is averaged over all the BDTs of the input ensemble.

This table shows that our framework results in significant compression factors (9 or above) for the largest ensembles, i.e. the ensembles counting the most members and for which the BDTs are the largest (i.e. Dark matter, Sea Surface Height). On the contrary, modest compression factors tend to be obtained for the smallest ensembles, counting few members and few features. The average relative reconstruction error seems acceptable overall: 0.13 and 0.17 on average for PD-PGA and MT-PGA respectively.

Table 4.1 – *Compression factor and average relative reconstruction error of our algorithm for PD-PGA and MT-PGA (for d_{max} = 3 and N_1 = 20).*

Dataset	N	$\lvert\mathcal{B}\rvert$	PD-PGA		MT-PGA	
			Factor	Error	Factor	Error
Asteroid Impact (3D)	7	1,295	2.97	0.07	4.84	0.22
Cloud processes (2D)	12	1,209	5.94	0.19	7.39	0.01
Viscous fingering (3D)	15	118	2.23	0.13	4.71	0.02
Dark matter (3D)	40	2,592	10.00	0.04	19.27	0.04
Volcanic eruptions (2D)	12	811	9.99	0.12	4.83	0.04
Ionization front (2D)	16	135	2.56	0.14	5.12	0.40
Ionization front (3D)	16	763	3.27	0.17	4.85	0.46
Earthquake (3D)	12	1,203	1.42	0.18	2.19	0.33
Isabel (3D)	12	1,338	5.49	0.27	9.25	0.05
Starting Vortex (2D)	12	124	1.76	0.07	4.42	0.01
Sea Surface Height (2D)	48	1,787	19.59	0.18	9.48	0.48
Vortex Street (2D)	45	23	1.86	0.04	11.84	0.02

Finally, note that for each ensemble, the merge tree based clustering (Chapter 3) computed from the input BDTs is strictly identical to the clustering computed from the reconstructed BDTs. This confirms the viability of our reconstructed BDTs, and their usability for typical visualization and analysis tasks.

4.4.2 Dimensionality Reduction

The MT-PGA basis $B_{\mathbb{B}}$ can also be used to generate 2D layouts of the ensemble, for its global visual inspection. This is achieved by embedding each input BDT $\mathcal{B}(f_j)$ as a point in the 2D plane, given its first two coordinates in $B_{\mathbb{B}}$. To prevent an artificial anisotropic distortion ($\alpha^j \in [0,1]^2$), we scale the coordinates of each BDT, to account for the variation in length of the axes, and we embed each BDT at coordinates $\left(\alpha_1^j \times W_2^{\mathcal{T}}(\mathcal{E}_1, \mathcal{E}_1'), \alpha_2^j \times W_2^{\mathcal{T}}(\mathcal{E}_2, \mathcal{E}_2')\right)$. This results in a summarization view of the overall ensemble, which groups together BDTs which are close (given the metric $W_2^{\mathcal{T}}$) and which exhibit a similar variability with regard to $\mathcal{B}^*$. Note that, given an arbitrary point of the 2D layout, its BDT (and MT) can be efficiently reconstructed with Equation 4.13. This enables interactive navigations within the ensemble (Figure 4.1f). We augment our planar layouts with the following two improvements, to further characterize the global and local variability in the ensemble.

***(i)* Principal geodesic surface.** To visually convey the curved (i.e. non-Euclidean) nature of the Wasserstein metric space $\mathbb{B}$, we introduce a 3D embedding of our planar layouts, which we call Principal Geodesic Surface (PGS). First, our 2D layout is sampled along a $N_x \times N_y$ regular grid $G_{\mathbb{B}}$ (in practice, $N_x = N_y = N_2$). For each vertex of $G_{\mathbb{B}}$, a BDT can be reconstructed (Equation 4.13), enabling the computation of a distance matrix $\mathbb{D}_{G_{\mathbb{B}}}$, where $\mathbb{D}_{G_{\mathbb{B}}}(i,j)$ is the $W_2^{\mathcal{T}}$ distance between the vertices i and j of $G_{\mathbb{B}}$. Then, $G_{\mathbb{B}}$ is embedded in 3D by multidimensional scaling [KW78] of $\mathbb{D}_{G_{\mathbb{B}}}$. The resulting PGS provides the same visual information as the planar layout, but in the form of a surface in 3D, parameterized by α_1 and α_2 (Figure 4.3), which conveys visually the curved nature of $\mathbb{B}$.

***(ii)* Persistence correlation view.** As detailed after, we compute the correlation $\rho(p_i, \alpha_k)$ between the coordinate α_k and the persistence of the i^{th} branch (i.e. the branch in the input BDTs mapped to the i^{th} branch of $\mathcal{B}^*$ given the optimal assignment induced by $W_2^{\mathcal{T}}$, Equation 3.4). Then, the i^{th}

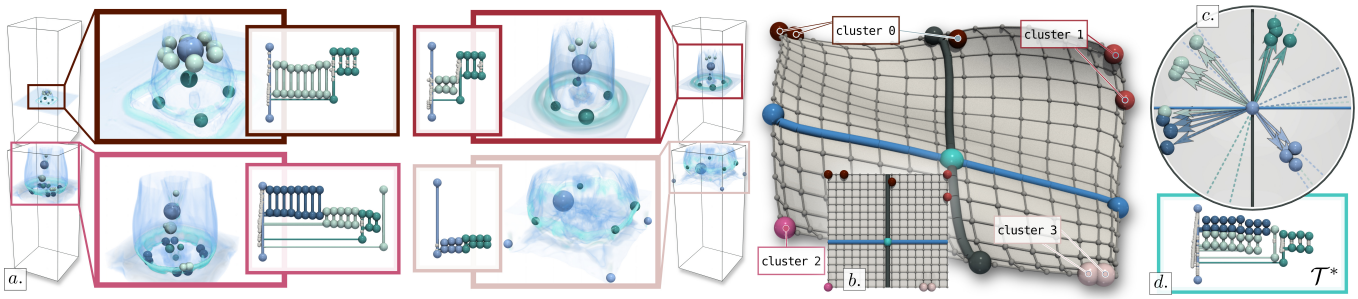


Figure 4.9 – *Feature variability interpretation with* Persistence Correlation Views *(PCV) on the* Ionization (3D) *ensemble.* (a)*: Four members of the ensemble (one per ground-truth class).* (b)*:* Principal Geodesic Surface *(PGS) and its planar layout, computed by MT-PGA (sphere color: ground-truth classes).* (c)*: In the correlation view, several features (i.e. branches) are located near the disk boundary, indicating high correlations. The dark green features are highly correlated with the direction* $(0, \alpha_2)$*, given their location (near the black axis, upper half). Back in the data, this indicates that the corresponding features (basis of the ionization front, green spheres in the data and merge tree insets) will be the most persistent for the ensemble members with high* α_2 *values (clusters 0 and 1,* (b)*). Light green features in the PCV (upper left quadrant ,* (c)*) are highly correlated with the direction* $(-\alpha_1, \alpha_2)$*: these features (front extremities, insets) will be the most persistent in the top left corner of the PGS (*(b)*, cluster 0). This is confirmed visually in the data (zoom insets), where the maxima are represented by spheres of matching color (the radius denotes persistence).*

branch of the barycenter BDT $\mathcal{B}^*$ can be embedded in a Persistence Correlation View (PCV), by placing an arrow between the origin and the point $(\rho_{p_i,\alpha_1}, \rho_{p_i,\alpha_2})$. This enables a *local* interpretation of the feature variability within the ensemble. Specifically, it enables the visual identification of the features whose persistence is strongly correlated with a given direction in the MT-PGA basis. Such features are located on the disk boundary of the correlation view (largest arrows in Figure 4.1, Figure 4.9, Figure 4.10), and they are the most responsible for the variability in the ensemble. For each of these features, their matching to the origin $\mathcal{B}^*$ of the MT-PGA basis is encoded with the color map, which enables their direct inspection in the data. We now describe the computation of the correlation between the persistence of the i^{th} feature of an input BDT $\mathcal{B}(f_j)$ and its coordinate α_k^j along the geodesic axis $\overleftrightarrow{\mathcal{A}_k}$. It is derived from the seminal Pearson's correlation [Pea95], applied on the above terms (made available by our framework). Let $\mathbb{P}$ be an $(N_1 \times N)$-matrix, such that the entry $\mathbb{P}(i,j)$ denotes the topological persistence, in the j^{th} input BDT, of the branch $b_j \in \mathcal{B}(f_j)$ mapped to the i^{th} most persistent branch of $\mathcal{B}^*$ given the optimal assignment induced by $W_2^{\mathcal{T}}$ (Equation 3.4). Next, let $\mathbb{A}$ be a $(d_{max} \times N)$-matrix, such that then entry $\mathbb{A}(k,j)$ denotes the coordinate α_k^j of the BDT $\mathcal{B}(f_j)$ along the axis $\overleftrightarrow{\mathcal{A}_k}$.

In the following, we aim at assessing how much the persistence of the branches in $\mathcal{B}(f_j)$ is correlated with the coordinate α_k^j. Let ρ_{p_i,α_k} be the

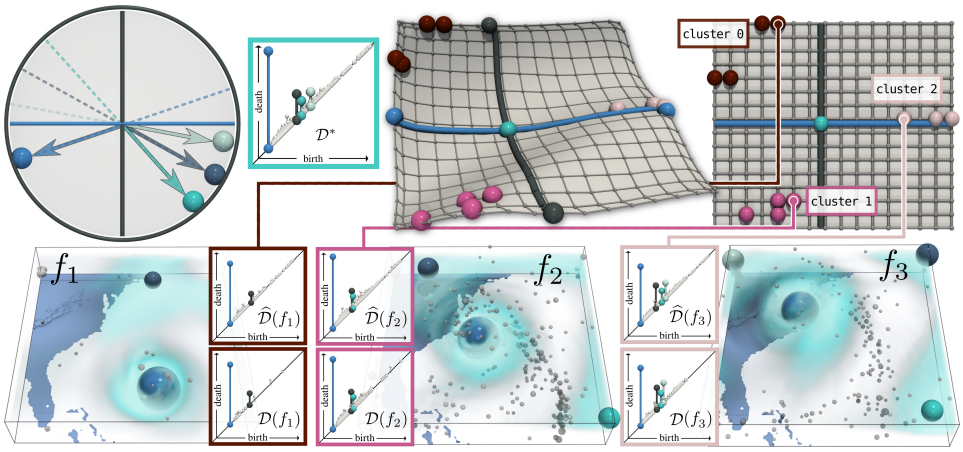


Figure 4.10 – *Feature variability interpretation for the* Isabel *ensemble, with PD-PGA. In the PCV (top left), the blue feature (eye of the hurricane) is highly correlated with the direction* $(-\alpha_1, 0)$*, which indicates that its persistence will be stronger for the leftmost points in the PGS (clusters 0 and 1) and weaker for the rightmost points (cluster 2). In other words, the winds in the eye of the hurricane are significantly weaker in the cluster 2 (landfall phase of the hurricane). On the contrary, the other highly correlated features (right half of the PCV) are correlated with the direction* $(\alpha_1, 0)$*: their persistence will be stronger for the rightmost points (cluster 2). This is confirmed visually when inspecting the corresponding features in the data and the input diagrams (bottom). Similarly to MT-PGA, PD-PGA enables a data reduction (compression factor:* 5.49*), while guaranteeing visually similar reconstructed diagrams (*$\widehat{\mathcal{D}}(f_j)$*).*

correlation between the persistence p_i (i^{th} line of $\mathbb{P}$) and the coordinate α_k (k^{th} line of $\mathbb{A}$). It is given by the following expression, where $\overline{p_i}$ and $\overline{\alpha_k}$ are the average values for the the i^{th} line of $\mathbb{P}$ and the k^{th} line of $\mathbb{A}$, and where σ_{p_i} and σ_{α_k} stand for their standard deviation:

$$\rho_{p_i,\alpha_k} = \frac{\sum_{j=1}^{N} \left(\mathbb{P}(i,j) - \overline{p_i}\right) \times \left(\mathbb{A}(k,j) - \overline{\alpha_k}\right)}{N \times \sigma_{p_i} \times \sigma_{\alpha_k}}.$$

Together, our Principal Geodesic Surface (PGS) coupled with our Persistence Correlation View (PCV) enable both a global and local inspection of the feature variability in the ensemble. In particular, in the example of Figure 4.1, our visualization indicates overall that the global maximum of the seismic wave (largest branch in the merge trees, purple, Figure 4.1c) is mostly correlated with the direction $(-\alpha_1, 0)$ in the PCV (Figure 4.1d), and therefore, mostly prominent in the cluster 1 (located on the left of the PGS, Figure 4.1b, for the lowest α_1 values). In contrast, the other features (blue, cyan and dark blue branches) are less correlated with this direction (shorter arrows in the PCV Figure 4.1d), which indicates that they are slightly less variable through this direction: their persistence de-

creases less quickly than the global maximum when transitioning from cluster 1 to 2 and 3 (dark red, orange and pink spheres respectively in the PGS Figure 4.1g). This indicates that the initial energy of the seismic wave (represented by the global maximum) quickly spreads into multiple wavefronts (spheres of matching colors in the data, Figure 4.1a), whose individual energy decreases through time more slowly than the global maximum (each cluster represents a different temporal phase in the simulation, from top to bottom in Figure 4.1a). Figure 4.9 and Figure 4.10 present a similar analysis and we refer the reader to the detailed captions for specific interpretations. Note that for Figure 4.10, the features do not exhibit a clear global structure and the persistence diagram can be used instead of the merge tree.

4.5 Results

This section presents experimental results obtained on a computer with two Xeon CPUs (3.2 GHz, 2x10 cores, 96GB of RAM). The input merge trees were computed with FTM [GFJT19a] and pre-processed to discard noisy features (persistence simplification threshold: 0.25% of the data range). We implemented our approach in C++ (with the OpenMP task runtime), as modules for TTK [TFL+17, BMBF+19]. Experiments were performed on the benchmark of public ensembles described in Appendix A, which includes a variety of simulated and acquired 2D and 3D ensembles extracted from previous work and past SciVis contests [Org04].

4.5.1 Time Performance

Barycenters and geodesics are computed with the approach of Chapter 3, which implements fine-grain task-based shared-memory parallelism. Specifically, during axis projection, $N \times N_2$ geodesics need to be computed (Section 4.3.2), each geodesic requiring typically $\mathcal{O}(|\mathcal{B}|^2)$ steps in practice, where $|\mathcal{B}|$ is the size of the input BDTs. In practice, this is the most expensive part of our algorithm. These geodesics are computed concurrently (by submitting each geodesic to the task pool). In comparison, the evaluations of the numerical expressions (Algorithm 1, lines 7, 8, 12 and 13) have a nearly negligible cost. Table 4.2 evaluates the time performance of our framework for persistence diagrams (PD-PGA) and merge trees (MT-PGA), for $d_{max} = 2$. In sequential mode, the running time is indeed a function of the size of the ensemble (N) and the size of trees ($|\mathcal{B}|$). It is

Table 4.2 – *Running times (in seconds) of our algorithm for PD-PGA and MT-PGA computation (for $d_{max} = 2$, first sequential, then with 20 cores).*

Dataset	N	$\lvert\mathcal{B}\rvert$	PD-PGA			MT-PGA		
			1 c.	20 c.	Speedup	1 c.	20 c.	Speedup
Asteroid Impact (3D)	7	1,295	1,392.17	147.40	9.44	1,180.72	117.97	10.01
Cloud processes (2D)	12	1,209	817.64	61.88	13.21	517.94	38.49	13.46
Viscous fingering (3D)	15	118	86.69	9.17	9.45	42.89	4.71	9.11
Dark matter (3D)	40	2,592	18,388.86	1,366.45	13.46	24,480.42	1,758.04	13.92
Volcanic eruptions (2D)	12	811	460.17	37.99	12.11	1,004.37	81.75	12.29
Ionization front (2D)	16	135	104.74	12.00	8.73	55.73	6.26	8.90
Ionization front (3D)	16	763	3,750.00	300.96	12.46	4,029.71	294.29	13.69
Earthquake (3D)	12	1,203	3,896.52	338.64	11.51	1,973.49	158.12	12.48
Isabel (3D)	12	1,338	1,969.79	164.49	11.98	1,472.54	115.66	12.73
Starting Vortex (2D)	12	124	17.71	2.72	6.51	11.51	1.65	6.98
Sea Surface Height (2D)	48	1,787	12,420.98	670.00	18.54	27,791.00	1,669.52	16.65
Vortex Street (2D)	45	23	18.75	2.69	6.97	35.79	3.93	9.11

slightly slower for MT-PGA than for PD-PGA, but timings remain comparable overall. In parallel, speedups are the most important for the largest ensembles. However, the iterative nature of our algorithm has an impact on parallel efficiency (the end of each loop implies a synchronization). Still, our parallelization significantly reduces computation times, with less than 6 minutes on average and at most 30 minutes for the largest ensembles, which we believe is an acceptable pre-processing time, prior to interactive exploration.

4.5.2 Framework Quality

Figure 4.7 and Figure 4.8 report compression factors for our application to data reduction (Section 4.4.1). These are ratios between the storage size of the input BDTs and that of the MT-PGA basis (barycenter, axes and coordinates). This factor is modest for a small example (Figure 4.7), with few branches (135) and few members (16). Then, the overhead of the MT-PGA basis is non-negligible. In contrast, for a larger ensemble (Figure 4.8), this overhead is negligible and high compression factors (30) can be achieved, while providing reconstructed merge trees which are highly similar visually and which are still viable for the applications.

Figure 4.11 provides a visual comparison of the planar layouts generated by a selection of typical dimensionality reduction techniques. This includes the classical Euclidean PCA (in $\mathbb{R}^{N_v}$, where N_v is the number of vertices in $\mathcal{M}$), MDS [KW78] and t-SNE [vdMH08] (both with $W_2^{\mathcal{T}}$). Certain approaches [RT16, AVRT16, LPW21] applied PCA on top of *vectorizations* of topological descriptors (Section 4.1.1). A similar strategy can

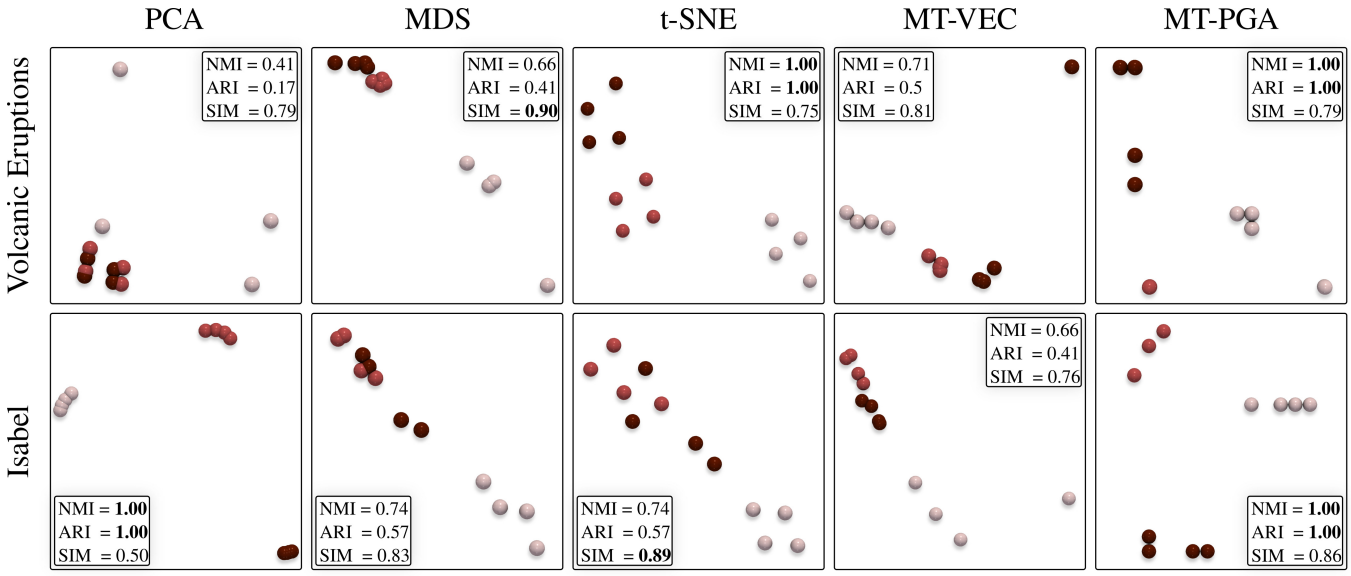


Figure 4.11 – *Comparison of planar layouts for typical dimensionality reduction techniques, on two input ensembles (bold: best value for each quality score). The color encodes the classification ground-truth (Appendix A).*

be considered in our case, by embedding each input BDT $\mathcal{B}(f_j)$ in $\mathbb{R}^{2\times|\mathcal{B}^*|}$, such that the i^{th} entry of this vector corresponds to the birth/death location of the i^{th} branch of $\mathcal{B}(f_j)$ (i.e. the branch of $\mathcal{B}(f_j)$ mapping to the i^{th} branch of $\mathcal{B}^*$ by the optimal assignment of $W_2^{\mathcal{T}}$, Equation 3.4). PCA is then computed for this vectorization (column *MT-VEC*). As shown in Figure 4.11, the layouts generated with MT-PGA nicely separates the ground-truth classes, while other techniques tend to artificially group them or even merge them. To further quantify this structure preservation, we run k-means in the 2D layouts and evaluate the quality of the resulting clustering (given the ground-truth Appendix A) with the normalized mutual information (NMI) and adjusted rand index (ARI). The MT-PGA layout is the only one in Figure 4.11 which generates an exact clustering in both cases ($NMI = ARI = 1$).

Table 4.3 also extends this quantitative comparison to all our input ensembles and confirms the superiority, on average, of MT-PGA for the preservation of the clusters. Table 4.3 also includes a metric similarity indicator, SIM, which evaluates the preservation by a layout of the Wasserstein metric $W_2^{\mathcal{T}}$. Specifically, given two points x and y in a planar layout (with BDTs $\mathcal{B}(f_x)$ and $\mathcal{B}(f_y)$), we first measure their pairwise distortion:

$$\delta(x,y) = \left(||x-y||_2 - W_2^{\mathcal{T}}\left(\mathcal{B}(f_x),\mathcal{B}(f_y)\right)\right)^2.$$

This measure is then normalized into:

$$\delta'(x,y) = \frac{\delta(x,y)}{\max_{\forall x\neq y}\left(\delta(x,y)\right)}.$$

Table 4.3 – *Comparison of layout quality scores, averaged over all ensembles (bold: best values). The layouts induced by MT-PGA better preserve the global structure of the ensemble and still preserve well $W_2^{\mathcal{T}}$.*

Indicator	PCA	MDS ($W_2^{\mathcal{T}}$) [KW78]	t-SNE ($W_2^{\mathcal{T}}$) [vdMH08]	MT-VEC	**MT-PGA**
NMI	0.79	0.82	0.88	0.78	**0.94**
ARI	0.71	0.73	0.84	0.64	**0.90**
SIM	0.60	**0.85**	0.75	0.76	0.80

Finally, we evaluate the global indicator $SIM := 1 - \overline{\delta'}$, where $\overline{\delta'}$ stands for the average of $\delta'(x, y)$ for all pairs (x, y) in the ensemble. SIM values lie within the interval $[0, 1]$ and are optimal near 1.

As expected, MDS maximizes this score by design, while MT-PGA produces the second best score. Overall, MT-PGA preserves well $W_2^{\mathcal{T}}$ as well as the global ensemble structure. In comparison to standard techniques (e.g. MDS or t-SNE), it also supports additional features, such as correlation views and interactive reconstructions (Figs. 4.1, 4.9 and 4.10).

Figure 5.13 and Figure 4.13 extends Figure 4.11 to all our test ensembles and it confirms visually the conclusions of the table of aggregated scores (Table 4.3). In particular, it confirms that MT-PGA provides a *trade-off* between the respective advantages of standard techniques such as MDS [KW78] and t-SNE [vdMH08]. Specifically, MDS is known to preserve the input metric well, while t-SNE tends to better preserve the global structure of the data (i.e. the ground-truth classification), at the expense of metric violation. MT-PGA provides a *balance* between these two behaviors: *(i)* it improves structure preservation over MDS (it provides equivalent or better NMI/ARI scores for 11 out of 12 ensembles) and *(ii)* it improves metric preservation over t-SNE (it provides an equivalent or better SIM score for 9 out of 12 ensembles). Visually, this means that MT-PGA groups together the members belonging to the same ground-truth class, while providing a layout which is more faithful than t-SNE's regarding the distances between the corresponding merge trees.

As can be expected, the straightforward PCA preserves the $W_2^{\mathcal{T}}$ metric poorly (as it is based on the L_2 norm). Since it relies on a rough approximation of $\mathbb{B}$, a PCA derived from a vectorization of the BDTs (MT-VEC) preserves poorly the global structure of the ensemble (MT-PGA provides equivalent or better NMI/ARI scores for 11 out of 12 ensembles).

Figure 4.14 reports the evolution of the normalized fitting energy for PD and MT-PGA computations ($d_{max} = 2$). Sudden energy drops can be observed with clear kinks in the curves, indicating the finalization of the

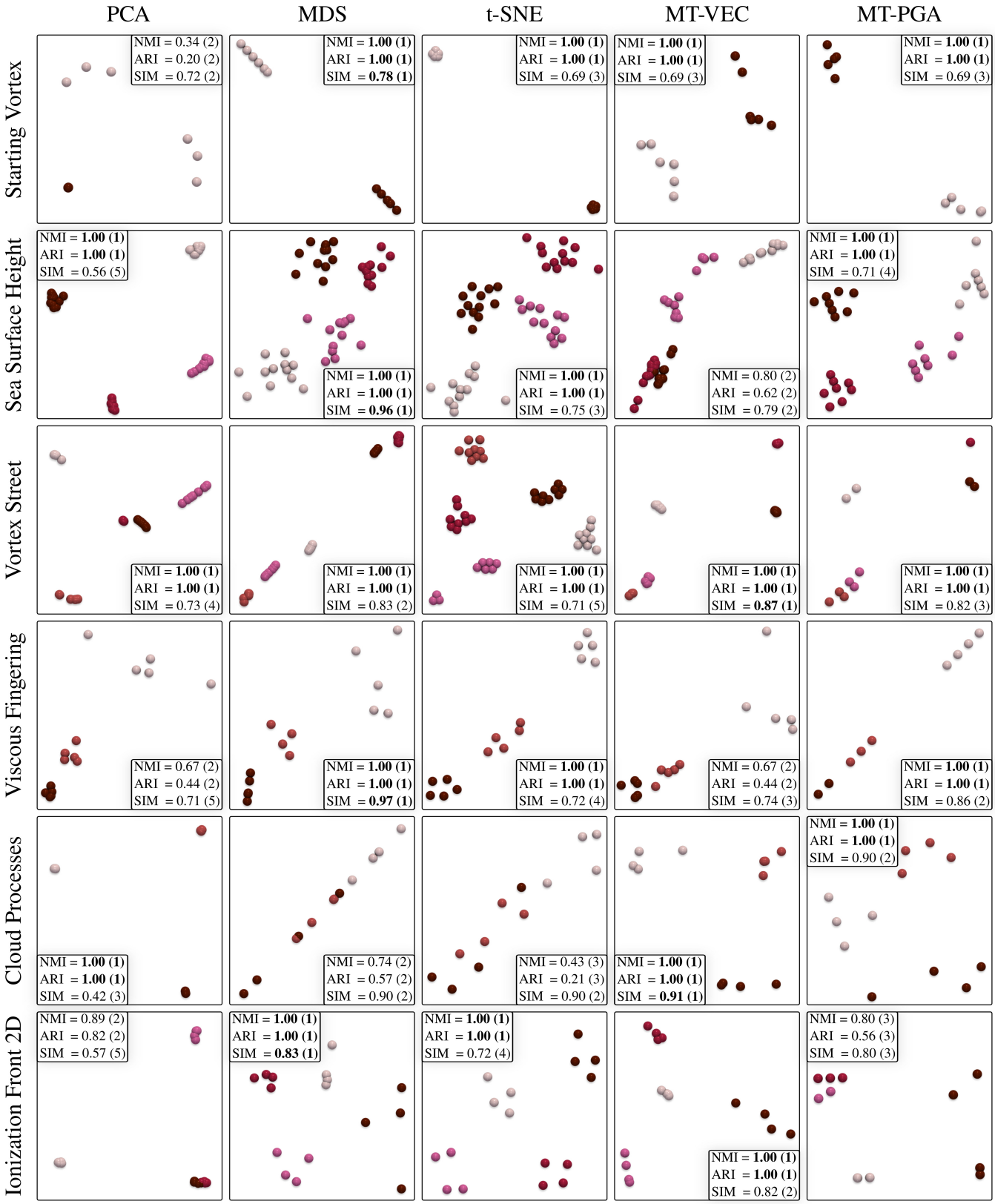


Figure 4.12 – *Comparison of planar layouts for typical dimensionality reduction techniques on the first half of our test ensembles. The color encodes the classification ground-truth Appendix A. For each quality score, the best value appears bold and the rank of the score among all methods is in parenthesis.*

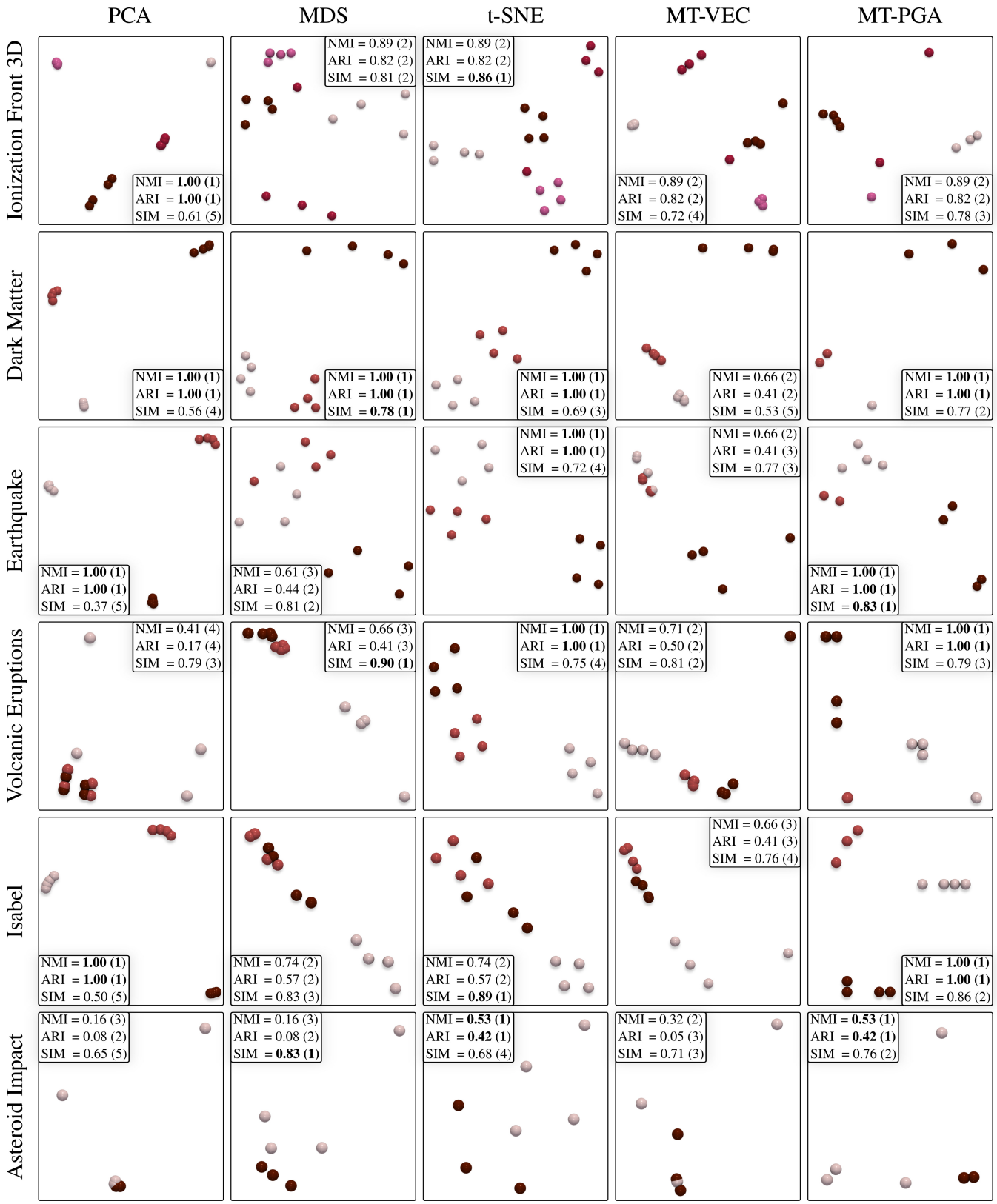


Figure 4.13 – *Comparison of planar layouts for typical dimensionality reduction techniques on on the second half of our test ensembles. The color encodes the classification ground-truth Appendix A. For each quality score, the best value appears bold and the rank of the score among all methods is in parenthesis.*

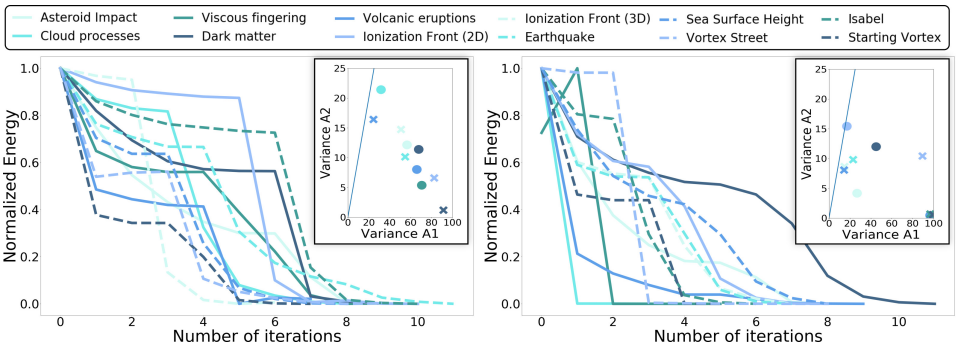


Figure 4.14 – *Evolution of the normalized fitting energy, for PD (left) and MT (right) PGA. The scatter plots show the projected variance ($\overleftrightarrow{\mathcal{A}_1}$ VS $\overleftrightarrow{\mathcal{A}_2}$).*

first dimension (Section 4.3.1), and the switch to the second, which immediately improves the overall fit. The algorithm stops when the energy decreases by less than 1%. The curve flat tails indicate that this criterion is reasonable. Figure 4.14 also reports scatter plots of the *projected variances*. For a given axis, it is the percentage of the variance of the projected trees (along the axis) over the global variance of the input BDTs (i.e. average of the squared $W_2^{\mathcal{T}}$ distances to the barycenter $\mathcal{B}^*$). For all examples, the corresponding point is located below the diagonal (blue line): the projected variance is indeed larger for the first axis than for the second. This confirms the ability of our algorithm, similarly to the classical PCA, to identify in practice the most informative directions first. This is confirmed visually in our 2D layouts, where the first axis (blue) is always longer than the second (black).

Figure 4.15 extends to 10 dimensions the scatter plots of projected variance reported in Figure 4.14, which were computed for only 2 dimensions. This figure confirms the conclusions of Figure 4.14. Except for a very mild oscillation for a specific dataset (*"Cloud processes"*, PD-PGA, with a local maximum of low amplitude at 4 dimensions), overall, the projected

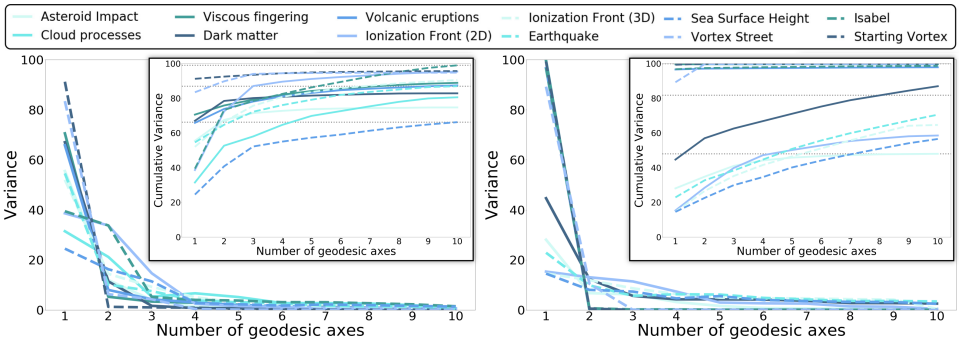


Figure 4.15 – *Evolution of the projected variance (and cumulative variance, inset) with the number of geodesic axes for PD-PGA (left) and MT-PGA (right).*

variance is indeed monotonically decreasing in practice for an increasing number of dimensions (i.e. the algorithm does tend to identify the most informative directions first).

4.5.3 Limitations

Our overall strategy (alternation of fitting and constraint enforcement) is similar at a high-level to previous work on the optimal transport of histograms [SC15]. Thus, it shares the same high-level limitations. The constraint enforcement (Section 4.3.3) induces, by construction, an energy increase. Then, it is possible that, at the next iteration, the fitting optimization (which is itself guaranteed to decrease the energy, Section 4.3.2) does not manage to compensate the above increase. This situation occurs during the first iteration for the *Viscous Fingering* ensemble (dark green line, Figure 4.14, right), where the energy increased between two iterations. This temporary energy increase is the only one we observed in all our experiments. Moreover, it did not impact the rest of the algorithm, as it was immediately compensated at the next iteration (Figure 4.14). Then, we believe that this theoretical limitation has a very limited impact in practice and that robust implementations can be obtained by stopping the algorithm in case of multiple, consecutive energy increases (which we have not observed). Another limitation involves the sampling of the geodesic axes (N_2, Section 4.3.2). When N_2 is too low, BDTs which are close in $\mathbb{B}$ may project to the same points on the geodesic axes, possibly resulting overall in collocated points in the MT-PGA basis. This can be easily resolved by increasing N_2, at the cost of increased computation times. Finally, similarly to barycenter optimization ([TMMH14, VBT20] or Chapter 3), the overall fitting energy (Equation 4.5) is non-convex and can in principle admit multiple local minimizers. However, our experiments indicate that the axes returned by our approach are relevant, as the projected variance does decrease for increasing dimensions (Section 4.5.2).

4.6 Summary

In this chapter, we presented a computational framework for the Principal Geodesic Analysis of merge trees (MT-PGA), with applications to data reduction and dimensionality reduction. In particular, the visualizations derived from our core contribution (Figure 4.1, Figure 4.9 and Figure 4.10) enable the interactive, visual inspection of the variability in the ensemble,

both at a global level (with our two-dimensional layouts) and at a feature level (with our persistence correlation views). Our framework trivially extends to extremum persistence diagrams and our algorithm enables in both cases PGA basis computations within minutes for real-life ensembles.

5 Wasserstein Auto-Encoders of Merge Trees and Persistence Diagrams

Contents

This chapter presents a computational framework for the Wasserstein auto-encoding of merge trees (MT-WAE), a novel extension of the classical auto-encoder neural network architecture to the Wasserstein metric space of merge trees (Chapter 3). In contrast to traditional auto-encoders which operate on vectorized data, our formulation explicitly manipulates merge trees on their associated metric space at each layer of the network, resulting in superior accuracy and interpretability. Our novel neural network approach can be interpreted as a non-linear generalization of previous linear attempts (Chapter 4) at merge tree encoding. It also trivially extends to persistence diagrams. Extensive experiments on public ensembles demonstrate the efficiency of our algorithms, with MT-WAE computations in the orders of minutes on average. We show the utility of our contributions in two applications adapted from previous work on merge tree encoding (Chapter 4). First, we apply MT-WAE to *data reduction* and reliably compress merge trees by concisely representing them with their coordinates in the final layer of our auto-encoder. Second, we document an application to *dimensionality reduction*, by exploiting the latent space of our auto-encoder, for the visual analysis of ensemble data. We illustrate the versatility of our framework by introducing two penalty terms, to help preserve in the latent space both the Wasserstein distances between merge trees, as well as their clusters. In both applications, quantitative experiments assess the relevance of our framework. Finally, we provide a C++ implementation that can be used for reproducibility.

The work presented in this chapter has been submitted to the journal IEEE Transactions on Visualization and Computer Graphics.

Our Contributions in one Image

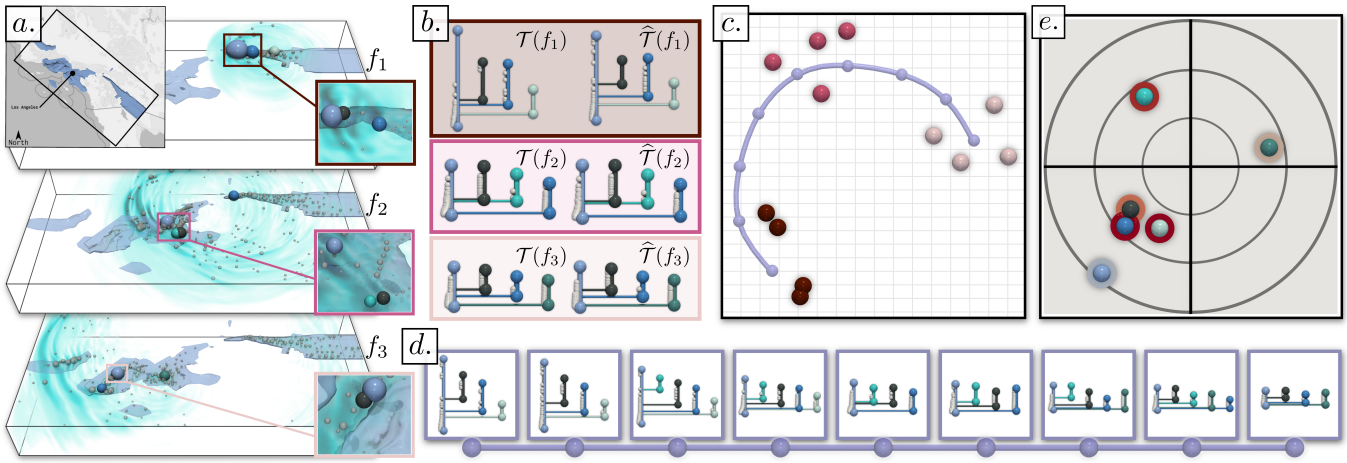


Figure 5.1 – *Visual analysis of the* Earthquake *ensemble* $\big($*(a) each ground-truth class is represented by one of its members*$\big)$*, with our Wasserstein Auto-Encoder of Merge Trees (MT-WAE). We apply our contributions to data reduction and compress the input trees* $\big($*(b), right*$\big)$ *by simply storing their coordinates in the last decoding layer of our network. We exploit the latent space of our network to generate 2D layouts of the ensemble* (c)*. In contrast to classical auto-encoders, MT-WAE maintains merge trees at each layer of the network, which results in improved accuracy and interpretability. Specifically, the reconstruction of user-defined locations* $\big($*(c), purple*$\big)$ *enables an interactive exploration of the latent space: the reconstructed curve* (d) *enables a continuous navigation between the clusters* $\big($*from dark red to pink and light pink,* $(c)$$\big)$*. MT-WAE also supports persistence correlation views* (e) *(adapted from Chapter 4), which reveal the barycenter's persistent features which exhibit the most variability in the ensemble (far from the center). Finally, by tracking the persistence evolution of individual features as they traverse the network down to its latent space, we introduce a* Feature Latent Importance *measure, which identifies the most* informative *features within the ensemble* $\big($*(e), red circles*$\big)$*.*

5.1 Context

Developing statistical analysis tools to support the interactive analysis and interpretation of ensemble data became an important challenge. Recently, several works explored this direction, in particular with the notion of *average topological representation* ([TMMH14, LCO18, VBT20, YWM+19b] and Chapter 3). These approaches can produce a topological representation which nicely summarizes the ensemble. Moreover, their application to clustering (Chapter 3) reveal its main trends. However, they do not provide any hints regarding the variability of the features in the ensemble. For this, in Chapter 4, we extended the notion of principal geodesic analysis to ensembles of merge trees. However, this approach implicitly assumes a linear relation between the merge trees of the ensemble. Specifically, it assumes that merge tree branches evolve linearly (in the birth/death space, Section 2.4) within the ensemble.

This chapter addresses this issue with a novel formulation based on neural networks and introduces the first framework for the non-linear encoding of merge trees, hence resulting in superior accuracy. Specifically, we formulate merge tree non-linear encoding as an auto-encoding problem (Section 5.2). We contribute a novel neural network called *Wasserstein Auto-Encoder of Merge Trees*. This network is based on a novel layer model, capable of processing merge trees natively, without pre-vectorization. We believe this contribution to be of independent interest, as it enables an accurate and interpretable processing of merge trees by neural networks (without restrictions to auto-encoders). We contribute an algorithm for the optimization of such a network (Section 5.3). We illustrate the relevance of our contributions for visual analysis with two applications, data reduction (Section 5.4.1) and dimensionality reduction (Section 5.4.2). Similarly to previous linear attempts (Chapter 4), since our approach is based on the Wasserstein distance between merge trees (Chapter 3), which generalizes the Wasserstein distance between persistence diagrams [EH09], our framework trivially extends to persistence diagrams by simply adjusting a single parameter.

5.1.1 Related Work

We refer to Section 3.1.1 for an overview of the literature regarding ensemble analysis and topological methods.

In Chapter 4, we extended the generic notion of principal geodesic analysis to the Wasserstein metric space of merge trees, resulting in im-

proved accuracy and interpretability with regard to the straightforward application of PCA on merge tree vectorizations. Similarly, Sisouk et al. [SDT23] introduced a simpler approach for the linear encoding of persistence diagrams, with a less constrained framework based on dictionaries. However, these approaches implicitly assume a linear relation between the topological descriptors of the ensemble. For instance, it assumes that a given feature (i.e. a given branch of the merge tree) evolves linearly in the birth/death space (Section 3.2) within the ensemble. However, this hypothesis is easily challenged in practice (Figure 5.3 and Figure 5.5), potentially leading to inaccuracies. Our work overcomes this limitation with a drastically different formulation (based on auto-encoding neural networks) and introduces the first framework for the non-linear encoding of merge trees, resulting in superior accuracy.

General purpose methods have been documented for non-linear encoding (e.g. *topological auto-encoders* [MHRB20], or *Wasserstein auto-encoders* [TBGS18]). Our work drastically differs from these methods, in terms of design and purpose. These methods [MHRB20, TBGS18, HMR21] employ a classical auto-encoder (Section 5.2.1) to which they add specialized penalty terms. Then, their input is restricted to point sets (or vectorized data). In contrast, our work focuses on sets of merge trees (or persistence diagrams). This different kind of input requires a novel neural network model, capable of processing these topological objects natively (Section 5.2).

5.1.2 Contributions

This chapter makes the following new contributions:

1. *An approach to Merge tree non-linear encoding:* We formulate the non-linear parametrization of the Wasserstein metric space of merge trees (and persistence diagrams) as an auto-encoding problem. Our formulation (Section 5.2) generalizes and improves previous linear attempts (Chapter 4).

2. *A vectorization-free neural network architecture for Merge Trees:* We contribute a novel neural network architecture called *Wasserstein Auto-Encoder of Merge Trees*, inspired by the classical auto-encoder, which can *natively* process merge trees (and persistence diagrams) *without* prior vectorization. For this, we contribute a novel layer model, which takes a set of merge trees on its input and produces another

set of merge trees on its output, along with their coordinates in the layer's parametrization. This results in superior accuracy (Section 5.5.2) and interpretability (Section 5.4.2). We contribute an algorithm for the optimization of this network (Section 5.3). We believe this contribution to be of independent interest.

3. *An application to data reduction:* We describe how to adapt previous work (Chapter 4) to our novel non-linear framework, in the context of data reduction applications (Section 5.4.1). Specifically, the merge trees of the input ensemble are significantly compressed, by solely storing the final decoding layer of the network, as well as the coordinates of the input trees in this layer. We illustrate the interest of our approach with comparisons to linear encoding (Chapter 4) in the context of feature tracking and ensemble clustering applications.

4. *An application to dimensionality reduction:* We describe how to adapt previous work (Chapter 4) to our novel non-linear framework, in the context of dimensionality reduction applications (Section 5.4.2). Specifically, each tree of the ensemble is embedded as a point in a planar view, based on its coordinates in our auto-encoder's latent space. To illustrate the versatility of our framework, we introduce two penalty terms, to improve the preservation of clusters and distances between merge trees.

5. *Implementation:* We provide a C++ implementation of our algorithms that can be used for reproducibility purposes.

5.2 Formulation

This section describes our novel extension of the classical auto-encoder neural network architecture to the Wasserstein metric space of merge trees, with the novel notion of *Merge Tree Wasserstein Auto-Encoder* (MT-WAE). First, we describe a geometric interpretation of the classical auto-encoders (Section 5.2.1), which we call in the following *Euclidean Auto-Encoders* (EAE). Next, we describe how to generalize each geometrical tool used in EAE (Section 5.2.1) to the Wasserstein metric space of merge trees (Section 5.2.2). Finally, once these tools are available, we formalize our notion of MT-WAE with a novel neural network architecture (Section 5.2.3), for which we document an optimization algorithm in Section 5.3.

5.2.1 An Interpretation of Euclidean Auto-Encoders

Let $P = \{p_1, p_2, \dots, p_N\}$ be a point set in a Euclidean space $\mathbb{R}^d$ (Figure 5.2a). The goal of Euclidean Auto-Encoders (EAE) is to define a d'-dimensional parameterization of P (with $d' \leq d$) which describes well the data (which enables its accurate *reconstruction*). Let $B_1 = \{b_1, b_2, \dots, b_{d'}\}$ be a basis of linearly independent vectors in $\mathbb{R}^d$ (Figure 5.2a). B_1 can be written in the form of a $d \times d'$ matrix, for which each of the d' columns is a vector of the basis. Then, one can express the coordinates $\psi_1(p_i) \in \mathbb{R}^{d'}$ of each point $p_i \in P$ with the basis B_1:

$$\psi_1(p_i) = \arg\min_{\alpha^i} ||p_i - B_1\alpha^i||_2^2 + o_1, \tag{5.1}$$

where o_1 is an *offset* vector of $\mathbb{R}^{d'}$, and where $\alpha^i \in \mathbb{R}^{d'}$ can be seen as a set of d' coefficients, to apply on the d' vectors of the basis B_1 to best estimate p_i. Note that Equation 5.1 can be re-written as a linear transformation:

$$\psi_1(p_i) = B_1^+ p_i + o_1, \tag{5.2}$$

where B_1^+ is the pseudoinverse of the matrix B_1 (Figure 5.2a-b).

Given this new parameterization ψ_1, one can estimate a *reconstruction* of the point p_i in $\mathbb{R}^d$, noted $\widehat{p_i}$. For this, let us consider another, similar, linear transformation ψ_2 (Figure 5.2c-d), defined respectively to a second basis B_2 (given as a $d' \times d$ matrix) and a second offset vector o_2 (in $\mathbb{R}^d$). Then, the reconstruction $\widehat{p_i}$ of each point p_i is given by:

$$\widehat{p_i} = \psi_2 \circ \psi_1(p_i) = B_2^+(B_1^+ p_i + o_1) + o_2.$$

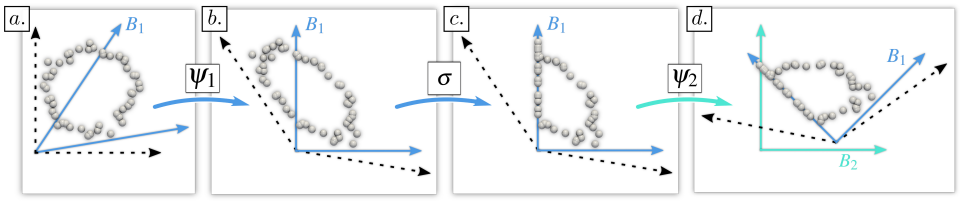


Figure 5.2 – *Geometric interpretation of Euclidean Auto-Encoders (EAE, Section 5.2.1). In its simplest form (one encoding and one decoding layer, $d' = 2$), an EAE can be viewed as the composition of a linear transformation ψ_1 $\big((a)$ to $(b)\big)$ defined with respect to a first basis B_1, followed by a non-linearity σ $\big($here* ReLU, *(b) to $(c)\big)$, followed by a second linear transformation ψ_2 $\big((c)$ to $(d)\big)$ defined with respect to a second basis B_2. Specifically, both ψ_1 and ψ_2 are optimized (via the optimization of B_1 and B_2) to minimize the reconstruction error between the input (a) and the output (d). In the case where σ is the identity, this reconstruction optimization is equivalent to Principal Component Analysis [BK88].*

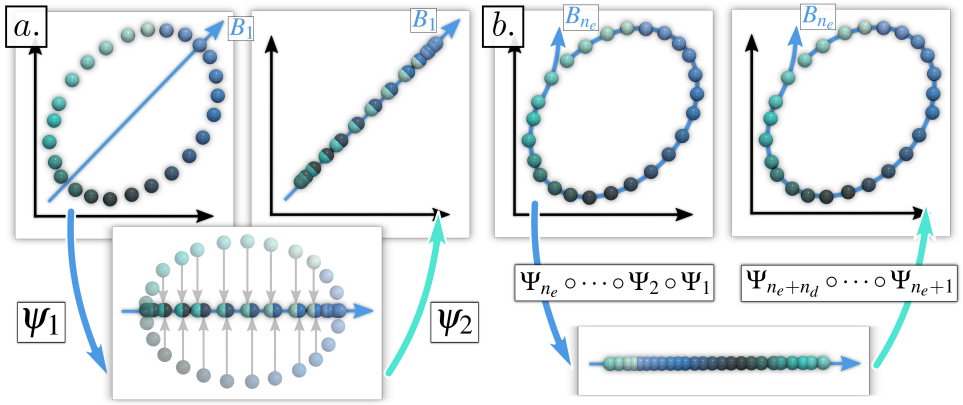


Figure 5.3 – *Comparison between PCA* (a) *and EAE* (b) *for the* 1*-dimensional encoding of a* 2*D point set sampling a* 1*-manifold (color: rotation angle). In its latent space* $\big((a)$*, bottom*$\big)$*, PCA linearly projects the input points to a line, hence interleaving points from the upper and lower parts of the circle. This results in a poor reconstruction* $\big((a)$*, right*$\big)$*, where points are interleaved along the axis* B_1*. In contrast, EAE optimizes a composition of non-linear transformations, which consistently* unwraps *the circle onto a line in its latent space* $\big((b)$*, bottom*$\big)$*, while nicely preserving the intrinsic parameterization of the circle (rotation angle). This results in an accurate reconstruction* $\big((b)$*, right*$\big)$*: the embedding of the axis* B_{n_e} *in the data defines a faithful* 1*-dimensional parameterization of the circle.*

To get an accurate reconstruction $\widehat{p_i}$ (Figure 5.2d) for all the points $p_i \in P$, one needs to optimize both ψ_1 and ψ_2, to minimize the following data fitting energy:

$$E_{L_2} = \sum_{i=1}^{N} ||p_i - \widehat{p_i}||_2^2 = \sum_{i=1}^{N} ||p_i - \psi_2 \circ \psi_1(p_i)||_2^2. \tag{5.3}$$

As discussed by Bourlard and Kamp [BK88], this formulation is a generalization of Principal Component Analysis (PCA) [Pea01], a seminal statistical tool for variability analysis. However, PCA assumes that the input point cloud can be faithfully approximated via linear projections. As shown in Figure 5.3, this hypothesis can be easily challenged in practice. This motivates a non-linear generalization of PCA, capable of faithfully approximating point clouds exhibiting non-linear structures (Figure 5.3).

Specifically, to introduce non-linearity, the above linear transformations ψ are typically composed with a non-linear function σ, called *activation function*, such that the transformation of each point p_i, noted $\Psi(p_i)$, is now given by: $\Psi(p_i) = \sigma\big(\psi(p_i)\big)$. For example, the rectifier activation function (*"ReLU"*) will take the j^{th} coordinate of each data point (i.e. $\big(\psi(p_i)\big)_j$) and *snap* it to zero if it is negative (Figure 5.2b-c). We call the above non-linear transformation Ψ a *transformation layer*. It is characterized by its own vector basis B and its own offset vector o.

Next, to faithfully approximate complicated non-linear input distributions, the above transformation layer is typically composed with a number $(n_e + n_d)$ of other transformation layers, defined similarly. Then, the initial data fitting energy (Equation 5.3) can now be generalized into:

$$E_{L_2} = \sum_{i=1}^{N} ||p_i - \Psi_{n_e+n_d} \circ \cdots \circ \Psi_{n_e+1} \circ \Psi_{n_e} \circ \cdots \circ \Psi_2 \circ \Psi_1(p_i)||_2^2, \qquad (5.4)$$

where each transformation layer Ψ_k is associated with its own d_k-dimensional vector basis B_k and its offset vector o_k. Then, the notion of *Auto-Encoder* is a specific instance of the above formulation, with:

1. $d_1 > d_2 > \cdots > d_{n_e}$, and
2. $d_{n_e} < d_{n_e+1} < \cdots < d_{n_e+n_d} = d$, and
3. $\sigma_{n_e+n_d}$ is the identity.

Specifically, n_e and n_d respectively denote the number of *Encoding* and *Decoding* transformation layers, while d_{n_e} is the dimension of the so-called *latent space*. In practice, d_{n_e} is typically chosen to be much smaller than the input dimensionality (d), for non-linear dimensionality reduction purposes (each input point p_i is then represented in d_{n_e} dimensions, according to its coordinates in B_{n_e}, noted $\alpha^i_{n_e} \in \mathbb{R}^{d_{n_e}}$).

Equation 5.4 defines an optimization problem whose variables (the $n_e + n_d$ bases and offset vectors) can be efficiently optimized (e.g. with gradient descent [KB15]) by composing the transformation layers within a neural network. Then, the gradient ∇E_{L_2} of E_{L_2} can be estimated by exploiting the automatic differentiation capabilities of modern neural network implementations [PGM+19], themselves based on the application of the chain-rule derivation on the above composition of transformation layers.

5.2.2 From EAE to MT-WAE

When the input data is not given as a point cloud in a Euclidean space (Section 5.2.1) but as an abstract set equipped with a metric, the above EAE formulation needs to be extended. For this, we redefine in this section the low-level geometrical tools used in EAE (Section 5.2.1), but within the context of the Wasserstein metric space $\mathbb{B}$ (Chapter 3). In particular, we formalize the following notions:

1. BDT vector (Figure 5.4b);

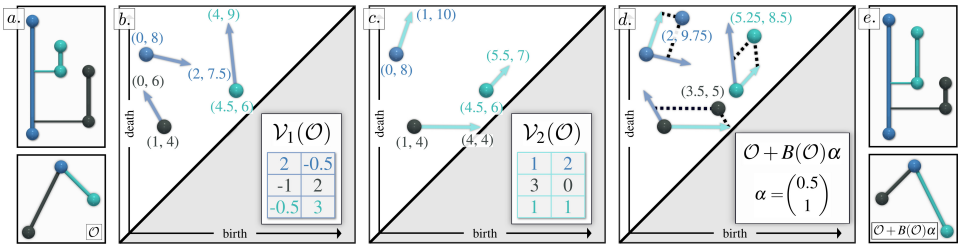


Figure 5.4 – *Low-level geometrical tools in the Wasserstein metric space of merge trees (Section 5.2.2). Given an origin MT and its BDT $\mathcal{O}$ (counting $|\mathcal{O}|$ nodes (a)), a BDT vector $\mathcal{V}_1(\mathcal{O})$ is defined in the birth/death space as a concatenation of $|\mathcal{O}|$ 2D vectors (blue arrows (b)). Given a second BDT vector $\mathcal{V}_2(\mathcal{O})$ (cyan arrows (c)), a basis $B(\mathcal{O})$ can be defined $((d)$, here with $d' = 2)$. For a given set of coefficients $\alpha \in \mathbb{R}^{d'}$, a new merge tree and its BDT (e) can be reconstructed by applying a sum of 2D displacements $\sum_{j=1}^{j=d'} \alpha_j (\mathcal{V}_j(\mathcal{O}))_i$ to each branch b_i of $\mathcal{O}$ $((d)$, black dashes$)$.*

2. BDT basis (Figure 5.4d);
3. BDT basis projection (Figure 5.4d);
4. BDT transformation layer (Figure 5.5b).

***(1)* BDT vector:** Given a BDT $\mathcal{B}$ with $|\mathcal{B}|$ branches, a *BDT vector* $\mathcal{V}(\mathcal{B}) \in \mathbb{B}$ is a vector in $\mathbb{R}^{2|\mathcal{B}|}$, which maps each branch $b \in \mathcal{B}$ to a new location in the 2D birth/death space. $\mathcal{B}$ is the origin of $\mathcal{V}(\mathcal{B})$. This is illustrated for example in Figure 5.4b), where the branches of a given merge tree (Figure 5.4a) are displaced in the birth-death plane (light blue vectors from the spheres of matching color).

***(2)* BDT basis:** Given an origin BDT $\mathcal{O}$, a d'-dimensional basis of BDT vectors, noted $B(\mathcal{O})$, is a set $\{\mathcal{V}_1(\mathcal{O}), \mathcal{V}_2(\mathcal{O}), \ldots, \mathcal{V}_{d'}(\mathcal{O})\}$ of d' linearly independent BDT vectors, having for common origin $\mathcal{O}$. This is shown in Figure 5.4d), where two BDT vectors (from Figure 5.4b), and Figure 5.4c), blue and green arrows) are combined into a basis. Section 5.3.4 clarifies the basis initialization by our approach.

***(3)* BDT basis projection:** Given an arbitrary BDT $\mathcal{B}$, its *projection error* $e(\mathcal{B})$ in a d'-dimensional BDT basis $B(\mathcal{O})$ is:

$$e(\mathcal{B}) = \min_{\alpha} \left(W_2^{\mathcal{T}} \left(\mathcal{B}, \mathcal{O} + B(\mathcal{O})\alpha \right) \right)^2, \tag{5.5}$$

where $\alpha \in \mathbb{R}^{d'}$ can be interpreted as a set of coefficients to apply on the d' BDT vectors of $B(\mathcal{O})$ to best estimate $\mathcal{B}$. Then, the projection of $\mathcal{B}$ in

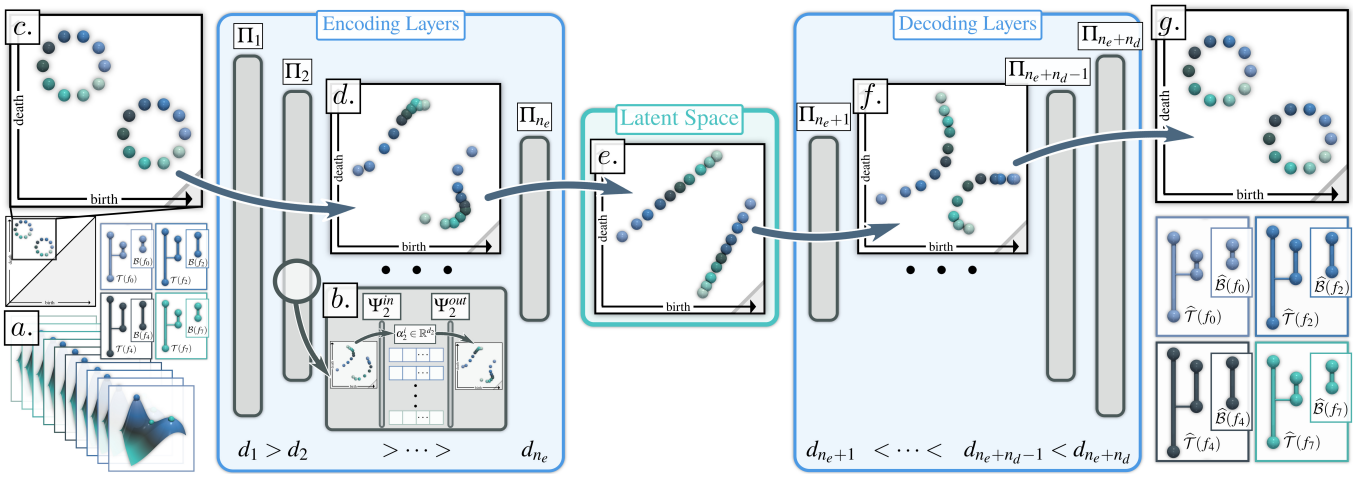


Figure 5.5 – ***Overview:*** *given an input ensemble, with its merge trees and BDTs* (a)*, our Wasserstein Auto-Encoder of Merge Trees (MT-WAE) optimizes an auto-encoder (for this example,* $n_e = n_d = 2$*) where each layer* Π_k *natively processes BDTs (without pre-vectorization). Specifically, each layer* Π_k *can be interpreted as a local auto-encoder* (b)*, where an* input *sub-layer* Ψ_k^{in} *transforms the input BDT* $\mathcal{B}_{k-1}(f_i)$ *into a set of coefficients* $\alpha_k^i \in \mathbb{R}^{d_k}$ *and where an* output *sub-layer* Ψ_k^{out} *transforms these coefficients back into a valid BDT* $\mathcal{B}_k(f_i)$*. The aggregated views* $\big((c),(d),(e),(f),(g)\big)$*, which overlap all the BDTs in the birth/death space (one color per BDT), show the ability of MT-WAE to progressively* unwrap *non-linear structures (circles) as the BDTs progress down the network, resulting in faithful local parameterizations in latent space* $\big((e)$*, the individual angular parameterizations of the circles are well preserved*$\big)$*, as well as accurate reconstructions* (g)*. This native support of BDTs results in a superior accuracy (Section 5.5.2) and an improved interpretability: individual features can now be tracked as they traverse the network, enabling new visual analysis capabilities (Section 5.4.2).*

$B(\mathcal{O})$, noted $\psi(\mathcal{B})$, is given by the optimal coefficients α associated to the projection error of $\mathcal{B}$ (Equation 5.5):

$$\psi(\mathcal{B}) = \arg\min_{\alpha} \Big(W_2^{\mathcal{T}}\big(\mathcal{B}, \mathcal{O} + B(\mathcal{O})\alpha\big)\Big)^2. \tag{5.6}$$

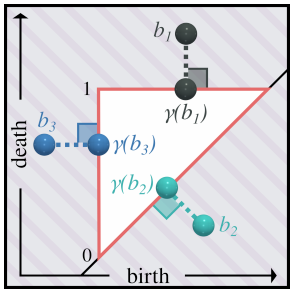
The above equation is a re-interpretation of Equation 5.1 (Section 5.2.1), where the L_2 norm (used for EAE) is replaced by the Wasserstein distance $W_2^{\mathcal{T}}$. This projection procedure is illustrated in Figure 5.4d). Given a BDT basis $B(\mathcal{O})$ (blue and green arrows), the projection of an arbitrary BDT $\mathcal{B}$ is the linear combination $\psi(\mathcal{B})$ of the BDT vectors of the basis which minimizes its Wasserstein distance $W_2^{\mathcal{T}}$ to $\mathcal{B}$. In the birth-death space, the branches of the basis origin $\mathcal{O}$ are represented by the colored spheres at the intersection between the blue and green arrows, while the branches of $\psi(\mathcal{B})$ are represented by the other spheres of matching color. In this example, the linear combination $\psi(\mathcal{B})$ which minimizes its distance to $\mathcal{B}$ is obtained for the coefficients $\alpha = (0.5, 1)$. Then, to go from $\mathcal{O}$ to $\psi(\mathcal{B})$, each branch b_i of $\mathcal{O}$ is displaced by 0.5 $\mathcal{V}_1(\mathcal{O})$ (intersection between the dashed lines and the blue arrows) and then by 1 $\mathcal{V}_2(\mathcal{O})$ (intersection between the dashed lines and the green arrows). The resulting merge tree is shown in Figure 5.4e). In contrast to the merge tree of the basis origin

(Figure 5.4a)), the persistence of the cyan branch has increased while that of the black branch has decreased.

***(4)* BDT transformation layer:** Once the above tools have been formalized, we can introduce the novel notion of *BDT transformation layer*. Similarly to the Euclidean case (Section 5.2.1), a non-linear activation function σ (in our case, *Leaky ReLU*) can be composed with the above projection, yielding the new function $\Psi(\mathcal{B}) = \sigma\big(\psi(\mathcal{B})\big)$. Note that, at this stage, $\Psi(\mathcal{B})$ can be interpreted as a set of coefficients to apply on the BDT basis $B(\mathcal{O})$ to best estimate $\mathcal{B}$. In other words, $\Psi(\mathcal{B})$ is not a BDT yet, but simply a set of coefficients, which can be used later to reconstruct a BDT. Thus, a second transformation needs to be considered, to transform the set of coefficients $\Psi(\mathcal{B})$ back into a BDT. Then, we define the notion of *BDT transformation layer*, noted $\Pi(\mathcal{B})$, as the composition $\Pi(\mathcal{B}) = \Psi^{out} \circ \Psi^{in}(\mathcal{B})$ (Figure 5.5b):

$$\Psi^{in}(\mathcal{B}) = \sigma\Big(\arg\min_{\alpha} \Big(W_2^{\mathcal{T}}\big(\mathcal{B}, \mathcal{O}^{in} + B^{in}(\mathcal{O}^{in})\alpha\big)\Big)^2\Big)$$
$$\Psi^{out}(\alpha) = \gamma\big(\mathcal{O}^{out} + B^{out}(\mathcal{O}^{out})\alpha\big),$$

where $\gamma(\mathcal{B})$ is a projection which transforms $\mathcal{B}$ into a *valid* BDT, i.e. which respects the *Elder rule* (Section 2.4). Given a branch $b \in \mathcal{B}$, γ enforces that: $\gamma(b)_x < \gamma(b)_y$ and $[\gamma(b)_x, \gamma(b)_y] \subseteq [0,1]$ (inset).



BDT transformation layers can be seen as *local* auto-encoders (Figure 5.5b): the first step Ψ^{in} converts an input BDT into a set of coefficients with a basis projection and a non-linearity, while the second step Ψ^{out} converts these coefficients into a BDT. Note that each BDT transformation layer is associated with its own input *and* output d'-dimensional bases $B^{in}(\mathcal{O}^{in})$ and $B^{out}(\mathcal{O}^{out})$.

The processing of an ensemble of BDTs by a BDT transformation layer is illustrated in Figure 5.5. Specifically, Figure 5.5c) shows a zoom of the birth-death space, where all the BDTs have been aggregated (one color per BDT, each BDT has two branches, hence two patterns appear, one circle per branch). The left inset of Figure 5.5b) shows the non-linearly transformed set of BDTs after the first BDT transformation layer, Π_1. Next, the first step Ψ_2^{in} of the next BDT layer Π_2 converts each BDT $\Pi_1\big(\mathcal{B}(f_i)\big)$ into a set of coefficients α_2^i. Finally, the second step Ψ_2^{out} of the layer Π_2 converts

each of these set of coefficients into a new, non-linearly transformed BDT $\Pi_2 \circ \Pi_1(\mathcal{B}(f_i))$ (Figure 5.5b), right inset).

5.2.3 MT-WAE Formulation

Now that the above geometrical tools have been introduced for the Wasserstein metric space of merge trees, we can now formulate MT-WAE by direct analogy to the Euclidean case (Section 5.2.1). Given a set $\mathcal{S}_\mathcal{B} = \{\mathcal{B}(f_1), \ldots, \mathcal{B}(f_N)\}$ of input BDTs, a MT-WAE is a composition of BDT transformation layers $\Pi_k(\mathcal{B})$ (Figure 5.5), minimizing the following energy:

$$E_{W_2^\mathcal{T}} = \sum_{i=1}^{N} \left(W_2^\mathcal{T} \left(\mathcal{B}(f_i), \Pi_{n_e+n_d} \circ \cdots \circ \Pi_{n_e} \circ \cdots \circ \Pi_1(\mathcal{B}(f_i)) \right) \right)^2, \quad (5.7)$$

where each BDT transformation layer Π_k is associated with its own d_k-dimensional input *and* output vector bases $B_k^{in}(\mathcal{O}_k^{in})$ and $B_k^{out}(\mathcal{O}_k^{out})$. Moreover, the dimensions of the successive bases are chosen such that:

1. $d_1 > d_2 > \cdots > d_{n_e}$, and
2. $d_{n_e} < d_{n_e+1} < \cdots < d_{n_e+n_d}$,

where n_e and n_d denote the number of *Encoding* and *Decoding* layers and where d_{n_e} is the dimension of the MT-WAE latent space. Equation 5.7 is a direct analog to the classical EAE (Equation 5.4): the standard transformation layers Ψ_k have been replaced by BDT transformation layers Π_k and the L_2 norm by the distance $W_2^\mathcal{T}$.

Figure 5.5 illustrates a network of BDT transformation layers optimized on a synthetic ensemble (our optimization algorithm is described in Section 5.3). As mentioned in the previous section, each input BDT (spheres in the aggregated birth-death views, one color per BDT) is non-linearly transformed by the BDT transformation layers (see for instance Figure 5.5d) and Figure 5.5f)). As a result, in this example, the BDT transformation layers progressively *unwrap* the non-linear structures in the birth-death plane (circles in Figure 5.5c)) as the BDTs traverse the network down to the latent space (Figure 5.5e)), where the resulting layout in the birth-death plane (line segments) manages to faithfully encode the purposely designed parameterization of the ensemble: the order of rotation angles (colors) is well preserved along the segments in latent space.

5.3 Algorithm

This section presents our algorithm for the minimization of Equation 5.7.

5.3.1 Overview

Algorithm 2 provides an overview of our main algorithm. The set of optimization variables, noted θ, includes the $(n_e + n_d)$ input *and* output BDT bases, along with their origins (line 2). The optimization of these variables follows the standard overall procedure for the optimization of a neural network. First, θ is initialized (Section 5.3.4). Then, the input ensemble of BDTs $\mathcal{S}_\mathcal{B}$ (Figure 5.5a) traverses the network to generate a reconstructed ensemble of BDTs, noted $\widehat{\mathcal{S}_\mathcal{B}}$ (line 7 and Figure 5.5g). This is traditionally denoted as the *Forward* propagation (Section 5.3.5). Given $\widehat{\mathcal{S}_\mathcal{B}}$, the energy $E_{W_2^\mathcal{T}}(\theta)$ (Equation 5.7) can be evaluated and its gradient $\nabla E_{W_2^\mathcal{T}}(\theta)$ can be estimated by automatic differentiation (based on the application of the

Algorithm 2 Wasserstein Auto-Encoder (algorithm overview).

Input: Set of BDTs $\mathcal{S}_\mathcal{B} = \{\mathcal{B}(f_1), \ldots, \mathcal{B}(f_N)\}$.

Output1: Set of $(n_e + n_d)$ input origins $\theta_{\mathcal{O}^{in}} = \{\mathcal{O}_1^{in}, \mathcal{O}_2^{in}, \ldots, \mathcal{O}_{n_e+n_d}^{in}\}$;

Output2: Set of $(n_e + n_d)$ input bases:
$\theta_{B^{in}} = \{B_1^{in}(\mathcal{O}_1^{in}), B_2^{in}(\mathcal{O}_2^{in}), \ldots, B_{n_e+n_d}^{in}(\mathcal{O}_{n_e+n_d}^{in})\}$;

Output3: For each of the input BDTs ($i \in \{1, 2, \ldots, N\}$), set of $(n_e + n_d)$ input coefficients: $\theta_\alpha^i = \{\alpha_1^i \in \mathbb{R}^{d_1}, \alpha_2^i \in \mathbb{R}^{d_2}, \ldots, \alpha_{n_e+n_d}^i \in \mathbb{R}^{d_{n_e+n_d}}\}$;

Output4: Set of $(n_e + n_d)$ output origins:
$\theta_{\mathcal{O}^{out}} = \{\mathcal{O}_1^{out}, \mathcal{O}_2^{out}, \ldots, \mathcal{O}_{n_e+n_d}^{out}\}$;

Output5: Set of $(n_e + n_d)$ output bases:
$\theta_{B^{out}} = \{B_1^{out}(\mathcal{O}_1^{out}), B_2^{out}(\mathcal{O}_2^{out}), \ldots, B_{n_e+n_d}^{out}(\mathcal{O}_{n_e+n_d}^{out})\}$.

1: // Overall set of optimization variables
2: $\theta \leftarrow \{\theta_{\mathcal{O}^{in}}, \theta_{B^{in}}, \theta_{\mathcal{O}^{out}}, \theta_{B^{out}}\}$;
3: // Initialization of the optimization variables (Section 5.3.4)
4: Initialize(θ);
5: **while** $E_{W_2^\mathcal{T}}(\theta)$ decreases **do**
6: // Forward propagation of the BDT ensemble $\mathcal{S}_\mathcal{B}$ (Section 5.3.5)
7: $\widehat{\mathcal{S}_\mathcal{B}} \leftarrow$ Forward($\mathcal{S}_\mathcal{B}, \theta$);
8: // Backward propagation (Section 5.3.6)
9: $\theta \leftarrow$ Backward($\mathcal{S}_\mathcal{B}, \widehat{\mathcal{S}_\mathcal{B}}$);
10: **end while**

chain-rule on the composition of BDT transformation layers). Given the gradient $\nabla E_{W_2^{\mathcal{T}}}(\theta)$, a step of gradient descent can be achieved to update the optimization variables θ (line 9). This is traditionally denoted as the *Backward* propagation (Section 5.3.6). These two steps are then iterated until the energy stops decreasing (in practice until it decreases by less than 1% between two iterations). For our implementation, we relied on the *PyTorch* neural network framework [PGM+19] for automatic differentiation, and we used its *Adam* solver for gradient descent [KB15].

5.3.2 Basis Projection

We start by describing an efficient *Assignment/Update* algorithm for the projection of a BDT $\mathcal{B}$ into a BDT basis $B(\mathcal{O})$ (Equation 5.6), as it is a core geometrical component used throughout our approach.

The purpose of the projection (Equation 5.6) is to find a set of coefficients $\alpha \in \mathbb{R}^{d'}$ to apply on the d' BDT vectors of $B(\mathcal{O})$ to best estimate $\mathcal{B}$. The geodesic analysis of merge trees (Chapter 4) faces a similar issue, but its formulation (restricting α to $[0,1]^{d'}$) allows for an iterative, brute-force optimization. Here, we introduce a more general and efficient strategy.

This discussion describes the case where the optimal assignment ϕ_* of the assignment step is a bijection between off-diagonal points of the 2D birth/death plane. Section 5.3.3 details the general case, where ϕ_* may send points of $\mathcal{B}$ to the diagonal of $\widehat{\mathcal{B}}$, and reciprocally.

***(1)* Assignment step:** Let us assume that we are given an initial set of coefficients α. Then, the *estimation* $\widehat{\mathcal{B}}$ of $\mathcal{B}$ is given by $\widehat{\mathcal{B}} \leftarrow \mathcal{O} + B(\mathcal{O})\alpha$. The purpose of the assignment step is to refine the evaluation of $W_2^{\mathcal{T}}(\mathcal{B}, \widehat{\mathcal{B}})$. For this, we first compute the optimal assignment ϕ_* between $\mathcal{B}$ and $\widehat{\mathcal{B}}$, w.r.t. Equation 2.3. Then, $W_2^{\mathcal{T}}(\mathcal{B}, \widehat{\mathcal{B}})$ can be re-written as:

$$W_2^{\mathcal{T}}(\mathcal{B}, \widehat{\mathcal{B}}) = \sum_{i=1}^{|\mathcal{B}|} ||b_i - \phi_*(b_i)||_2^2. \tag{5.8}$$

***(2)* Update step:** Given the above estimation $\widehat{\mathcal{B}}$, the goal of the update step is to improve the coefficients α, in order to decrease $W_2^{\mathcal{T}}(\mathcal{B}, \widehat{\mathcal{B}})$. Let $\widehat{\mathcal{B}}'$ be a vector representation of $\widehat{\mathcal{B}}$. Specifically, $\widehat{\mathcal{B}}'$ is a vector in $\mathbb{R}^{2|\widehat{\mathcal{B}}|}$ which concatenates the coordinates in the 2D birth/death plane of each branch b_i of $\widehat{\mathcal{B}}$. $\widehat{\mathcal{B}}'$ can be decomposed into $\mathcal{O}' + \left(B(\mathcal{O}')\right)'\alpha$, where $\left(B(\mathcal{O}')\right)'$ is a $(2|\widehat{\mathcal{B}}|) \times d'$ matrix. Additionally, let $\mathcal{B}'$ be a similar vector representation of $\mathcal{B}$, *but* where the entries have been specifically re-ordered such that, for

each of its 2D entries, we have:

$$(\mathcal{B}')_i = \phi_*^{-1}((\widehat{\mathcal{B}}')_i). \tag{5.9}$$

Intuitively, $\mathcal{B}'$ is a re-ordered vector representation of $\mathcal{B}$, such that its i^{th} entry exactly matches though ϕ_* with the i^{th} entry of $\widehat{\mathcal{B}}'$. Given this vector representation, the Wasserstein distance $W_2^{\mathcal{T}}(\mathcal{B}, \widehat{\mathcal{B}})$ for a fixed optimal assignment ϕ^* (Equation 5.8) can then be re-written as an L_2 norm:

$$W_2^{\mathcal{T}}(\mathcal{B}, \widehat{\mathcal{B}}) = ||\mathcal{B}' - \widehat{\mathcal{B}}'||_2^2. \tag{5.10}$$

Then, given the optimal assignment ϕ_*, the optimal $\alpha_* \in \mathbb{R}^{d'}$ are:

$$\begin{aligned}
\alpha_* &= \arg\min_\alpha ||\mathcal{B}' - \widehat{\mathcal{B}}'||_2^2 \\
\alpha_* &= \arg\min_\alpha ||\mathcal{B}' - \left(\mathcal{O}' + \left(B(\mathcal{O}')\right)'\alpha\right)||_2^2 \\
\alpha_* &= \arg\min_\alpha ||\mathcal{B}' - \mathcal{O}' - \left(B(\mathcal{O}')\right)'\alpha||_2^2.
\end{aligned}$$

Similarly to the Euclidean case (Equation 5.2), it follows then that α_* can be expressed as a function of the pseudoinverse of $\left(B(\mathcal{O}')\right)'$:

$$\alpha_* = \left(\left(B(\mathcal{O}')\right)'\right)^+(\mathcal{B}' - \mathcal{O}'). \tag{5.11}$$

At this stage, the estimation $\widehat{\mathcal{B}}$ can be *updated* with the above optimized coefficients α_*: $\widehat{\mathcal{B}} \leftarrow \mathcal{O} + B(\mathcal{O})\alpha_*$.

The above *Assignment/Update* sequence is then iterated. Each iteration decreases the *projection error* $e(\mathcal{B})$ constructively: while the *Update* phase (2) optimizes α (Equation 5.5) to minimize the projection error under the current assignment ϕ_*, the next *Assignment* phase (1) further improves (by construction) the assignments (term $W_2^{\mathcal{T}}$ in Equation 5.5), hence decreasing the projection error overall. In our implementation, this iterative algorithm stops after a predefined number of iterations n_{it}.

5.3.3 General Formulation of Basis Projection

Section 5.3.2 presents an *Assignment/Update* algorithm to project an input BDT $\mathcal{B}$ into a given BDT basis $B(\mathcal{O})$.

In the *Assignment* phase, given an initial set of coefficients $\alpha \in \mathbb{R}^{d'}$, the *estimation* $\widehat{\mathcal{B}}$ of $\mathcal{B}$ is given by :

$$\widehat{\mathcal{B}} \leftarrow \mathcal{O} + B(\mathcal{O})\alpha. \tag{5.12}$$

Given this estimation $\widehat{\mathcal{B}}$, the assignment step first evaluates the Wasserstein distance $W_2^{\mathcal{T}}(\mathcal{B}, \widehat{\mathcal{B}})$. For this, the optimal assignment ϕ_* between $\mathcal{B}$ and $\widehat{\mathcal{B}}$ is computed with regard to the Wasserstein distance (Equation 3.4).

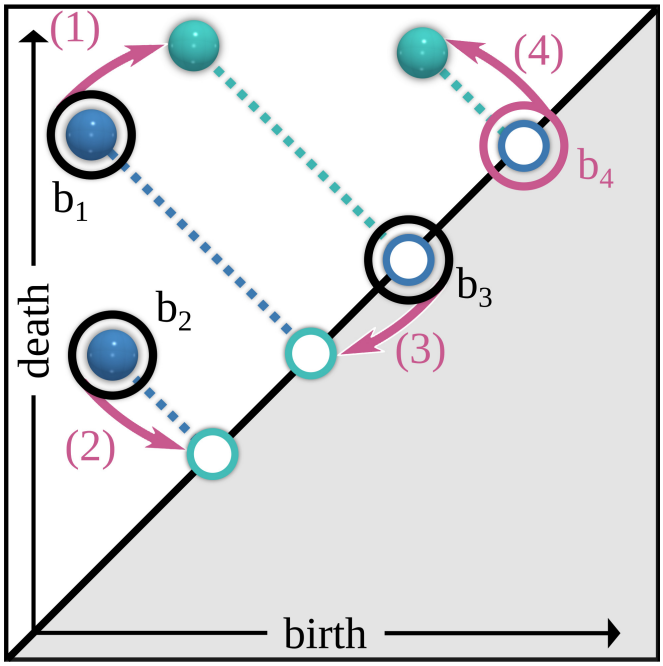


Figure 5.6 – *When considering an assignment (purple arrows) in the 2D birth/death planes between* augmented *BDTs, four cases can occur (purple numbers).* ***Case* (1):** *an* off-diagonal *branch (blue dot) can be mapped to an* off-diagonal *branch (green dot).* ***Case* (2):** *an* off-diagonal *branch (blue dot) can be mapped to an augmented* diagonal *branch (green circle).* ***Case* (3):** *an augmented* diagonal *branch (blue circle) can be mapped to an augmented* diagonal *branch (green circle).* ***Case* (4):** *an augmented* diagonal *branch (blue circle) can be mapped to an* off-diagonal *branch (green dot). Section 5.3.2 covers the cases* (1) *and* (2)*. Section 5.3.3 generalizes this formulation to all cases.*

Then, given the optimal assignment ϕ_*, the Wasserstein distance $W_2^{\mathcal{T}}(\mathcal{B}, \widehat{\mathcal{B}})$ can be re-written as:

$$W_2^{\mathcal{T}}(\mathcal{B}, \widehat{\mathcal{B}}) = \sum_{i=1}^{|\mathcal{B}|} \begin{cases} 0 & \text{if both } b_i \text{ and } \phi_*(b_i) \text{ are on the diagonal,} \\ ||b_i - \phi_*(b_i)||_2^2 & \text{otherwise.} \end{cases} \tag{5.13}$$

The purpose of the subsequent *Update* step is precisely to optimize α in order to minimize the evaluation of $W_2^{\mathcal{T}}(\mathcal{B}, \widehat{\mathcal{B}})$ by the above equation. For this, one needs to isolate from Equation 5.13 all the terms involving α from those which do not.

In the simple case where ϕ_* describes a bijection between off-diagonal points (case covered in Section 5.3.2), no branch of $\mathcal{B}$ depends on α. Then the isolation of the terms involving α is simple: only the branches of $\widehat{\mathcal{B}}$ depend on α (Equation 5.12).

In the more general case, things are a bit more involved. As illustrated in Figure 5.6, the computation of the optimal assignment $\phi_* : \mathcal{B} \to \widehat{\mathcal{B}}$

(purple arrows) implies a pre-processing phase of *augmentation* of the 2D birth/death plane. As described in Section 2.4, $\widehat{\mathcal{B}}$ is augmented with the diagonal projections of the branches of $\mathcal{B}$ (Figure 5.6, green circles) and $\mathcal{B}$ is augmented with the diagonal projections of the branches of $\widehat{\mathcal{B}}$ (Figure 5.6, blue circles). This augmentation phase enables the modeling of the destruction (or creation) of features during the assignment ϕ_* (between the blue and green items, Figure 5.6). Then, the following four cases can occur (Figure 5.6):

Case *(1)*: An *off-diagonal* branch $b_1 \in \mathcal{B}$ is mapped to an *off-diagonal* branch $\phi_*(b_1) \in \widehat{\mathcal{B}}$. Then, the birth/death values of b_1 do not depend on α and only the birth/death values of $\phi_*(b_1)$ do. This corresponds to the case covered by Section 5.3.2.

Case *(2)*: An *off-diagonal* branch $b_2 \in \mathcal{B}$ is mapped to a *diagonal* branch $\phi_*(b_2) \in \widehat{\mathcal{B}}$. Then, the birth/death values of b_2 do not depend on α. Then, this case is also covered by Section 5.3.2.

Case *(3)*: A *diagonal* branch $b_3 \in \mathcal{B}$ is mapped to a *diagonal* branch $\phi_*(b_3) \in \widehat{\mathcal{B}}$. In that case, the ground distance $d_2\big(b_3, \phi_*(b_3)\big)$ (Equation 2.2) is set to zero by convention (first line of Equation 5.13). This models the fact that both b_3 and $\phi_*(b_3)$ are dummy features (with zero persistence) and that their ground distance, which is arbitrary, should not be taken into account in $W_2^{\mathcal{T}}(\mathcal{B}, \widehat{\mathcal{B}})$. Therefore, we simply remove b_3 from $\mathcal{B}$ and $\phi_*(b_3)$ from $\widehat{\mathcal{B}}$. This removal discards this first line of Equation 5.13, which can then be re-written in the general form:

$$W_2^{\mathcal{T}}(\mathcal{B}, \widehat{\mathcal{B}}) = \sum_{i=1}^{|\mathcal{B}|} ||b_i - \phi_*(b_i)||_2^2. \tag{5.14}$$

Then, with this removal, the diagonal-diagonal assignments do not constitute a special case anymore.

Case *(4)*: An *off-diagonal* branch $b_4 \in \mathcal{B}$ is mapped to a *diagonal* branch $\phi_*(b_4) \in \widehat{\mathcal{B}}$. In that case, b_4 turns out to be an *augmented* point of $\mathcal{B}$. In Figure 5.6, these are reported with circles while original (i.e. non-augmented) points are reported with dots. Specifically, b_4 has been precisely inserted such that $b_4 = \Delta\big(\phi_*(b_4)\big)$:

$$(b_4)_x = (b_4)_y = \frac{1}{2}\Big(\big(\phi_*(b_4)\big)_x + \big(\phi_*(b_4)\big)_y\Big).$$

Given Equation 5.12, $\phi_*(b_4)$ can be re-written as:

$$\phi_*(b_4) = o_4 + B(\mathcal{O})\alpha,$$

where o_4 is a branch of $\mathcal{O}$. Then, the birth/death values of b_4 are:

$$(b_4)_x = (b_4)_y = \tfrac{1}{2}\Big(\big(o_4 + B(\mathcal{O})\alpha\big)_x + \big(o_4 + B(\mathcal{O})\alpha\big)_y\Big)$$
$$(b_4)_x = (b_4)_y = \tfrac{1}{2}\big((o_4)_x + (o_4)_y\big) + \tfrac{1}{2}\Big(\big(B(\mathcal{O})\alpha\big)_x + \big(B(\mathcal{O})\alpha\big)_y\Big).$$

Then, it follows that b_4 can be re-written as:

$$b_4 = \Delta(o_4) + \Delta\big(B(\mathcal{O})\alpha\big). \tag{5.15}$$

From Equation 5.15, it is clear that the coordinates of b_4 are dependent on α. In short, this is due to the fact that b_4 has been purposely inserted in $\mathcal{B}$ as the diagonal projection of a branch of $\widehat{\mathcal{B}}$ which, itself, depends on α. Thus, to account for this special case, we need to further isolate the terms of Equation 5.15 depending on α (i.e. $\Delta\big(B(\mathcal{O})\big)$, as described next.

Similarly to Section 5.3.2, let $\widehat{\mathcal{B}}'$ be a vector representation of $\widehat{\mathcal{B}}$. Specifically $\widehat{\mathcal{B}}'$ is a vector in $\mathbb{R}^{2|\widehat{\mathcal{B}}|}$ which concatenates the coordinates in the birth/death plane of each branch b_i of $\widehat{\mathcal{B}}$. $\widehat{\mathcal{B}}'$ can be decomposed into $\mathcal{O}' + \big(B(\mathcal{O}')\big)'\alpha$, where $\big(B(\mathcal{O}')\big)'$ is a $(2|\widehat{\mathcal{B}}|) \times d'$ matrix. Also, let $\mathcal{B}'$ be a similar vector representation of $\mathcal{B}$, *but* where the entries have been specifically re-ordered such that, for each of its 2D entries, we have:

$$(\mathcal{B}')_i = \phi_*^{-1}\big((\widehat{\mathcal{B}}')_i\big).$$

Then $\mathcal{B}'$ can be decomposed as a sum of two vectors of $\mathbb{R}^{2|\widehat{\mathcal{B}}|}$:

$$\mathcal{B}' = \mathcal{B}'_1 + \mathcal{B}'_2,$$

such that $\mathcal{B}'_1$ has all its entries set to 0 except those covered by the above cases *(1)* and *(2)* (Figure 5.6), and that $\mathcal{B}'_2$ has all its entries set to 0 except those covered by the above case *(4)* (purple, Figure 5.6).

Given Equation 5.15, each non-zero entry i of $\mathcal{B}'_2$ can be re-written as:

$$(\mathcal{B}'_2)_i = \Delta(o_i) + \Delta\bigg(\Big(\big(B(\mathcal{O})\big)'\alpha\Big)_i\bigg), \tag{5.16}$$

where o_i is the i^{th} entry of $\mathcal{O}$. Then, the vector $\mathcal{B}'_2$ can be further decomposed as follows:

$$\mathcal{B}'_2 = \mathcal{B}'_3 + \mathcal{B}'_4,$$

such that each non-zero entry i of $\mathcal{B}'_3$ is equal to $\Delta(o_i)$ and each non-zero entry i of $\mathcal{B}'_4$ is equal to $\Delta\bigg(\Big(\big(B(\mathcal{O})\big)'\alpha\Big)_i\bigg)$.

Let B'_2 be a $(2|\widehat{\mathcal{B}}|) \times d'$ matrix such that:

$$\mathcal{B}'_4 = B'_2\alpha.$$

At this stage, we have:

$$\mathcal{B}' = \mathcal{B}'_1 + \mathcal{B}'_3 + \mathcal{B}'_4 = \mathcal{B}'_1 + \mathcal{B}'_3 + B'_2\alpha. \tag{5.17}$$

At this point, we managed to isolate the terms in $\mathcal{B}'$ which are dependent on α (i.e. $B'_2\alpha$). Then, similarly to Section 5.3.2, for a fixed optimal assignment ϕ_*, the Wasserstein distance $W_2^{\mathcal{T}}(\mathcal{B}, \widehat{\mathcal{B}})$ can be re-written as an L_2 norm:

$$W_2^{\mathcal{T}}(\mathcal{B}, \widehat{\mathcal{B}}) = ||\mathcal{B}' - \widehat{\mathcal{B}}'||_2^2.$$

Then, given ϕ_*, by using Equation 5.17, the optimal $\alpha_* \in \mathbb{R}^{d'}$ are:

$$\begin{aligned}
\alpha_* &= \arg\min_\alpha ||\mathcal{B}' - \widehat{\mathcal{B}}'||_2^2 \\
\alpha_* &= \arg\min_\alpha ||\mathcal{B}'_1 + \mathcal{B}'_3 + B'_2\alpha - \Big(\mathcal{O}' + \big(B(\mathcal{O}')\big)'\alpha\Big)||_2^2 \\
\alpha_* &= \arg\min_\alpha ||\mathcal{B}'_1 + \mathcal{B}'_3 - \mathcal{O}' - \Big(\Big(\big(B(\mathcal{O}')\big)' - B'_2\Big)\alpha\Big)||_2^2.
\end{aligned}$$

Then, similarly to the Euclidean case (Equation 5.2), it follows then that α_* can be expressed as a function of the pseudoinverse of $\Big(\big(B(\mathcal{O}')\big)' - B'_2\Big)$:

$$\alpha_* = \Big(\big(B(\mathcal{O}')\big)' - B'_2\Big)^+(\mathcal{B}'_1 + \mathcal{B}'_3 - \mathcal{O}'). \tag{5.18}$$

In short, the general expression of the optimal coefficients α_* (Equation 5.18) is a generalization of Equation 5.11, such that the branches of $\mathcal{B}$ dependent on α (case *(4)*) have been integrated within the pseudoinverse operation.

5.3.4 Initialization

Now that we have introduced the core low-level procedure of our approach (Section 5.3.2), we can detail the initialization step of our framework, which consists in identifying a relevant initial value for the overall optimization variable θ (line 4, Algorithm 2). The BDT transformation layers Π_k are initialized one after the other, i.e. for increasing values of k.

***(1)* Input initialization:** For each BDT transformation layer Π_k, its input origin $\mathcal{O}_k^{in}$ is initialized as the Wasserstein barycenter $\mathcal{B}_*$ [PVDT22] of the BDTs on its input. Next, the first vector of B_k^{in}, is given by the optimal assignment (w.r.t. Equation 2.3) between $\mathcal{O}_k^{in}$ and the layer's input BDT $\mathcal{B}$

which maximizes $W_2^{\mathcal{T}}(\mathcal{O}_k^{in}, \mathcal{B})$, i.e. which induces the worst projection error $e(\mathcal{B})$ (Equation 5.5) given an empty basis. Next, the remaining $(d_k - 1)$ vectors of B_k^{in} are initialized one after the other, by including at each step the vector formed by the optimal assignment between $\mathcal{O}_k^{in}$ and the layer's input BDT $\mathcal{B}$ which induces the maximum projection error $e(\mathcal{B})$ (Equation 5.5), given the already initialized vectors. Note that this step makes an extensive usage of the projection procedure introduced in Section 5.3.2. Finally, if the dimension d_k of Π_k is greater than the number of input BDTs, the remaining vectors are initialized randomly, with a controlled norm (set to the mean of the already initialized vectors).

***(2)* Output initialization:** For each BDT transformation layer Π_k, its output origin $\mathcal{O}_k^{out}$ and basis B_k^{out} are initialized as random linear transformations of its input origin and basis. Specifically, let W be a random matrix of size $(2|\mathcal{O}_k^{out}| \times 2|\mathcal{O}_k^{in}|)$. Given the vector representation $\mathcal{O}_k^{in\prime}$ of $\mathcal{O}_k^{in}$ (see Section 5.3.2), we initialize $\mathcal{O}_k^{out}$ such that: $\mathcal{O}_k^{out\prime} \leftarrow W\mathcal{O}_k^{in\prime}$. Similarly, the output basis of Π_k is initialized such that: $B_k^{out\prime} \leftarrow WB_k^{in\prime}$.

5.3.5 Forward Propagation

Algorithm 3 presents the main steps of our forward propagation.

Algorithm 3 Forward propagation in our Wasserstein Auto-Encoder.

Input1: Set of input BDTs $\mathcal{S}_{\mathcal{B}} = \{\mathcal{B}(f_1), \ldots, \mathcal{B}(f_N)\}$.

Input2: Current value of the overall optimization variable θ.

Output: Set of reconstructed BDTs $\widehat{\mathcal{S}_{\mathcal{B}}} = \{\widehat{\mathcal{B}(f_1)}, \ldots, \widehat{\mathcal{B}(f_N)}\}$.

for $\mathcal{B} \in \mathcal{S}_{\mathcal{B}}$ **do**
  $\mathcal{B}_0 \leftarrow \mathcal{B}$.
  // For each BDT transformation layer Π_k.
  for $k \in \{1, 2, \ldots, n_e + n_d\}$ **do**
    // Section 5.3.2 and Section 5.3.3.
    $\psi_k^{in}(\mathcal{B}_{k-1}) \leftarrow$basisProjection$(\mathcal{B}_{k-1}, \mathcal{O}_k^{in}, B_k^{in})$.
    // Section 5.2.2.
    $\Psi_k^{in}(\mathcal{B}_{k-1}) \leftarrow \sigma\big(\psi_k^{in}(\mathcal{B}_{k-1})\big)$.
    $\mathcal{B}_k \leftarrow \Psi_k^{out}\big(\Psi_k^{in}(\mathcal{B}_{k-1})\big) = \gamma\big(\mathcal{O}_k^{out} + B_k^{out}(\mathcal{O}_k^{out})\Psi_k^{in}(\mathcal{B}_{k-1})\big)$.
  end for
  $\widehat{\mathcal{S}_{\mathcal{B}}} \leftarrow \widehat{\mathcal{S}_{\mathcal{B}}} \cup \mathcal{B}_{n_e+n_d}$
end for

This procedure follows directly from our formulation (Section 5.2.2). Each input BDT $\mathcal{B} \in \mathcal{S}_\mathcal{B}$ is processed independently (line 1). Specifically, $\mathcal{B}$ will traverse the network one layer Π_k at a time (line 4). Within each layer Π_k, the projection through the input sub-layer Ψ_k^{in} is computed (line 8) by composing a non-linearity σ with the basis projection (Section 5.2.2). This yields a set of coefficients representing the input BDT. Next, following Section 5.2.2, these coefficients are transformed back into a valid BDT with the output sub-layer (line 9). At the end of this process, a set of reconstructed BDTs $\widehat{\mathcal{S}_\mathcal{B}}$ is available, for a fixed value of θ.

5.3.6 Backward Propagation

Given the set of reconstructed BDTs $\widehat{\mathcal{S}_\mathcal{B}}$ for the current value of θ (Section 5.3.5), the data fitting energy (Equation 5.7) is evaluated. Specifically, for each input BDT $\mathcal{B} \in \mathcal{S}_\mathcal{B}$, the optimal assignment ϕ_* w.r.t. Equation 2.3 is computed between $\mathcal{B}$ and its reconstruction, $\mathcal{B}_{n_e+n_d}$, provided on the output of the network. Next, similarly to Section 5.3.2 for basis projections, the vector representation $\mathcal{B}'$ of $\mathcal{B}$ is constructed and re-ordered such that the i^{th} entry of this vector corresponds to the pre-image by ϕ_* of the i^{th} entry of $\mathcal{B}'_{n_e+n_d}$ (c.f. Equation 5.16). Then, given the optimal assignment ϕ_*, similarly to Section 5.3.2, $W_2^\mathcal{T}(\mathcal{B}, \mathcal{B}_{n_e+n_d})$ can be expressed as an L_2 norm (Equation 5.10). Given the set Φ_* of all the optimal assignments between the input BDTs and their output reconstructions, $E_{W_2^\mathcal{T}}(\theta)$ is then evaluated:

$$E_{W_2^\mathcal{T}}(\theta) = \sum_{i=1}^{N} ||\mathcal{B}(f_i)' - \mathcal{B}_{n_e+n_d}(f_i)'||_2^2. \tag{5.19}$$

At this stage, for a given set Φ_* of optimal assignments, the evaluation of Equation 5.19 only involves basic operations (as described in the previous sections: vector re-orderings, pseudoinverse computations, linear transformations, and compositions). All these operations are supported by the automatic differentiation capabilities of modern neural frameworks (in our case *PyTorch* [PGM+19]), enabling the automatic estimation of $\nabla E_{W_2^\mathcal{T}}(\theta)$. Then, θ is updated by gradient descent [KB15].

Our overall optimization algorithm (Algorithm 2) can be interpreted as a global instance of an *Assignment/Update* strategy. Each backward propagation updates the overall variable θ to improve the data fitting energy (Equation 5.7), while the next forward propagation improves the network outputs and hence their assignments to the inputs. In the remainder, the

terms PD-WAE and MT-WAE refer to the usage of our framework with persistence diagrams or merge trees respectively.

5.3.7 Computational Parameters

The Wasserstein distance $W_2^{\mathcal{T}}$ is subject to three parameters (ϵ_1, ϵ_2 and ϵ_3, Section B.1), for which we use the recommended default values ($\epsilon_1 = 0.05$, $\epsilon_2 = 0.95$, $\epsilon_3 = 0.9$, Section B.1) when considering merge trees (MT-WAE). In contrast, when considering persistence diagrams, we switch ϵ_1 to 1 (ϵ_2 and ϵ_3 do not have any effect then) and $W_2^{\mathcal{T}}$ becomes equivalent to $W_2^{\mathcal{D}}$ (see Chapter 3 or Section B.1). Then our framework computes a Wasserstein Auto-Encoder of extremum persistence diagrams (PD-WAE for short).

Our main algorithm is subject to meta-parameters. n_{it} stands for the number of iterations in our basis projection procedure (Section 5.3.2). In practice, we set $n_{it} = 2$.

The number, size and dimensionality of the layers of our MT-WAE are also meta-parameters. Unless specified otherwise, we use only one encoding layer and one decoding layer, i.e. $n_e = n_d = 1$, with $d_{n_e} = 2$ (for dimensionality reduction purposes) and $d_{n_e+n_d} = 16$. For data reduction purposes and computational cost control, we also restrict the size of the origins and bases of the sub-layers of our network. Let $|\mathcal{S}_{\mathcal{B}}|$ be the total number of branches in the ensemble, i.e. $|\mathcal{S}_{\mathcal{B}}| = \sum_{i=1}^{N} |\mathcal{B}(f_i)|$. We restrict the maximum size of the following origins as follows: $|\mathcal{O}_1^{in}| \leq 0.2|\mathcal{S}_{\mathcal{B}}|$, $|\mathcal{O}_1^{out}| \leq 0.1|\mathcal{S}_{\mathcal{B}}|$, $|\mathcal{O}_2^{in}| \leq 0.1|\mathcal{S}_{\mathcal{B}}|$, $|\mathcal{O}_2^{out}| \leq 0.2|\mathcal{S}_{\mathcal{B}}|$. This origin size control also implicitly restricts the size of the corresponding bases. Overall, when integrating all these constraints, the number of variables in our networks is bounded by $\big((d_{n_e}+1) \times 2 \times (0.2+0.1) + (d_{n_d}+1) \times 2 \times (0.1+0.2)\big) \times |\mathcal{S}_{\mathcal{B}}| = 12|\mathcal{S}_{\mathcal{B}}|$. In practice, our networks optimized $68,902$ variables on average (per ensemble).

5.4 Applications

This section illustrates the utility of our framework in concrete visualization tasks: data reduction and dimensionality reduction. These applications and use-cases are adapted from Chapter 4, to facilitate comparisons between previous work on the linear encoding of topological descriptors Chapter 4 and our novel non-linear framework.

5.4.1 Data Reduction

As discussed in Chapter 4, like any data representation, merge trees can benefit from lossy compression. For example, for the in-situ analysis of high-performance simulations [ABG+15], each individual time-step of the simulation can be represented and stored to disk in the form of a topological descriptor [BNP+21]. In this context, this lossy compression eases the manipulation of the generated ensemble of topological descriptors (i.e. it facilitates its storage and transfer). Previous work has investigated the reduction of an ensemble of merge trees via linear encoding (Chapter 4). In this section, we improve this application by extending it to non-linear encoding, thereby enabling more accurate data reductions. Specifically, the input ensemble $\mathcal{S}_\mathcal{B}$ of BDTs is compressed, by only storing to disk:
(1) the output sub-layer of the last decoding layer of the network, noted $\Psi^{out}_{n_e+n_d}$ (i.e. its origin, $\mathcal{O}^{out}_{n_e+n_d}$, as well as its basis, $B^{out}_{n_e+n_d}(\mathcal{O}^{out}_{n_e+n_d})$)
(2) the corresponding N BDT coefficients $\alpha^i_{n_e+n_d} \in \mathbb{R}^{d_{n_e+n_d}}$.

Note that an alternative reduction strategy would consist in storing the N BDT coefficients in latent space directly (i.e. $\alpha^i_{n_e} \in \mathbb{R}^{d_{n_e}}$), which would be typically more compact than the N BDT coefficients in the last output sub-layer ($\alpha^i_{n_e+n_d} \in \mathbb{R}^{d_{n_e+n_d}}$). However, in order to decompress this representation, one would need to store to disk the entire set of n_d decoding layers. This significant overhead would only be compensated for ensembles counting an extremely large number N of members. In our experiments (Section 5.5), $N = 48$ for the largest ensemble. Thus, we

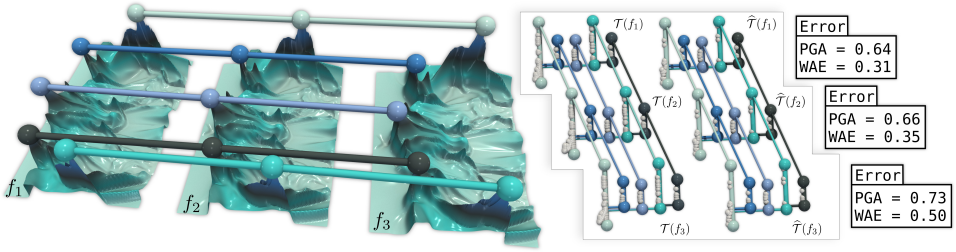


Figure 5.7 – *Application of our data reduction approach to feature tracking (experiment adapted from Chapter 3 and Chapter 4 for comparison purposes). The* 5 *most persistent maxima (spheres) of three time steps (ion density during universe formation [Org04]) are tracked through time (left, f_1, f_2 and f_3) by considering the optimal assignment (Equation 2.3) between the corresponding merge trees (inset, left). The same tracking procedure is applied to the merge trees compressed by WAE (inset, right). Similarly to PGA Chapter 4, the resulting tracking is identical with the compressed trees. However, in comparison to PGA Chapter 4, for a target compression factor of* 13.44*, the relative reconstruction error (right) is clearly improved with WAE.*

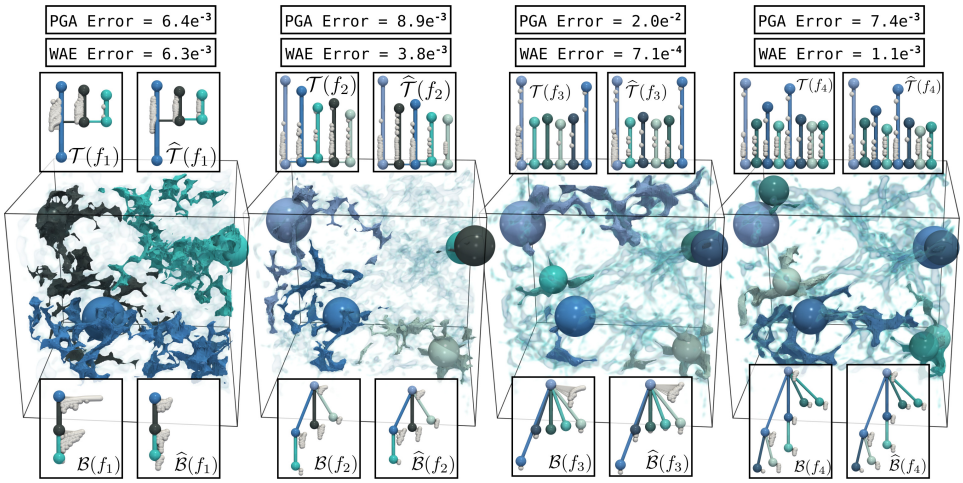


Figure 5.8 – *Application of our data reduction approach to topological clustering (experiment adapted from Chapter 3 and Chapter 4 for comparison purposes). The ensemble is clustered (Chapter 3) based on the merge trees (top left insets) of the ensemble members (each column represents a member from one of the* 4 *clusters). The same clustering procedure (Chapter 3) is applied to the merge trees compressed by WAE (top right insets). Similarly to PGA (Chapter 4), the resulting clustering is identical with the compressed trees (it exactly matches the ground-truth classification, see Appendix A). However, in comparison to PGA (Chapter 4), for a target compression factor of* 15.09*, the relative reconstruction error (top) is clearly improved with WAE. Visually, the compressed trees look very similar to the original ones: the prominent features (in colors, non-prominent features are shown with small white nodes) are well preserved in terms of number and persistence. The same holds for the compressed BDTs (bottom right insets) which are nearly isomorphic to the original BDTs (bottom left insets, the color shows the assignment between the original and compressed BDTs).*

focus on the first strategy described above (storing potentially larger sets of coefficients, but a smaller number of decoding layers).

The compression factor can be controlled with two input parameters: *(i)* $d_{n_e+n_d}$ controls the dimensionality of the last decoding layer (hence its ability to capture small variabilities) and *(ii)* $|\mathcal{O}^{out}_{n_e+n_d}|$ controls the size of the origin of the last decoding sub-layer (hence its ability to capture small features). The resulting reconstruction error (Equation 5.7) will be minimized for large values of both parameters, while the compression factor will be minimized for low values. In the following experiments, we set both parameters to their default values (Section B.1). To decompress a BDT $\mathcal{B}(f_i)$, its stored coefficients $\alpha^i_{n_e+n_d}$ are simply propagated through the stored output sub-layer of the network, $\Psi^{out}_{n_e+n_d}$.

Figure 5.7 and Figure 5.8 show two examples of visualization tasks (feature tracking and ensemble clustering, use cases replicated from Chapter 3 and Chapter 4 for comparison purposes). In these experiments, the

BDTs have been compressed with the strategy described above. Next, the de-compressed BDTs have been used as an input to these two analysis pipelines. In both cases, the output obtained with the de-compressed BDTs is identical to the output obtained with the *original* BDTs. This shows the viability of the de-compressed BDTs and it demonstrates the utility of this reduction scheme.

Table 5.1 and Table 5.2 report a comparison between the reconstruction error generated by our Wasserstein Auto-Encoder (WAE) approach and the Principal Geodesic Analysis (PGA) approach of Chapter 4, for the application to data reduction, in the case of persistence diagrams (Table 5.1) and merge trees (Table 5.2).

Specifically, we compute the reconstruction error of each input BDT $\mathcal{B}(f_i)$ via the distance $W_2^{\mathcal{T}}$ to its reconstruction (computed by the method under consideration, PGA or WAE). To be comparable across ensembles, this distance is then divided by the maximum $W_2^{\mathcal{T}}$ distance observed among two input BDTs in the ensemble. Finally, this *relative* reconstruction error is *averaged* over all the BDTs of the ensemble.

To enable a fair comparison, we set the number of axis of PGA, noted d_{max}, to 3 (as reported in the original data reduction description in Section 4.4.1) and we set the number of dimensions in the latent space of WAE to the same value (i.e. $d_{n_e} = 3$). We also set the maximum size of the PGA origin, noted N_1, to $0.1|\mathcal{S}_{\mathcal{B}}|$, where $|\mathcal{S}_{\mathcal{B}}|$ is the total number of

Table 5.1 – *Comparison of the Average Relative Reconstruction (ARR) Error (rounded to two decimal after the point), between PD-PGA (Chapter 4) ($d_{max} = 3$ and $N_1 \leq 0.1|\mathcal{S}_{\mathcal{B}}|$) and our approach PD-WAE ($d_{n_e} = 3$ and $|\mathcal{O}_{n_e}^{out}| \leq 0.1|\mathcal{S}_{\mathcal{B}}|$), for identical compression factors. Bold numbers in the* Ratio *column indicate instances where PD-WAE achieved a lower (hence better) reconstruction error.*

Dataset	N	$\lvert\mathcal{B}\rvert$	Compression Factor	ARR Error		Ratio
				PD-PGA (Chapter 4)	PD-WAE	
Asteroid Impact (3D)	7	1,295	7.36	0.01	0.01	**0.73**
Cloud processes (2D)	12	1,209	7.99	0.12	0.10	**0.81**
Viscous fingering (3D)	15	118	7.87	0.02	0.01	**0.67**
Dark matter (3D)	40	316	8.68	0.01	4e-03	**0.39**
Volcanic eruptions (2D)	12	811	7.56	0.02	0.01	**0.36**
Ionization front (2D)	16	135	8.01	0.03	0.02	**0.69**
Ionization front (3D)	16	763	7.68	0.05	0.03	**0.65**
Earthquake (3D)	12	1,203	7.59	0.04	0.02	**0.51**
Isabel (3D)	12	1,338	7.58	0.08	0.08	**0.91**
Starting Vortex (2D)	12	124	7.39	0.01	3e-03	**0.51**
Sea Surface Height (2D)	48	1,787	24.66	0.16	0.16	1.02
Vortex Street (2D)	45	23	15.83	2e-03	9e-04	**0.40**

Table 5.2 – *Comparison of the Average Relative Reconstruction (ARR) Error (rounded to two decimal after the point), between MT-PGA (Chapter 4) ($d_{max} = 3$ and $N_1 \leq 0.1|\mathcal{S}_\mathcal{B}|$) and our approach MT-WAE ($d_{n_e} = 3$ and $|\mathcal{O}^{out}_{n_e}| \leq 0.1|\mathcal{S}_\mathcal{B}|$), for identical compression factors. Bold numbers in the* Ratio *column indicate instances where MT-WAE achieved a lower (hence better) reconstruction error.*

Dataset	N	$\lvert\mathcal{B}\rvert$	Compression Factor	ARR Error		Ratio
				MT-PGA (Chapter 4)	MT-WAE	
Asteroid Impact (3D)	7	1,295	13.68	0.13	0.12	**0.93**
Cloud processes (2D)	12	1,209	13.84	2e-04	7e-07	**3e-03**
Viscous fingering (3D)	15	118	13.21	8e-04	7e-07	**8e-04**
Dark matter (3D)	40	316	15.09	2e-04	2e-05	**0.08**
Volcanic eruptions (2D)	12	811	13.83	0.01	2e-03	**0.41**
Ionization front (2D)	16	135	13.44	0.19	0.14	**0.78**
Ionization front (3D)	16	763	13.89	0.24	0.22	**0.92**
Earthquake (3D)	12	1,203	14.07	0.14	0.10	**0.75**
Isabel (3D)	12	1,338	14.03	3e-03	2e-03	**0.72**
Starting Vortex (2D)	12	124	11.92	2e-04	2e-06	**0.01**
Sea Surface Height (2D)	48	1,787	14.36	0.23	0.22	**0.92**
Vortex Street (2D)	45	23	20.27	3e-04	9e-05	**0.26**

branches in the ensemble, i.e. $|\mathcal{S}_\mathcal{B}| = \sum_{i=1}^{N} |\mathcal{B}(f_i)|$. Similarly, for WAE, we set the maximum size of the latent output origin $|\mathcal{O}^{out}_{n_e}|$ to $0.1|\mathcal{S}_\mathcal{B}|$.

For both methods (PGA and WAE), the compression factor is fixed to a common value on a per ensemble basis. The compression factor of WAE is controlled by adjusting, for the last decoding layer, its dimensionality noted $d_{n_e+n_d}$, and the maximum size of its output origin, noted $|\mathcal{O}^{out}_{n_e+n_d}|$.

Both tables show that WAE clearly outperforms PGA (Chapter 4)in terms of average relative reconstruction error, with an average improvement of 37% for persistence diagrams, and 52% for merge trees.

Finally, note that for each ensemble, the merge tree based clustering (Chapter 3) computed from the input BDTs is strictly identical to the clustering computed from the reconstructed BDTs. This confirms the viability of our reconstructed BDTs, and their usability for typical visualization and analysis tasks.

5.4.2 Dimensionality Reduction

This section describes how to use MT-WAE to generate 2D layouts of the ensemble, for the global visual inspection of the ensemble. This is achieved by setting $d_{n_e} = 2$ and by embedding each BDT $\mathcal{B}(f_i)$ as a point in the plane, at its latent coordinates $(\alpha^i_{n_e})_1$ and $(\alpha^i_{n_e})_2$. This results in a summarization view of the ensemble, grouping similar BDTs together (Figure 5.1c). The flexibility of our framework allows to further improve

the quality of this 2D layout. Specifically, we introduce two penalty terms aiming at *(1)* improving the preservation of the Wasserstein metric $W_2^{\mathcal{T}}$ and *(2)* improving the preservation of the clusters of BDTs.

***(1)* Metric preservation:** In order to improve the preservation of the Wasserstein metric $W_2^{\mathcal{T}}$ in the latent space, and hence in the 2D layout, we introduce the following penalty term $P_M(\theta)$:

$$P_M(\theta) = \sum_{\forall i \in \{1,\dots,N\}} \sum_{\forall j \neq i \in \{1,\dots,N\}} \left(W_2^{\mathcal{T}}\left(\mathcal{B}(f_i), \mathcal{B}(f_j)\right) - ||\alpha_{n_e}^i - \alpha_{n_e}^j||_2 \right)^2.$$

Concretely, given two BDTs $\mathcal{B}(f_i)$ and $\mathcal{B}(f_j)$, $P_M(\theta)$ penalizes the variations between their Wasserstein distances and the Euclidean distances between their coordinates $\alpha_{n_e}^i$ and $\alpha_{n_e}^j$ in the latent space.

The integration of the penalty term $P_M(\theta)$ in our optimization algorithm (Section 5.3) is straightforward. The Wasserstein distance matrix, which stores at its entry (i, j) the distance $W_2^{\mathcal{T}}\left(\mathcal{B}(f_i), \mathcal{B}(f_j)\right)$, is computed in a pre-processing stage. Since this matrix is a constant during the optimization, the expression of $P_M(\theta)$ only involves basic operations supported by automatic differentiation. Then, given a blending weight $\lambda_M \in [0, 1]$ (in practice, we set $\lambda_M = 1$), the penalty term $\lambda_M P_M(\theta)$ is simply added to the expression of the reconstruction energy $E_{W_2^{\mathcal{T}}}(\theta)$ (Equation 5.19). Next, the corresponding gradient is evaluated by automatic differentiation and the overall energy is optimized by gradient descent [KB15] , as originally described in Section 5.3.6.

***(2)* Clusters preservation:** We introduce an additional penalty term to improve the preservation of the *natural* clusters of BDTs in the latent space, and hence in the 2D layout. Let $C \in \mathbb{R}^{kN}$ be the vector modeling the input k clusters: the entry $(ik) + j$ of this vector is equal to 1 if the BDT $\mathcal{B}(f_i)$ belongs to the cluster j, 0 otherwise. This *input clustering vector* can be provided either by a pre-defined ground-truth, by interactive user inputs or by any automatic clustering algorithm. In our experiments, to construct this vector C, we used the extension of the k-means clustering to the Wasserstein metric space of merge trees (Chapter 3). Next, in the latent space, we consider the classic k-means algorithm [Elk03, CKV13], where each BDT $\mathcal{B}(f_i)$ is clustered according to its latent coordinates $\alpha_{n_e}^i \in \mathbb{R}^2$. This yields a set of k centroids in the 2D latent space $c_l \in \mathbb{R}^2$ with $l \in \{0, 1, \dots, k-1\}$. To evaluate the similarity between this clustering and the input clustering vector C, we use the celebrated *SoftMax* function [GBC16].

Specifically, we consider the *latent clustering vector* $C' \in \mathbb{R}^{kN}$, such that the entry $(C')_{(ik)+j}$ denotes the probability that the 2D point $\alpha^i_{n_e}$ belongs to the cluster j (β is set to 5):

$$(C')_{(ik)+j} = \frac{e^{-\beta||\alpha^i_{n_e} - c_j||_2}}{\sum_{l=0}^{k-1} e^{-\beta||\alpha^i_{n_e} - c_l||_2}}.$$

Then, the clustering penalty term $P_C(\theta)$ is given by the Kullback-Leibler divergence (a standard indicator for probability similarity):

$$P_C(\theta) = KL(C, C') = \sum_{i=0}^{kN-1} C(i) log\Big(\frac{C(i)}{C'(i)}\Big).$$

Similarly to the metric penalty term, given a blending weight $\lambda_C \in [0,1]$ (in practice, we set $\lambda_C = 1$), the penalty term $\lambda_C P_C(\theta)$ is added to the expression of the reconstruction energy $E_{W_2^T}(\theta)$ (Equation 5.19). The corresponding gradient is estimated by automatic differentiation and the overall energy is optimized by gradient descent [KB15], as originally described in Section 5.3.6.

We augment our 2D layouts with *Persistence Correlation Views* (PCV) which were introduced in Chapter 4. In short, the PCV embeds a branch b of the barycenter $\mathcal{B}_*$ (Chapter 3) as a point in 2D, in order to represent the variability of the corresponding feature in the ensemble, as a function of the coordinates in latent space. Specifically, the optimal assignments ϕ_{*_i} between $\mathcal{B}_*$ and each input BDT $\mathcal{B}(f_i)$ is first computed (Equation 2.3). Next, for a given branch $b \in \mathcal{B}_*$, the Pearson correlation $\rho\big(p_{b_i}, (\alpha^i_{n_e})_1\big)$ between the persistence p_{b_i} of $\phi_{*_i}(b) \in \mathcal{B}(f_i)$ and the first coordinate in latent space $(\alpha^i_{n_e})_1$ is computed for the ensemble (i.e. for $i \in \{1, 2, \ldots, N\}$). Next, the Pearson correlation $\rho\big(p_{b_i}, (\alpha^i_{n_e})_2\big)$ is computed similarly with regard to the second coordinate in latent space $(\alpha^i_{n_e})_2$. Finally, b is embedded in the PCV at the coordinates $\Big(\rho\big(p_{b_i}, (\alpha^i_{n_e})_1\big), \rho\big(p_{b_i}, (\alpha^i_{n_e})_2\big)\Big)$. To avoid clutter in the visualization, we only report the most persistent branches of $\mathcal{B}_*$ in the PCV. Intuitively, points in the PCV which are located far away from the center, along a given direction, indicate a strong correlation between that direction in latent space, and the persistence of the corresponding feature in the ensemble.

PCVs enable the identification of patterns of feature variability within the ensemble, as discussed in Figure 5.9. This case study considers the *Isabel* ensemble, which consists of 12 scalar fields representing the wind velocity magnitude in a hurricane simulation. The ensemble comes with a ground-truth classification (Appendix A): 4 members correspond to the

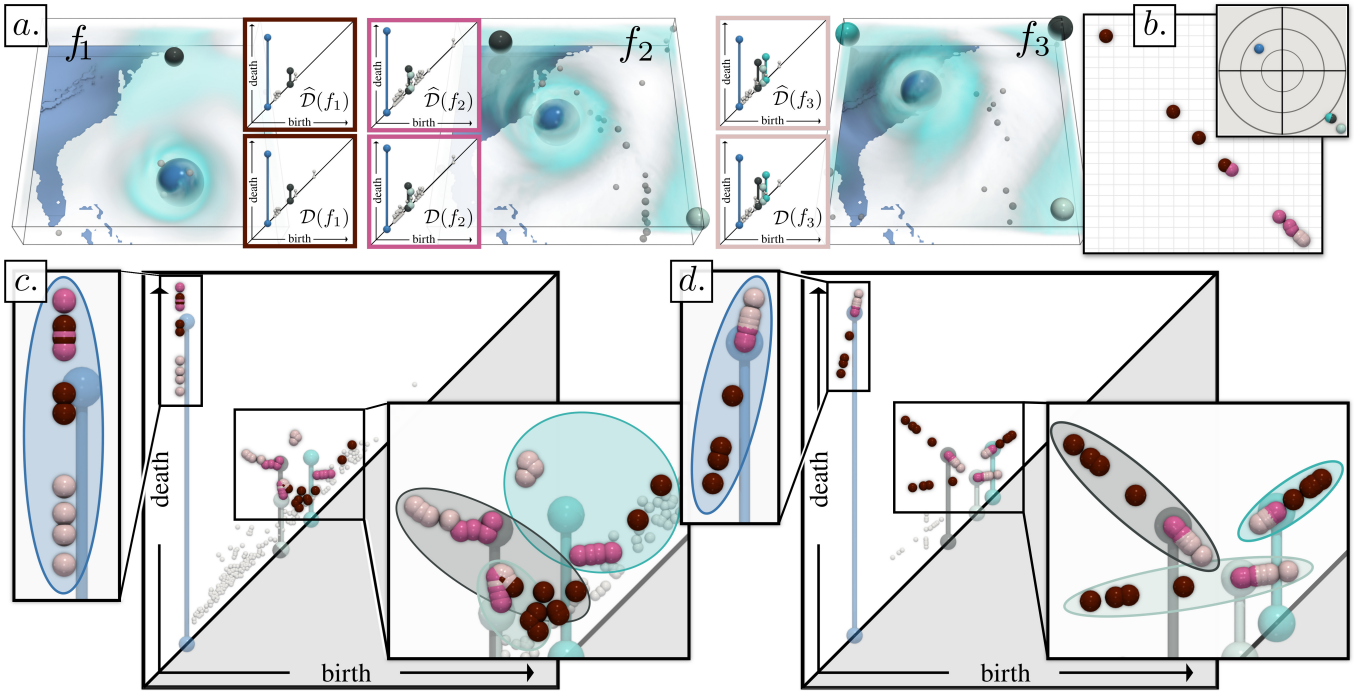


Figure 5.9 – *Visual analysis of the* Isabel *ensemble (one member per ground-truth class, top), with PD-WAE (in this example, $n_e = n_d = 2$). Our work enables data reduction, while providing reconstructed diagrams $\big((a)$, top insets$\big)$ which are highly similar to the input (below). The 2D layout generated by PD-WAE (b) recovers the temporal structure of the ensemble (the clusters are aligned in chronological order, from dark red to light pink). The PCV (grey inset) indicates that the peripheral gusts of wind in the data (white, black and cyan spheres) grow in importance when moving towards the bottom right corner of the latent space (their persistence increases over time). In contrast, the hurricane eye (light blue) exhibits less variability, but with stronger values towards the top left corner of the latent space (start of the sequence). The aggregated views (bottom, overlapping all diagrams, transparent: barycenter) for the input (c) and the latent space (d) show that PD-WAE nicely recovers a per-feature parameterization in the latent space (one ellipse per barycenter feature), which is locally consistent with the data temporal evolution (cluster color).*

formation of the hurricane (e.g. f_1, Figure 5.9), 4 other members to its drift (e.g. f_2, Figure 5.9) and 4 other members to its landfall (e.g. f_3, Figure 5.9). For this ensemble, our PD-WAE approach produces a 2D layout $\big($Figure 5.9b$\big)$ which manages to recover the temporal coherency of the ensemble: the formation (dark red), drift (pink) and landfall (light pink) clusters are arranged in order along a line $\big($direction $(1, -1)\big)$. This shows the ability of PD-WAE to recover the intrinsic structure of the ensemble (here its temporal nature). The PCV (grey inset) further helps appreciate the variability of the features in the ensemble. There, each colored point indicates a persistent feature of the barycenter: the eye of the hurricane is represented by the blue sphere, while the cyan, black and white sphere represent peripheral gusts of wind $\big($see the matching features in the data, Figure 5.9a$\big)$. The PCV clearly identifies two patterns of feature variability, along the direction $(1, -1)$, which coincides with the temporal alignment

of the clusters in latent space. Specifically, it indicates that the persistence of the hurricane eye will be larger in the top left corner of the latent space, i.e. towards the beginning of the temporal sequence (dark red cluster). This is confirmed visually when inspecting the persistence diagrams of the individual members (the blue feature is less persistent in $\mathcal{D}(f_3)$ than in $\mathcal{D}(f_1)$ and $\mathcal{D}(f_2)$). In short, this visually encodes the fact that the strength of the hurricane eye decreases with time. In contrast, the features corresponding to peripheral wind gusts (cyan, black and white spheres) exhibit a common variability pattern, distinct form that of the hurricane eye: the persistence of the corresponding features increases as one moves along the direction $(1, -1)$ in latent space, i.e. as time increases (pink and light pink clusters). This is confirmed visually in the individual members, where the persistence of these features is larger in $\mathcal{D}(f_3)$ than in $\mathcal{D}(f_1)$ and $\mathcal{D}(f_2)$. In short, this visually encodes the fact that the strength of the peripheral wind gusts increases with time. Overall, while the 2D layout generated by PD-WAE enables the visualization of the intrinsic structure of the ensemble (here, its temporal nature), the PCV enables the visualization and interpretation of the variability in the ensemble at a feature level. The caption of Figure 5.1 includes a similar discussion for MT-WAE.

Figure 5.10 provides a qualitative comparison of the 2D layouts and Persistence Correlation Views (PCVs) between PD-PGA (Chapter 4) and our novel non-linear framework, PD-WAE (see Section 5.5.2 for an extensive quantitative comparison). Specifically, it shows that, while PD-PGA (Figure 5.10a)) manages to isolate the clusters well, its 2D layout does not recover the intrinsic, one-dimensional, temporal structure of the ensemble. In contrast, as discussed above, PD-WAE (Figure 5.10b)) manages to recover this intrinsic structure and produces a linear alignment of the clusters along the direction $(1, -1)$, in order of their temporal appearance.

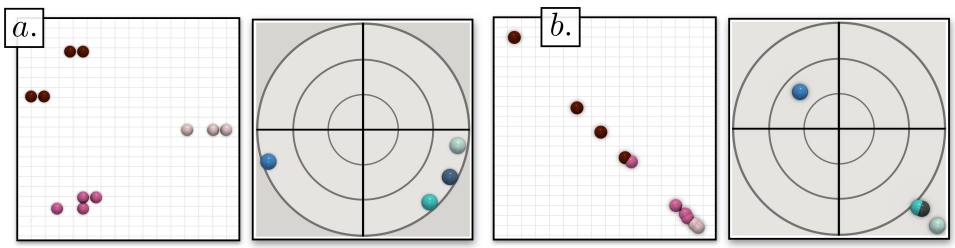


Figure 5.10 – *Qualitative comparison of the 2D layouts (left, white inset) and PCVs (right, grey inset) between PD-PGA (Chapter 4)* (a) *and PD-WAE* (b) *for the* Isabel *ensemble. PD-WAE manages to recover the intrinsic temporal structure of the ensemble and produces a linear alignment of the clusters in order of temporal appearance (dark red, pink, light pink).*

This alignment greatly facilitates the interpretation of the PCV, since time is now visually encoded there by the linear direction $(1, -1)$) (whereas it would be encoded by a curve in the case of PD-PGA).

In contrast to traditional auto-encoders, our approach maintains topological descriptors throughout the network. This results in an improved interpretability and enables new visual capabilities:

***(1)* Latent feature transformation:** As discussed in Figure 5.9, it is now possible to visualize how topological descriptors are (non-linearly) transformed by the auto-encoder. Specifically, the aggregated views of the birth/death space (bottom) illustrates how PD-WAE *unwraps* the diagrams in latent space, nicely recovering the data temporal evolution at a feature level (see the temporally consistent linear arrangements of points for each barycenter feature in Figure 5.9d), from dark red to pink and light pink).

***(2)* Latent space navigation:** Given a point in latent space, it is now possible to efficiently reconstruct its BDT/MT by propagating its latent coordinates through the decoding layers, enabling an interactive exploration of the merge tree latent space (Figure 5.1d).

***(3)* Feature traversal analysis:** For each consecutive layers Π_k and Π_{k+1}, we compute the optimal assignment (Equation 2.3) between their input origins, $\mathcal{O}_k^{in}$ and $\mathcal{O}_{k+1}^{in}$. Next, we compute the optimal assignment between the barycenter $\mathcal{B}_*$ (Chapter 3) of the input ensemble and the first origin $\mathcal{O}_1^{in}$. This yields an explicit tracking of each branch b of $\mathcal{B}_*$ down to the latent space. We introduce the notion of *Feature Latent Importance* (FLI), given by the persistence of b in latent space, divided by its original persistence. FLI indicates if a feature gains (or looses) importance in latent space. This enables the identification of the most *informative* features in the ensemble.

This is illustrated in Figure 5.1e), where the cyan, white, dark blue and black features exhibit large FLI values (red circles). In the trees, these features are indeed present in most of the ensemble $\big($Figure 5.1d$)\big)$, with only moderate variations in persistence. Interestingly, the global maximum of seismic wave (light blue feature) is not a very informative feature (light blue circle): while it is also present throughout the sequence, its persistence decreases significantly (left to right). This visually encodes that, as the seismic wave travels from the epicenter, the strength of its global

maximum is no longer significant in front of other local maxima (which illustrates the energy diffusion process).

5.5 Results

This section presents experimental results obtained on a computer with two Xeon CPUs (3.2 GHz, 2x10 cores, 96GB of RAM). The input merge trees were computed with FTM [GFJT19a] and pre-processed to discard noisy features (persistence simplification threshold: 0.25% of the data range). We implemented our approach in C++ (with OpenMP and PyTorch's C++ API [PGM+19]), as modules for TTK [TFL+17, BMBF+19]. Experiments were performed on a set of 12 public ensembles described in Appendix A, which includes a variety of simulated and acquired 2D and 3D ensembles extracted from previous work and past SciVis contests [Org04].

5.5.1 Time Performance

The Wasserstein distance computation (Equation 2.3) is the most expensive sub-procedure of our approach. It intervenes during energy evaluation (Section 5.3.6) but also at each iteration of basis projection (Section 5.3.2), itself occuring at each propagation iteration (Section 5.3.5), for each input BDT. To compute this distance, we use a fine-grain task-based parallel algorithm (Chapter 3). We leverage further parallelism to accelerate the process. Specifically, for each input BDT, its forward propagation (Section 5.3.5) can be run in a distinct parallel task. Similarly, when evaluating

Table 5.3 – *Running times (in seconds) of our algorithm for PD-WAE and MT-WAE computation (first sequential, then with 20 cores).*

Dataset	N	$\lvert\mathcal{B}\rvert$	PD-WAE			MT-WAE		
			1 c.	20 c.	Speedup	1 c.	20 c.	Speedup
Asteroid Impact (3D)	7	1,295	2,819.38	989.42	2.85	5,946.81	1,522.89	3.90
Cloud processes (2D)	12	1,209	11,043.20	1,318.63	8.37	16,150.90	2,566.98	6.29
Viscous fingering (3D)	15	118	1,345.31	268.78	5.01	3,727.64	417.31	8.93
Dark matter (3D)	40	316	125,724.00	10,141.30	12.40	135,962.00	8,051.02	16.89
Volcanic eruptions (2D)	12	811	4,925.15	638.58	7.71	4,151.04	449.14	9.24
Ionization front (2D)	16	135	627.04	95.31	6.58	1,140.44	144.57	7.89
Ionization front (3D)	16	763	17,285.10	1,757.71	9.83	101,788.00	5,350.46	19.02
Earthquake (3D)	12	1,203	14,272.10	2,074.52	6.88	9,888.68	1,024.45	9.65
Isabel (3D)	12	1,338	3,485.03	436.56	7.98	23,240.90	1,669.20	13.92
Starting Vortex (2D)	12	124	215.46	95.45	2.26	281.80	147.51	1.91
Sea Surface Height (2D)	48	1,787	41,854.70	2,901.39	14.43	222,594.00	13,540.20	16.44
Vortex Street (2D)	45	23	460.46	152.35	3.02	117.88	38.88	3.03

the overall energy (Section 5.3.6), the distance between an input BDT and its output reconstruction is computed in a distinct parallel task for each BDT. Table 5.3 evaluates the time performance of our framework for persistence diagrams (PD-WAE) and merge trees (MT-WAE). In sequential mode, the computation time is a function of the ensemble size (N) and the tree sizes ($|\mathcal{B}|$). In parallel, the iterative nature of our approach (Algorithm 2) challenges parallel efficiency. However, timings are still improved after parallelization (orders of minutes on average), with a very good parallel efficiency for the largest ensembles.

5.5.2 Framework Quality

Figure 5.7 and Figure 5.8 report compression factors for our application to data reduction (Section 5.4.1). These are ratios between the storage size of the input N BDTs and that of their compressed form. For a fixed target compression factor, WAE clearly improves the reconstruction error over linear encoding (PGA, Chapter 4). Table 5.1 and Table 5.2 extend this error comparison to all our test ensembles, and shows that, for identical compression factors, our framework improves the reconstruction error over PGA (Chapter 4) by 37% for persistence diagrams, and 52% for merge trees, hence confirming the accuracy superiority of WAE over PGA.

Figure 5.11 provides a visual comparison for the planar layouts generated by a selection of typical dimensionality reduction techniques, applied on the input merge tree ensemble (i.e. each point is a merge tree). This experiment is adapted from Chapter 4 for comparison purposes. This figure reports quantitative scores. For a given technique, to quantify its ability to preserve the *structure* of the ensemble, we run k-means in the 2D layouts and evaluate the quality of the resulting clustering (given the ground-truth, Appendix A) with the normalized mutual information (NMI) and adjusted rand index (ARI). To quantify its ability to preserve the *geometry* of the ensemble, we report the metric similarity indicator SIM Chapter 4, which evaluates the preservation of the Wasserstein metric $W_2^{\mathcal{T}}$. All these scores vary between 0 and 1, with 1 being optimal.

MDS [KW78] and t-SNE [vdMH08] have been applied on the distance matrix of the input merge trees (Wasserstein distance, Equation 2.3, default parameters, Section B.1). By design, MDS preserves well the metric $W_2^{\mathcal{T}}$ (good SIM), at the expense of mixing ground-truth classes together (low NMI/ARI). t-SNE behaves symmetrically (higher NMI/ARI, lower SIM). We applied PGA (Chapter 4) by setting the origin size parameter to

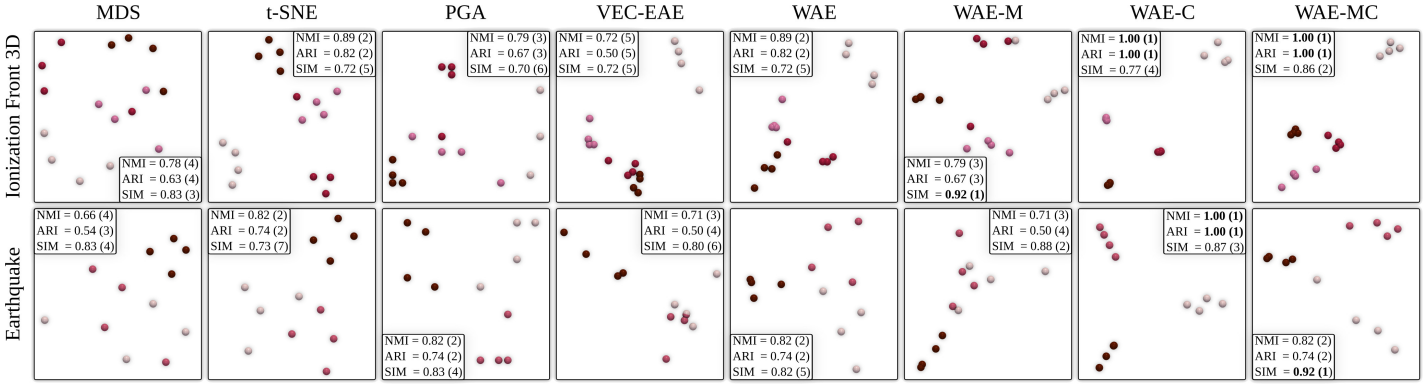


Figure 5.11 – *Comparison of planar layouts for typical dimensionality reduction techniques, on two merge tree ensembles. The color encodes the classification ground-truth (Appendix A). For each quality score, the best value appears bold and the rank of the score is shown in parenthesis. This experimental protocol is adapted from Chapter 4 for comparison purposes.*

a value compatible to our latent space ($0.1|\mathcal{S}_{\mathcal{B}}|$, Section 5.3.7). As expected, PGA provides a trade-off between the extreme behaviors of MDS and t-SNE, with an improved cluster preservation over MDS (NMI/ARI), and an improved metric preservation over t-SNE (SIM). WAE also constitutes a trade-off between MDS and t-SNE, but with improved quality scores over PGA.

We compare our approach to a standard auto-encoder (EAE, Section 5.2.1) applied on a vectorization of the input merge trees. Each input BDT $\mathcal{B}(f_i)$ is embedded in $\mathbb{R}^{2|\mathcal{B}_*|}$, such that the j^{th} entry of this vector corresponds to the birth/death location of the branch of $\mathcal{B}(f_i)$ which maps to the j^{th} branch of the barycenter $\mathcal{B}_*$ (Chapter 3). Next, we feed these vectorizations to an EAE, with the same meta-parameters as our approach (i.e. number of layers, dimensionality per layer). The corresponding results appear in the VEC-EAE column. Our approach (WAE) outperforms this straightforward application of EAE, with clearly higher clustering scores (NMI/ARI) and improved metric scores (SIM).

Figure 5.11 also reports the layouts obtained with our approach after enabling the metric penalty term (WAE-M), the clustering penalty term (WAE-C) and both (WAE-MC), c.f. Section 5.4.2. WAE-M (respectively WAE-C) significantly improves the metric (respectively cluster) preservation over MDS (repectively t-SNE). The combination of the two terms (WAE-MC) improves both quality scores *simultaneously*: it outperforms MDS (SIM) *and* it improves t-SNE (NMI/ARI). In other words, WAE-MC improves established methods by outperforming them on their dedicated criterion (SIM for MDS, NMI/ARI for t-SNE).

Figure 5.13 extends Figure 5.11 to all our test ensembles. It confirms visually the conclusions of the table of aggregated scores (Table 5.4).

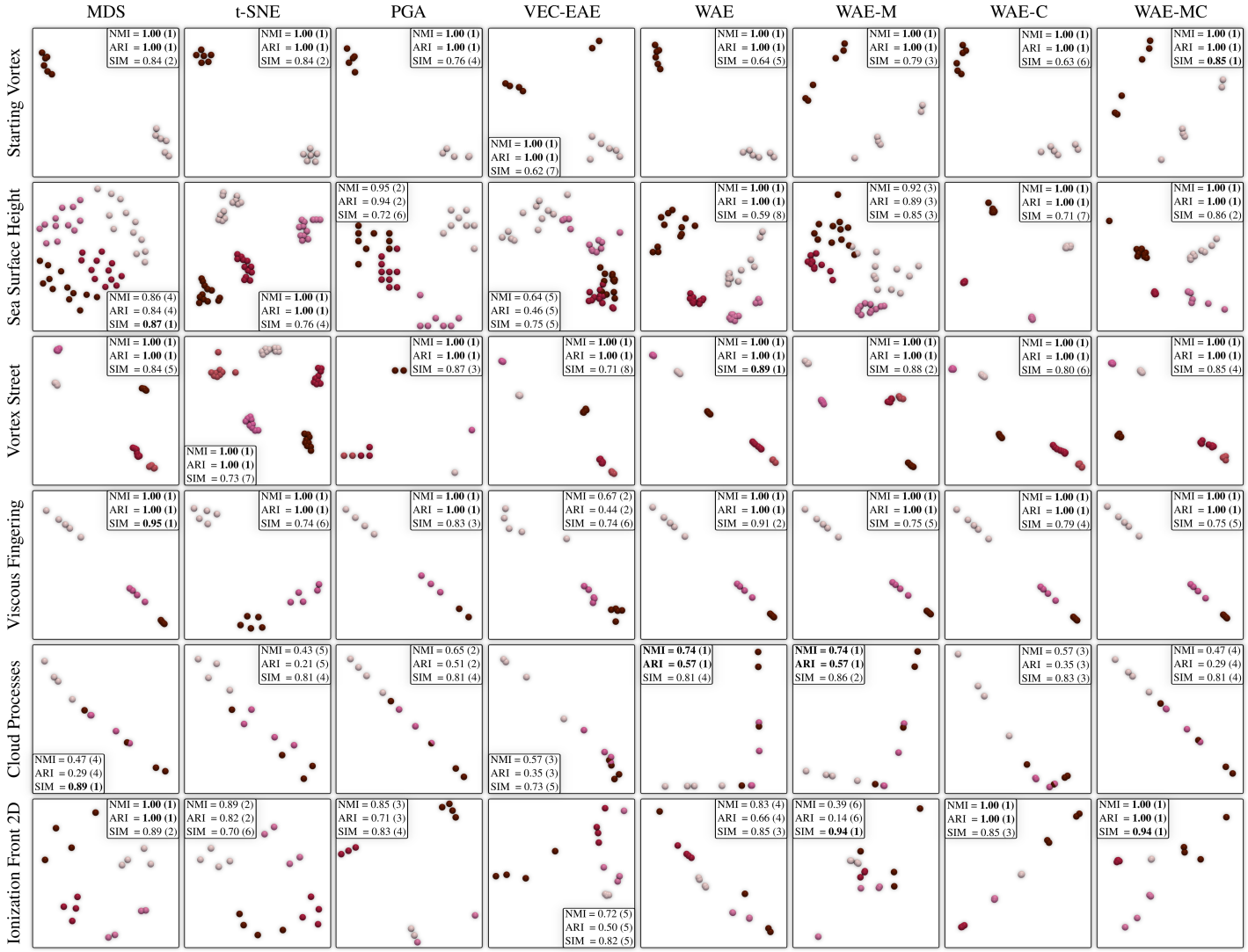


Figure 5.12 – *Comparison of planar layouts for typical dimensionality reduction techniques on the first half of our merge tree ensembles. The color encodes the classification ground-truth (Appendix A). For each quality score, the best value appears bold and the rank of the score among all methods is in parenthesis.*

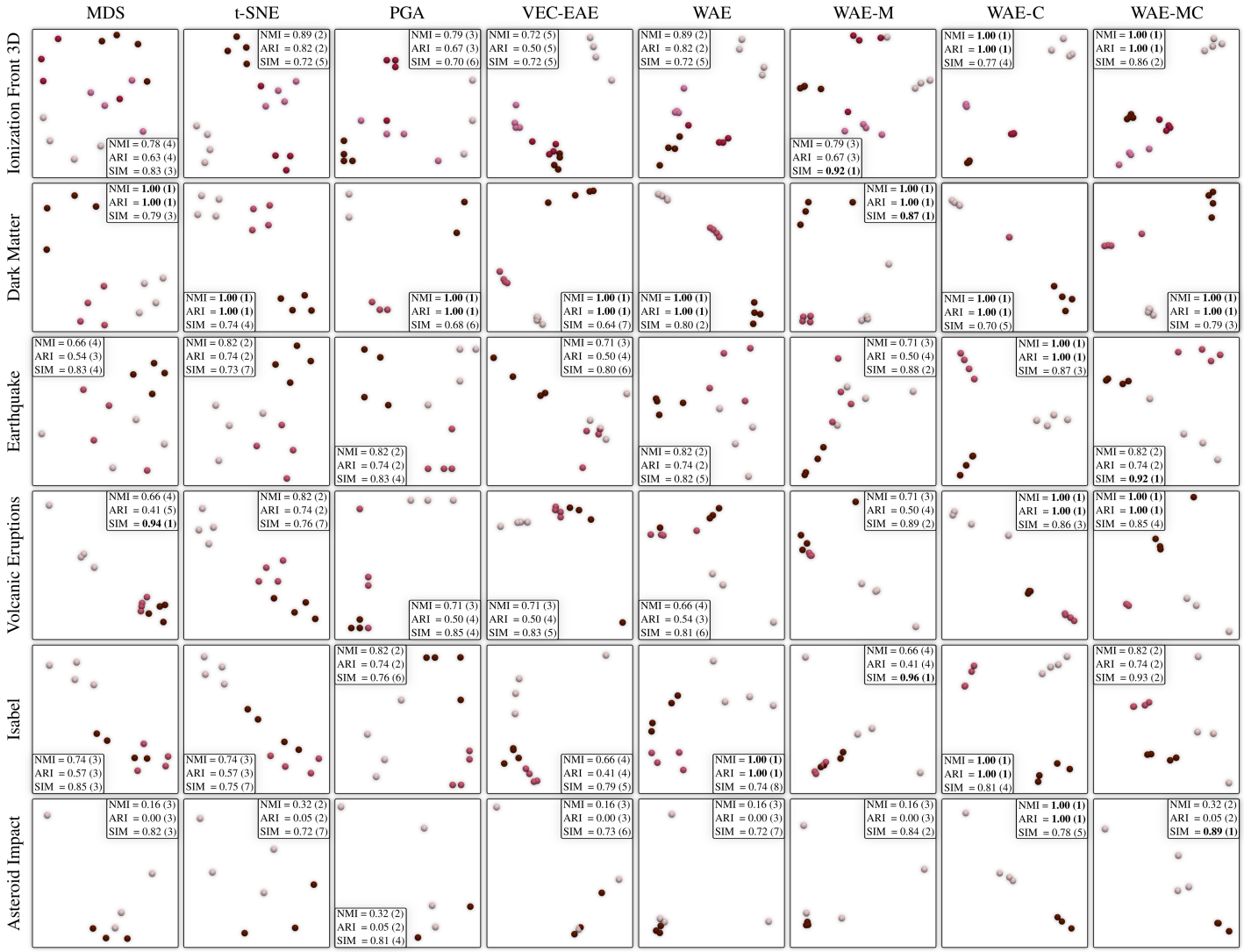


Figure 5.13 – *Comparison of planar layouts for typical dimensionality reduction techniques on the second half of our merge tree ensembles. The color encodes the classification ground-truth (Appendix A). For each quality score, the best value appears bold and the rank of the score among all methods is in parenthesis.*

Table 5.4 – *Comparison of* aggregated *layout quality scores (i.e. averaged over* all *merge tree ensembles, bold: best values). WAE-MC provides both superior metric (SIM) and cluster (NMI/ARI) preservation to pre-existing techniques (MDS [KW78], t-SNE [vdMHo8], PGA (Chapter 4), VEC-EAE).*

Indicator	MDS	t-SNE	PGA	VEC-EAE	**WAE**	**WAE-M**	**WAE-C**	**WAE-MC**
NMI	0.78	0.83	0.82	0.71	0.84	0.76	**0.96**	0.87
ARI	0.68	0.75	0.74	0.55	0.77	0.63	**0.95**	0.82
SIM	0.86	0.75	0.79	0.74	0.78	**0.87**	0.78	0.86

In particular, it confirms that WAE behaves as a *trade-off* between the respective advantages of standard techniques, such as MDS [KW78] and t-SNE [vdMHo8]. Specifically, MDS is known to preserve the input metric well, while t-SNE tends to better preserve the global structure of the data (i.e. the ground-truth classification), at the expense of metric violation. Our approach (WAE) provides a trade-off between these two extreme behaviors: *(i)* it improves over MDS in terms of structure preservation (it provides equivalent or better NMI/ARI scores for 11 out of 12 ensembles) and *(ii)* it improves over t-SNE in terms of metric preservation (it provides an equivalent or better SIM score for 9 out of 12 ensembles). WAE also outperforms VEC-AE and improves PGA on most ensembles. Finally, the combination of our two penalty terms, WAE-MC, simultaneously outperforms MDS on metric preservation and t-SNE on cluster preservation (hence maximizing all criteria at once), for 8 of the 12 ensembles.

Table 5.4 also extends our quantitative analysis to all our test ensembles. It confirms the clear superiority of WAE over VEC-EAE. It also confirms that the combination of our penalty terms (WAE-MC) provides the best metric (SIM) and cluster (NMI/ARI) scores over existing techniques.

Figure 5.14 reports the evolution of the normalized reconstruction error for PD and MT-WAE computations, for all our test ensembles. As often

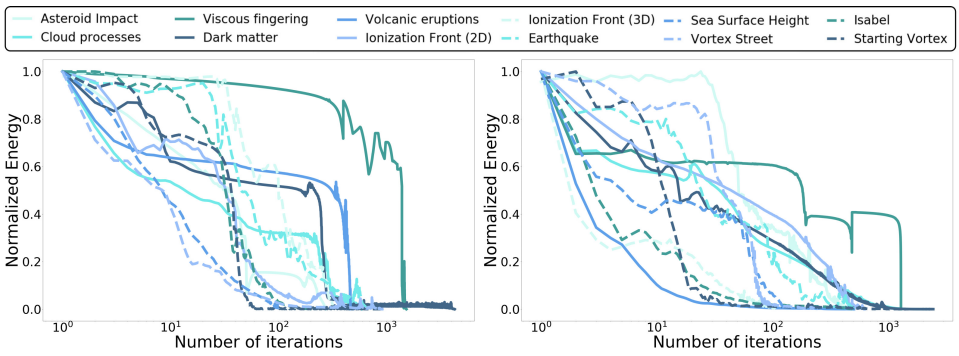


Figure 5.14 – *Evolution of the normalized reconstruction error along the iterations, for PD-WAE (left) and MT-WAE (right).*

observed when optimizing neural networks, typical energy oscillations are present in both cases. However, Figure 5.14 shows that these oscillations do not prevent the networks to converge (i.e. to reach a state where the energy decreases by less than 1% between consecutive iterations).

5.5.3 Empirical Stability Evaluation

As documented in Chapter 3 introducing the Wasserstein distance between merge trees ($W_2^{\mathcal{T}}$), saddle swap instabilities in the merge trees are commonly addressed with a *saddle-merging* pre-processing ([SMKN20] or Section B.1). This procedure consists in moving each branch b up the BDT $\mathcal{B}(f)$, if its saddle is too *close* to that of its parent branch (i.e. closer in normalized f values than a threshold ϵ_1, see Section B.1). As documented in Chapter 3 with practical stability evaluations (Figure 3.13), this simple saddle-merging pre-processing drastically improves in practice the robustness of the metric $W_2^{\mathcal{T}}$ to additive noise. Thus, this saddle-merging pre-processing is of paramount importance for the practical usage of $W_2^{\mathcal{T}}$ on real-life datasets and we recommend to use $\epsilon_1 = 0.05$ as a default value (Section B.1). Note that this parameter ϵ_1 acts as a control knob, which balances the practical stability of the metric with its discriminative power (for $\epsilon_1 = 1$, $W_2^{\mathcal{T}} = W_2^{\mathcal{D}}$).

In this section, we study the practical stability of our non-linear framework for merge tree encoding (WAE) to additive noise, in order to document the impact of the underlying metric's stability on the outcome of the analysis.

Setup

For this experiment, we specifically generated a synthetic ensemble, in order to control both its intrinsic parameterization and its classification. For this, we proceeded as follows.

First, four 2D scalar fields (Figure 5.15, top left inset) were generated by sampling a 2D basis of Gaussian mixtures with controlled parameterization. Specifically, the scalar field being the origin of the basis has two hills ($f_{(0,0)}$, dark red frame, top left inset). The extremity of the first (horizontal) axis ($f_{(1,0)}$, red frame, top left inset) has exactly the same hills, but with a first *additional* maximum (cyan sphere). The extremity of the second (vertical) axis ($f_{(0,1)}$, pink frame, top left inset) has a second additional maximum (white sphere). Finally, the fourth dataset ($f_{(1,1)}$, light pink frame, top left inset) has both extra maxima (cyan and white spheres).

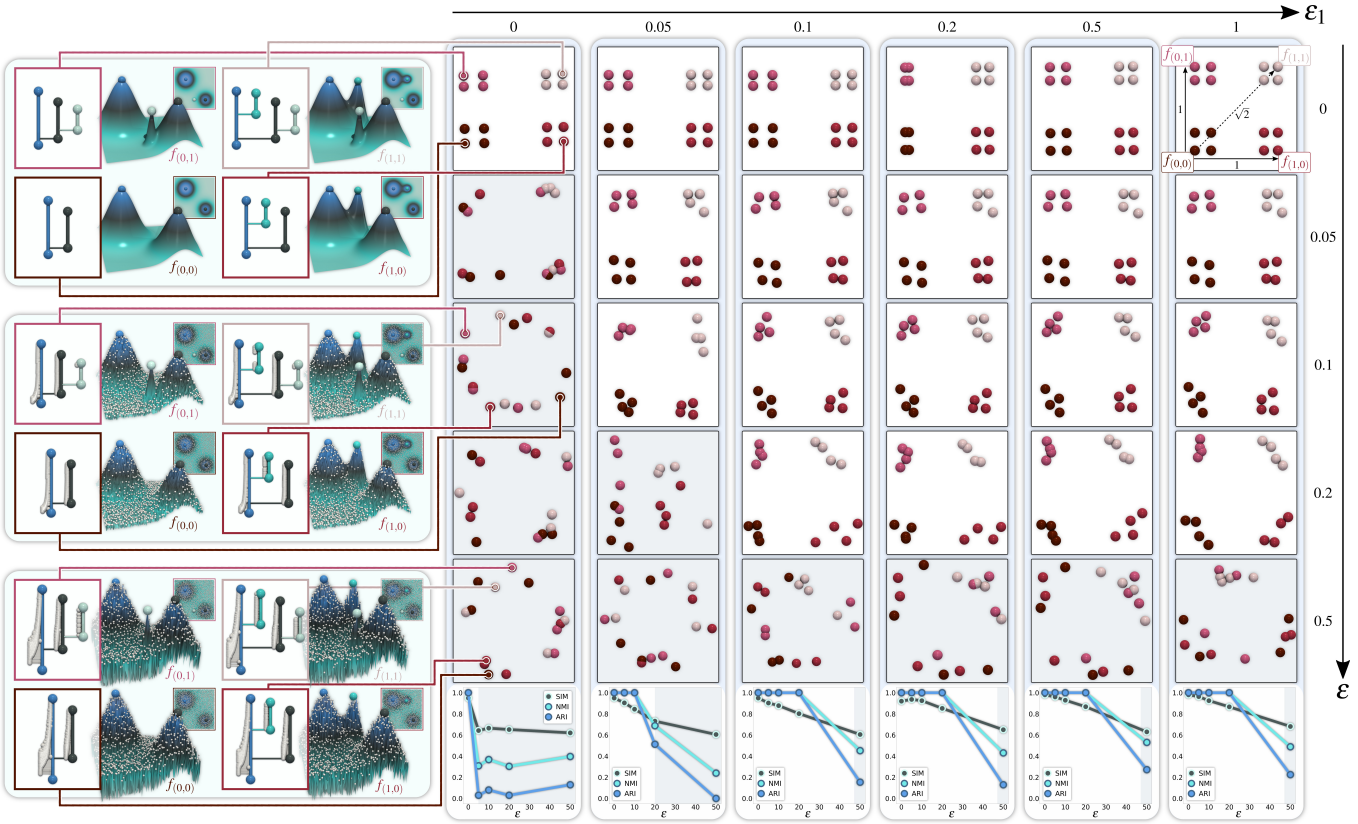


Figure 5.15 – ***Empirical stability evaluation:*** *a synthetic ensemble of sixteen 2D scalar fields is specifically designed by sampling a 2D basis of Gaussian mixtures, with a controlled parameterization (see Section 5.5.3 for a detailed specification). This yields a* ground-truth *parameterization and classification of the ensemble (four clusters: dark red, red, pink, light pink). Five versions of this ensemble are created, for increasing levels of additive noise (from $\epsilon = 0$ to $\epsilon = 0.5$, top to bottom). For each ensemble, a 2D layout is generated by our non-linear framework WAE (right insets), for increasing values of the parameter ϵ_1 from left to right (i.e. from the strict Wasserstein distance between merge trees, $W_2^{\mathcal{T}}$ for $\epsilon_1 = 0$, to progressive blends towards the Wasserstein distance between persistence diagrams, $W_2^{\mathcal{D}}$ for $\epsilon_1 = 1$). In the 2D layout and the quality scores (bottom curves), a grey background indicates an unstable computation (i.e. NMI and ARI are both below 1). For the default recommended value of the parameter ϵ_1 (0.05, Section B.1), WAE with $W_2^{\mathcal{T}}$ recovers well the ground-truth parameterization and classification (similarly to $W_2^{\mathcal{D}}$), up to a level of additive noise of $\epsilon = 0.1$. For $\epsilon_1 \geq 0.1$, the 2D layouts generated by WAE provide a similar level of robustness for $W_2^{\mathcal{T}}$ and $W_2^{\mathcal{D}}$ (bottom curves).*

These Gaussian mixtures were generated by adjusting the height of the additional maxima (cyan and white spheres, Figure 5.15) such that their diagrams describe a square on the Wasserstein metric space (see the top right 2D layout of Figure 5.15):

$$\begin{aligned} W_2^{\mathcal{D}}\left(\mathcal{D}(f_{(0,0)}, \mathcal{D}(f_{(1,0)})\right) &= W_2^{\mathcal{D}}\left(\mathcal{D}(f_{(1,0)}, \mathcal{D}(f_{(1,1)})\right) \\ &= W_2^{\mathcal{D}}\left(\mathcal{D}(f_{(1,1)}, \mathcal{D}(f_{(0,1)})\right) \\ &= W_2^{\mathcal{D}}\left(\mathcal{D}(f_{(0,1)}, \mathcal{D}(f_{(0,0)})\right) \\ &= 1, \end{aligned}$$

and:

$$\begin{aligned} W_2^{\mathcal{D}}\big(\mathcal{D}(f_{(0,0)}, \mathcal{D}(f_{(1,1)})\big) &= W_2^{\mathcal{D}}\big(\mathcal{D}(f_{(1,0)}, \mathcal{D}(f_{(0,1)})\big) \\ &= \sqrt{2}. \end{aligned}$$

Next, we repeated this square generation process, around each corner of the above square, but this time with a smaller side length (equal to 0.15 in the Wasserstein metric space, instead of 1).

Overall, this results in a total of 16 scalar fields, specifically organized along a *ground-truth* 2-dimensional parameterization of the Wasserstein metric space, with a natural *ground-truth* classification (corresponding to the closest corner of the 2D grid, see the top right 2D layout of Figure 5.15):

- Class 1 (bottom left corner, dark red spheres in Figure 5.15):
  - $f_{(0,0)}$, $f_{(0.15,0)}$, $f_{(0.15,0.15)}$, $f_{(0,0.15)}$;
- Class 2 (bottom right corner, red spheres in Figure 5.15):
  - $f_{(0.85,0)}$, $f_{(1,0)}$, $f_{(1,0.15)}$, $f_{(0.85,0.15)}$;
- Class 3 (top right corner, light pink spheres in Figure 5.15):
  - $f_{(0.85,0.85)}$, $f_{(1,0.85)}$, $f_{(1,1)}$, $f_{(0.85,1)}$;
- Class 4 (top left corner, bright pink spheres in Figure 5.15):
  - $f_{(0,0.85)}$, $f_{(0.15,0.85)}$, $f_{(0.15,1)}$, $f_{(0,1)}$;

Given the above ground-truth parameterization, we call the *ground-truth distance matrix*, noted $\mathbb{D}$, the matrix defined such that each of its entries (i, j) is equal to $W_2^{\mathcal{D}}\big(\mathcal{D}(f_i), \mathcal{D}(f_j)\big)$.

Next, we generated additional versions of the above ensemble, by introducing a random additive noise in the scalar fields, with a control on the maximum normalized amplitude $\epsilon \in [0, 1]$ (i.e. the maximum amplitude of the noise is a fraction ϵ of the global function range of the input scalar field). Specifically, we considered the noise levels $\epsilon \in \{0, 0.05, 0.1, 0.2, 0.5\}$. Overall, this results in 5 ensembles of 16 scalar fields each.

Protocol

Given the above ensembles, we first consider our non-linear framework for persistence diagrams, namely PD-WAE. Specifically, we generated, for

each noise level, a 2D layout of the ensemble with PD-WAE (Section 5.4.2). This is shown in the rightmost column of Figure 5.15 ($\epsilon_1 = 1$). We quantitatively evaluate the quality of this 2D layout along two criteria: metric preservation and cluster preservation.

First, given the 2D layout of the ensemble, we compute a distance matrix D in 2D, which we compare to the *ground-truth* distance matrix $\mathbb{D}$ (see Section 5.5.3) with the *SIM* indicator (Chapter 4) (which varies between 0 and 1, 1 being optimal). Second, given the 2D layout of the ensemble, we compute a k-means clustering in 2D (with $k = 4$) and we compare the resulting classification to the *ground-truth* classification with the *NMI* and *ARI* indicators (which vary between 0 and 1, 1 being optimal).

To study the stability to additive noise of our framework when considering the Wasserstein distance between merge trees, we have replicated the above experiment for 5 more values of the control parameter ϵ_1 (in Figure 5.15, from left to right: 0, 0.05, 0.1, 0.2 and 0.5). Overall this results in the 2D array represented in Figure 5.15 where each column denotes a specific value of the control parameter ϵ_1 and where each line denotes a specific noise level ϵ.

Analysis

In the absence of noise ($\epsilon = 0$, top row) and for arbitrary values of the parameter ϵ_1, our non-linear WAE framework manages to produce a 2D layout of the ensemble which is faithful to the ground-truth parameterization (high *SIM* values, bottom curves in Figure 5.15) and which preserves the ground-truth clusters (colors from dark red to light pink, Figure 5.15, high *NMI/ARI* values).

As soon as noise is introduced ($\epsilon >= 0.05$), the strict distance $W_2^{\mathcal{T}}$ ($\epsilon_1 = 0$, leftmost column) becomes unstable, as originally documented in Chapter 3. As a consequence, both the ground-truth classification and parameterization are not recovered by MT-WAE in the 2D layout: spheres of different colors are mixed together (as assessed by the low *NMI/ARI* values, leftmost curves, Figure 5.15) and the spheres are no longer organized along a 2D grid (as assessed by the lower *SIM* values, leftmost curve, Figure 5.15). In contrast, with the original Wasserstein distance between persistence diagrams ($\epsilon_1 = 1$, rightmost column), up to a significant level of noise ($\epsilon = 0.2$), both the ground-truth classification and parameterization are well preserved in the 2D layout generated by PD-WAE: the spheres with the same color remain clustered (high *NMI/ARI* values, rightmost

curves) and the spheres are properly arranged along a 2D grid (high *SIM* values, rightmost curve).

For the recommended value of the control parameter ϵ_1 (0.05, Section B.1), MT-WAE still manages to recover well the ground-truth parameterization and classification, up to a noise level of $\epsilon = 0.1$ (perfect clustering, with high *SIM* values). For a larger value of ϵ_1 ($\epsilon_1 \geq 0.1$), the 2D layouts generated by MT-WAE are very similar to these generated with PD-WAE (rightmost column), with identical stability indicators (*SIM* and *NMI/ARI* curves, bottom).

In conclusion, this experiment shows that for mild levels of noise ($\epsilon < 0.1$), the recommended value of ϵ_1 (0.05) results in a stable MT-WAE computation. For larger noise levels, MT-WAE provides similar stability scores to PD-WAE for values of ϵ_1 which are still reasonable in terms of discriminative power ($\epsilon_1 = 0.1$).

5.5.4 Limitations

As discussed in Section B.1, the parameter ϵ_1 of the Wasserstein distance between merge trees ($W_2^{\mathcal{T}}$) acts as a control knob, that balances the practical stability of the metric with its discriminative power. Specifically, for $\epsilon_1 = 1$, we have $W_2^{\mathcal{T}} = W_2^{\mathcal{D}}$ and $W_2^{\mathcal{T}}$ is stable, but less discriminative. In Chapter 3 we showed experimentally that for relatively low values of ϵ_1 (0.05), $W_2^{\mathcal{T}}$ still behaved in a stable manner in practice for reasonable noise levels. Our overall MT-WAE framework behaves similarly. Section 5.5.3 provides a detailed empirical stability evaluation of our framework in the presence of additive noise. In particular, this experiment shows that for reasonable levels of additive noise ϵ (normalized with regard to the function range), typically $\epsilon < 0.1$, the recommended default value of ϵ_1 (0.05) results in a stable MT-WAE computation. For larger noise levels ($\epsilon > 0.1$), MT-WAE provides similar stability scores to PD-WAE, for values of ϵ_1 which are still reasonable in terms of discriminative power ($\epsilon_1 = 0.1$).

A possible direction to improve the practical stability of the framework without having to deal with a control parameter such as ϵ_1 would be to consider branch decompositions driven by other criteria than persistence (such as hyper-volume [CSvdP04] for instance). However, the persistence criterion plays a central role in the Wasserstein distance between merge trees, as discussed in Section 3.4, in particular to guarantee that interpolated BDTs computed during geodesic construction can indeed be inverted into a valid MT. Thus, other branch decomposition criteria than persis-

tence would require to derive a completely new procedure for several key components of our framework, such as geodesic computation or barycenter estimation. This is an orthogonal research direction to this work, which we leave for future work.

Similarly to other optimization problems based on topological descriptors ([TMMH14, VBT20], Chapter 3 or Chapter 4), our energy is not convex. However, our experiments indicate that our initialization strategy (Section 5.3.4) leads to relevant solutions, which can be successfully applied for visualization (Section 5.4).

Since it is based on neural networks, our approach inherits from their intrinsic limitations. Specifically, the energy is not guaranteed to monotonically decrease over the iterations. However, this theoretical limitation has never translated into a practical limitation in our experiments. Figure 5.14 provides a detailed analysis of the energy evolution along the iterations of our algorithm, for all our test ensembles. In particular, it shows that temporary energy increases can indeed be observed, but without preventing the network from converging overall.

Like other neural methods, our approach is conditioned by the meta-parameters defining the network (i.e. number of layers, dimensionality of each layer, etc.). However, we ran our experiments with fairly basic values for these meta-parameters (Section 5.3.7) and still obtained substantial improvements over linear encoding based on PGA (Chapter 4). This indicates that optimizing in the future these meta-parameters is likely to improve the quality of our framework, however possibly at the expense of longer computations.

5.6 Summary

In this chapter, we presented a computational framework for the Wasserstein Auto-Encoding of merge trees (and persistence diagrams), with applications to data reduction and dimensionality reduction. Our approach improves previous linear attempts at merge tree encoding, by generalizing them to non-linear encoding, hence leading to lower reconstruction errors. In contrast to traditional auto-encoders, our novel layer model enables our neural networks to process topological descriptors natively, without pre-vectorization. As shown in our experiments, this contribution leads not only to superior accuracy (Section 5.5.2) but also to superior interpretability (Section 5.4.2): with our work, it is now possible to interactively explore the latent space and analyze how topological features are

transformed by the network in its attempt to best encode the ensemble. Overall, the visualizations derived from our contribution (Figure 5.1 and Figure 5.9) enable the interactive, visual inspection of the ensemble, both at a global level (with our 2D layouts) and at a feature level. Specifically, our novel notion of *feature latent importance* enables the identification of the most informative features in the ensemble.

Conclusion

6

In this thesis, we focused on the development of analysis methods for ensembles of topological descriptors. The resulting tools allow the analysis of an ensemble of scalar fields based on their topological representations, effectively addressing the primary challenges arising from modern data, their ever-increasing size and complexity, by representing them in a concise manner. This shift of representation permits to operate, with a coarser granularity, directly on structures of the data rather than on their tiniest details (such as pixels of an image), allowing a better and easier interpretability of the results. Those given by our methods give insights about an ensemble of datasets, by visually presenting them to end users.

6.1 Summary of Contributions

The contributions of this thesis encompass different facets of ensemble analysis, from the summarization of a whole ensemble with the computation of *representative* descriptors, to the identification of distinct patterns of variability within it. All methods presented in this thesis are implemented in the open-source library Topology ToolKit [TFL+17], making them easily accessible and reusable through the various examples we also provide. Moreover, we have provided curated and preprocessed versions of ensembles of datasets from the SciVis contest available at that time (described in Appendix A), making them readily usable in software such as ParaView or with VTK.

Distances, Geodesics and Barycenters of Merge Trees

In order to tackle the lack of discriminability of persistence diagrams we first proposed in Chapter 3 to extend the available tools for their anal-

ysis to merge trees, being representations that capture more effectively the global structure of the data by encoding the features in a hierarchical manner instead of representing them individually. We have proposed a new distance taking into account the additional structural information provided by merge trees and being purposely designed for the efficient computation of geodesics and barycenters. Moreover, it generalizes the Wasserstein distance between persistence diagrams due to a parameter allowing to weight how much the hierarchical structure of the trees should be taken into account in the computation. When the tree structures are not taken into account, merge trees can be considered equivalent to persistence diagrams, and the proposed distance thus becomes identical to the Wasserstein distance between persistence diagrams. This parameter can be used as a trade-off, in order to balance the stability of the diagrams with the discriminability of the merge trees. When computing geodesics or barycenters, in order to ensure that the core properties defining merge trees are respected, we introduce a normalization term which enforces the nested birth/death values of the tree hierarchy. We show the utility of these contributions with applications such as feature tracking, temporal reduction and the clustering of an ensemble of scalar fields based on their merge trees.

Linear Variability Analysis of Topological Descriptors

With the aim of going beyond the simple notion of average that is the barycenter, we have explored methods in Chapter 4 for analyzing the variability of an ensemble of persistence diagrams or merge trees. To achieve this, we have adapted the well-known Principal Component Analysis (PCA) framework to make it able to process natively these specific objects. This adaptation is made through the redefinition of the low-level geometric tools used in the Euclidean space to the Wasserstein space of merge trees or persistence diagrams. Our method computes a basis capturing the most informative geodesics (i.e. analog of straight lines in the space of merge trees) corresponding to directions in the ensemble having the most variability. Moreover, we adapt to our framework the correlation view, a core tool in the PCA framework, indicating which features are the most responsible for the variability. In our context, it results in a visualization showing which structures of the ensemble vary the most. Once again, due to the coarser granularity of topological abstractions, it enhances the interpretability of the variability of an ensemble by focusing

on entire structures rather than their tiniest details. This framework allows the extension of two typical applications of PCA. First, data reduction, where the input merge trees or diagrams are compressed, by keeping only the computed basis and the coordinates of the input abstractions on this basis, allowing their reconstruction given the model. Then, dimensionality reduction, where these coordinates are used in a two-dimensional layout showing how the members of the ensemble are arranged with each other.

Non-Linear Variability Analysis of Topological Descriptors

The linear encoding given by the adaptation of PCA to merge trees or persistence diagrams has some limitations regarding the versatility of the patterns it can model. Specifically, the directions of variability extracted by linear encoding will describe constant growth or decay of structures in the ensemble and can only find relations among structures that evolve together in a simple linear manner. This fact motivated the exploration of methods that could capture more complex, non-linear, patterns. To tackle this, we have adapted in Chapter 5 a generalization of PCA, the Auto-Encoder, allowing such extraction of non-linear patterns, to merge trees and persistence diagrams. This adaptation was made through the development of a novel neural network layer capable of natively processing these topological abstractions without pre-vectorization, operating non-linear transformations on these objects, inspired by those being done on vectors in Euclidean space with usual neural network layers. The versatility of our optimization procedure was shown through the introduction of two penalty terms, to help preserve in the latent space the distances of the original space as well as their clusters. We extended, using the new proposed method, the applications of the previous contribution, data reduction and dimensionality reduction. Compared to the linear encoding, this method allows an improvement of 23% in average regarding clustering preservation metrics and 10% for the distance preservation metric when choosing our best model for each of these categories of metrics.

6.2 Discussion

Some limitations regarding the different contributions of this thesis were already discussed in the corresponding chapters. However, we would like here to add or emphasize some restrictions of our work and possible solutions that could be explored to solve them.

The main limitation regarding the use of merge trees in our work is probably their lack of theoretical stability. For instance, as mentionned in Chapter 3, saddle-swaps could change the tree hierarchy and therefore forbid some matchings between structures that would have made sense from an application perspective. Moreover, the proposed distance between merge trees can be unstable when high level of noise is present in the data. To tackle these issues, we have shown experimentally that these instabilities can be mitigated using the saddle-merging parameter, allowing to choose a trade-off between stability and discriminability depending on the necessities of the application. We have shown empirically (Figure 3.13 and Figure 5.15) that even for small values of this parameter, when discriminability is preferred over stability, that the proposed metric between merge trees still remains stable under reasonnable noise levels.

Other strategies could be explored to tackle these instabilities. For instance, the size of the domain, in terms of number of simplices, impacts directly the number of features that are only due to noise, hence their impact on the metric. The usage of usual methods of noise removal such as averaging values could be studied, in addition to other mechanisms, in a pre-processing step for topological methods. However it could significantly alter sharp and thin structures if the averaging is too wide and introduces approximations on the birth and death values of structures. The impact of noise-related features on the metric is especially relevant when the Wasserstein parameter p is set to low values, like 2 in our work, implying that the metric is sensitive to even the smallest differences, possibly those coming from noise in the data. Increasing p will give more weight to the larger discrepancies hence possibly not taking into account the noise if the latter is not too important. A high value of p leads to a metric with stronger stability properties but being less informative, when using the extreme value $p = \infty$, only the maximum difference is taking into account, providing very little information about the comparison of two scalar fields. Moreover, the definition and computation of geodesics and barycenters are simplified by the fact that $p = 2$, altering this parameter would necessitate a redefinition of these concepts.

Using another value for the p parameter will however not tackle the structural instabilities of the merge tree hierarchy like saddle-swap.One line of research that was explored recently was to use other branch decompositions than the persistence-based one [BYM+14, WLG22]. These methods involve the search for a *good* branch decomposition, usually minimizing the distance, hence tackling the structural instabilities (that usu-

ally implies a higher distance) but at the cost of a search space becoming extremely large. Moreover, taking other branch decompositions than the persistence-based one removes all relations with the Wasserstein distance between persistence diagrams that assigns persistence pairs with each other. Other kind of metrics were also proposed based on the tree structures [WG22, MBW14]. However, all the above-mentioned methods have a higher complexity making them hardly praticable for real-life datasets with merge trees having thousands of nodes like our distance can tackle. A trade-off between these metrics and ours could be interesting to explore, to tackle structural instabilities while being usable on real-life datasets.

Another main limitation of our work is due to the computation time of ouf methods whose bottleneck is usually the distance computation between topological abstractions, especially in our last contribution involving lot of distance computations. We tackled this problem by parallelizing the distances that could be parallelized, by using the Auction algorithm instead of the classical Munkres method being much slower and by removing the low persistence pairs that are usually not important in visualization applications. Still, there is room for improvement, one possibility would be to use the Sinkhorn algorithm to solve optimal transport problems [Cut13] being known for its efficiency. However, this method usually does not lead to exact one-to-one assignments between the structures of the topological abstractions being a key property needed for interpretability in visualization tasks. Another possiblity that could be interesting would be to develop approximation methods for the distance computation of topological abstractions. [VBT20] explored this possibility for persistence diagrams using progressive mechanisms resulting in significantly faster computation. However, when dealing with merge trees, we usually have many small assignment problems instead of a single large one indicating that such mechanisms may not be as effective as it is for persistence diagrams. We could also look at variants of the Wasserstein distance such as the Sliced Wasserstein distance [RPDB12] being much faster and bounded by above by the Wasserstein distance. This method does not, however, directly gives an assignment between the structures of the diagrams. The computation of a barycenter under this metric was studied in [BRPP15]. However, this metric has unfortunately not yet been studied for merge trees.

6.3 Perspectives

Generalization to other Topological Abstractions

A natural direction for future work is the extension of the proposed tools to more advanced topological abstractions such as contour trees, Reeb graphs or Morse-Smale complexes. Specifically, the same analogy regarding the lack of discriminability of persistence diagrams compared to merge trees can be made by comparing merge trees to Reeb graphs for instance. These other topological abstractions capture additional information, being not captured by persistence diagrams and merge trees, that could be used to improve results in terms of discriminability and interpretability. However, as shown in this thesis, the extension of tools from persistence diagrams to merge trees is not trivial and we could expect the adaptation to these more advanced topological abstractions to be as complex if not even more. The definition of informative and computable metrics between these abstractions is still an open research problem. Regarding contour trees, a possibility would be to extend the edit distance between unrooted trees [DG18] by incorporating edit costs related to scalar values of contour trees, as the edit distance between rooted trees [Zha96] was extended to merge trees [SMKN20]. However, this edit distance has a cubic complexity that could challenge its use on real-life datasets yielding contour trees with thousands of nodes. One could add constraints to this edit distance to reduce its complexity as the unconstrained edit distance between rooted trees [ZSS92] was constrained in [Zha96]. These constraints were actually extremely helpful in terms of complexity but also regarding how the structural information of the trees can be taken into account in the computation (to forbid some matchings for instance).

However, the definition of a metric does not ensure easily computable geodesics and barycenters. As shown in our first contribution, some mechanisms were needed to be introduced in order to ensure the preservation of the structural properties defining merge trees during the computation of geodesics or barycenters.

Adaptation of other Methods to Topological Abstractions

In the continuity of our work, other methods from machine learning or optimal transport could be adapted to topological abstractions.

For instance, variations of the k-means algorithm such as fuzzy k-means, k-medoids, online k-means or Kohonen (self-organizing) map

could be explored. For these last two, we believe that it could be achievable without too much difficulty since online k-means consists in updating centrois using a weighted average, mechanism that can be managed by our framework, and Kohonen map is roughly speaking online k-means where the update of a centroid also updates, in a lesser extent, neighbor centroids according to a grid connecting them. The k-medoids method is known to be more complex and computationally expensive even for vectors in Euclidean space. An approximation is usually made such as Partitioning Around Medoids (PAM) [kme90], since the original problem is NP-hard. The adaptation to persistence diagrams or merge trees seems more complex, especially since the distance computation for these objects is the bottleneck in terms of computation time and this method requires a lot of them. A naive and easy approach would be to adapt PAM to these topological abstractions, a better approach would be to develop a specialized method for these objects tackling this computation time bottleneck. Fuzzy k-means was already explored for persistence diagrams [DAWL23] and could be explored for merge trees.

Another future line of research in that context would be to adapt other kinds of neural network layers to topological abstractions. For instance layers specialized in time series with attention mechanism such as the recent transformer layer. In the light of how we adapted the usual neural network layer to topological abstraction in our work, a geometric interpretation of these layers could help to define them in the Wasserstein space of persistence diagrams or merge trees in order to natively process them.

Regarding optimal transport, future work could explore the use of the Sinkhorn algorithm for optimal transport [Cut13] in our framework. The main challenge would be to manage to have one-to-one matchings from the solution given by this algorithm for interpretability in visualization applications. A possibility would be to derive one-to-one matchings that approximate the solution, for instance by using it as an initialization for one step of the Auction algorithm. The use of this method in our framework could be addressed in, at least, two different ways. A first one, where all the small assignment problems involved in a distance between merge trees are solved by this algorithm, however, in that case it is not clear if the improvement over the Auction algorithm would be significant due to the small size of the assigment problems. A second one, looking much more difficult, where a single assignment problem is solved, including the BDTs isomorphism constraint in the matrix scaling algorithm. Another line of research would be to adapt optimal transport projection methods

such as the Sliced Wasserstein distance [RPDB12] or the Projection Pursuit [MKZ+19] to merge trees. Once again, it could be done locally, on all assigment problems, or globally, by taking into account the BDTs isomorphism constraint. A remaining important problem would be, however, to establish a relation between the original distance and the distance using these methods.

Enhance Topological Abstractions with Geometric Information

Topological asbtractions are extremely helpful tools to summarize in a concise manner scalar fields and by encoding directly the structures in a compact manner. We could however argue, in specific application scenarios, that this reduction looses too much information about the geometrical properties of the structures when encoding them with only two values, their birth and death during the filtration. One line of research that was explored in that direction was to add geometrical information, usually on the persistence pairs, and use that information in the metric computation. For instance, the geometrical lifting [SPCT18] adds the coordinates of the critical points in the domain (usually the extremum in pratice) to the persistence pairs and use this information in addition to the birth and death values when computing distance between persistence pairs. In [SSW14] the volume (or area) of the structure, in terms of number of simplices, is taken into account to convery its geometric size. Inspired by [TN13], the geodesic distance between extrema could also be used to compare structures. These methods however still only add global information on the geometry of the features. Without necessarily developping totally new representations, another axis of development that could be interesting would be to add an additional structure on top of the existing topological abstractions that would describe the features geometry. A motivation would be to make a trade-off between the original data representation and the current topological abstractions. A naive direction would be to take a subset of the simplices constituing a structure to best describe it in a non-exhaustive manner. A better possibility would be to parametrize in a concise manner, with few parameters, the geometrical *shape* of each structure. Once these objects are defined, a natural direction would be the definition of analysis tools such as metrics, geodesics, barycenters and so on using these new objects.

Appendix: Data Specification

This appendix provides a complete specification of the ensemble datasets used in this thesis. In particular, we document the data provenance, its representation, its pre-processing when applicable, and we specify the associated ground-truth classification.

All of these ensemble datasets were extracted from public repositories. We additionally provide a set of scripts which automatically download all of these datasets (at the exception of *Asteroid impact* and *Cloud processes*, for which the dataset providers need to be contacted personally), pre-process them with TTK and output them in VTK file format, with the ground-truth classification attached to the files as meta-data (i.e. *"Field Data"* in the VTK terminology). For convenience, we also provide an archive containing the entire curated ensemble datasets (in VTK file format). All of this new material (scripts and curated data) is located at the following address: https://github.com/MatPont/WassersteinMergeTreesData.

Moreover, we also provide in additional material all the ensembles of merge trees computed from these datasets (in the code archive containing the implementation of our method).

Asteroid Impact

This ensemble is composed of 7 members, given as 3D regular grids (sampled at $300 \times 300 \times 300$, implicitly triangulated by TTK). It has been made available in the context of the SciVis contest 2018 [PG18]. Each member corresponds to the last time step of the simulation of the impact of an asteroid with the sea at the surface of the Earth, for two configurations of asteroid diameter. The considered scalar field is the matter density, which is one of the variables of the simulation which discriminates well the asteroid from the water and the ambient air. This ensemble corresponds to a

parameter study (in this case, studying the effect of the asteroid's diameter on the resulting wave), which is a typical task in numerical simulation. In this application, salient maxima capture well the asteroid and large water splashes. Thus, each member is represented by the split tree (capturing maxima). The associated ground-truth classification assigns members computed with similar asteroid diameters to the same class. Thus, the corresponding classification task consists in identifying, for a given member, its correct asteroid diameter class. The ground-truth classification is as follows:

- **Class 1** (3 members): yA11, yB11, yC11.
- **Class 2** (4 members): yA31, yA32, yB31, yC31

Another selection of the original data has been used for the evaluation of our temporal reduction framework (Figure 3.9). For this experiment, we used the asteroid diameter configuration *"yA31"* and considered the following time steps, organized in 4 phases (according to the SciVis contest companion documentation [PG18]):

- **Phase 1, initial state** (5 time steps): 01141, 03429, 05700, 07920, 09782
- **Phase 2, approach** (5 time steps): 13306, 16317, 18124, 19599, 21255
- **Phase 3, impact** (5 time steps): 28649, 31737, 34654, 37273, 39476
- **Phase 4, aftermath** (5 time steps): 44229, 45793, 47190, 48557, 49978

Cloud Processes

This ensemble is composed of 12 members, given as 2D regular grids (sampled at 1430×1557, implicitly triangulated by TTK). Each member corresponds to a time step of the simulation of cloud formations [WCG+17]. For this application, large clouds are well captured by the maxima of the pressure variable (pre-processed with 10 iterations of smoothing). Thus, split trees (capturing maxima) are considered for this ensemble. The associated ground-truth classification assigns each time step to one of the three key phases of the simulation. The corresponding classification task therefore consists in identifying, for each time step, to which phase it belongs. The ground-truth classification is as follows:

- **Class 1** (4 members): 0, 5, 10, 15
- **Class 2** (4 members): 500, 505, 510, 515

- **Class 3** (4 members): 1000, 1005, 1010, 1015

Viscous Fingering

This ensemble is composed of 15 members, given as 3-dimensional point clouds (representing a particle-based flow simulation). Each point cloud is turned into a Eulerian representation of the variables by using the *"Gaussian Resampling"* filter of ParaView, effectively transforming, via interpolation [She68], each ensemble member into a 3D regular grid (sampled at $50 \times 50 \times 50$, implicitly triangulated by TTK). The original data has been made available in the context of the SciVis contest 2016 [GGH+16]. Each member corresponds to the last time step of the simulation of a viscous fingering phenomenon, occurring when dissolving salt in water. The considered scalar field is the salt concentration, whose salient maxima capture well the most prominent fingers. Thus, each member is represented by the split tree (capturing maxima). Given the studied physical phenomenon, the simulation approach is not deterministic, resulting in distinct outputs for identical initial configurations. In this application, three distinct solver resolutions have been considered, corresponding to three distinct numbers of particles (resolution code 20: 194k particles, resolution code 30: 544k particles, resolution code 44: 1.7M particles). Thus, this ensemble corresponds to a parameter study (in this case, studying the effect of the input resolution on the output fingering), which is a typical task in numerical simulation. The associated ground-truth classification assigns members with the same input resolution to the same class. Thus, the corresponding classification task consists in identifying, for a given ensemble member, its corresponding particle count. The ground-truth classification is as follows:

- **Class 1, resolution 20** (5 members): 20run1, 20run3, 20run4, 20run5, 20run6.
- **Class 2, resolution 30** (5 members): 30run1, 30run2, 30run3, 30run4, 30run5
- **Class 3, resolution 44** (5 members): 30run1, 30run2, 30run3, 30run4, 30run5

Dark Matter

This ensemble is composed of 40 members, given as 3-dimensional point clouds (representing a particle-based simulation). Each point cloud is

turned into a Eulerian representation of the variables by using the *"Gaussian Resampling"* filter of ParaView, effectively transforming, via interpolation [She68], each ensemble member into a 3D regular grid (sampled at $100 \times 100 \times 100$, implicitly triangulated by TTK). The original data has been made available in the context of the SciVis contest 2015 [HGTS15]. Each member corresponds to a time step of a simulation of the universe formation, where regions of high concentration of dark matter form a filament structure known as the *cosmic web*. The considered scalar field is therefore dark matter density, whose salient maxima capture well large clusters of galaxies. Thus, each member is represented by the split tree (capturing maxima). The associated ground-truth classification assigns each time step to one of the four key phases of the simulation. The corresponding classification task therefore consists in identifying, for each time step, to which phase it belongs. The ground-truth classification is as follows:

- **Class 1** (10 members): 0.0200, 0.0300, 0.0400, 0.0500, 0.0600, 0.0700, 0.0800, 0.0900, 0.1000, 0.1100
- **Class 2** (10 members): 0.2700, 0.2800, 0.2900, 0.3000, 0.3100, 0.3200, 0.3300, 0.3400, 0.3500, 0.3600
- **Class 3** (10 members): 0.5900, 0.6000, 0.6100, 0.6200, 0.6300, 0.6400, 0.6500, 0.6600, 0.6700, 0.6800
- **Class 4** (10 members): 0.9100, 0.9200, 0.9300, 0.9400, 0.9500, 0.9600, 0.9700, 0.9800, 0.9900, 1.0000

Volcanic Eruptions

This ensemble is composed of 12 members, given as 2D regular grids (sampled at 500×500, implicitly triangulated by TTK). Each member corresponds to an observation of a volcanic eruption, obtained by satellite imaging (as this data exhibits a bit of noise, it has been pre-simplified by removing all saddle-maxima pairs with a persistence lower than 0.5% of the data range). The original data has been made available in the context of the SciVis contest 2014 [FHG+14]. The considered scalar field is the sulfur dioxide concentration, for which salient maxima correspond to volcanic eruptions. Thus, each observation is represented by the split tree (capturing maxima). Each member corresponds to a specific acquisition period, itself corresponding to the eruption of one particular volcano at

the surface of the Earth. The associated ground-truth classification assigns observations acquired in the same period of time to the same class. The corresponding classification task therefore consists in identifying, for each observation (taken at a specified date), the erupting volcano it corresponds to. The ground-truth classification is as follows:

- **Class 1** (4 members): 150_am, 150_pm, 151_am, 151_pm
- **Class 2** (4 members): 156_am, 156_pm, 157_am, 157_pm
- **Class 3** (4 members): 164_am, 164_pm, 165_am, 165_pm

Ionization Front (3D)

This ensemble is composed of 16 members, given as 3D regular grids (sampled at $300 \times 124 \times 124$, implicitly triangulated by TTK). Each member corresponds to a time step of a simulation of ionization front propagation [TCWN08]. For this application, large ionization flares are well captured by salient maxima of the ion concentration. Thus, split trees (capturing maxima) are considered for this ensemble. The associated ground-truth classification assigns each time step to one of the four key phases of the simulation. The corresponding classification task therefore consists in identifying, for each time step, to which phase it belongs. The ground-truth classification is as follows:

- **Class 1** (4 members): 0025, 0026, 0027, 0028
- **Class 2** (4 members): 0075, 0076, 0077, 0078,
- **Class 3** (4 members): 0125, 0126, 0127, 0128
- **Class 4** (4 members): 0175, 0176, 0177, 0178,

Ionization Front (2D)

This ensemble is a 2D version of the above ensemble, where the dataset providers have selected a 2D slice in the center of the volume (sampled at 600×248). The associated classification task is therefore identical.

Earthquake

This ensemble is composed of 12 members, given as 3D regular grids (sampled at $375 \times 188 \times 50$, implicitly triangulated by TTK). Each member corresponds to a time step of the simulation of an earthquake at the

San Andreas fault [ODM+06]. For this application, the shock wave can be tracked with the local maxima of the wave front velocity magnitude (this scalar field is pre-processed to pre-simplify all saddle-maxima pairs with a persistence smaller than 0.05% of the data range). Thus, split trees (capturing maxima) are considered for this ensemble. The associated ground-truth classification assigns each time step to one of the three key phases of the simulation. The corresponding classification task therefore consists in identifying, for each time step, to which phase it belongs. The ground-truth classification is as follows:

- **Class 1** (4 members): 002700, 002900, 003100, 003300
- **Class 2** (4 members): 007700, 007900, 008100, 008300
- **Class 3** (4 members): 011700, 011900, 012100, 012300

Isabel

This ensemble is composed of 12 members, given as 3D regular grids (sampled at $250 \times 250 \times 50$, implicitly triangulated by TTK). Each member corresponds to a time step of the simulation of the Isabel hurricane [WBKS04]. This ensemble has been used in previous work [FFST18, VBT20] and the corresponding VTK files are available at the following address: https://github.com/julesvidal/wasserstein-pd-barycenter. In this application, the eyewall of the hurricane is typically characterized by high wind velocities, well captured by the the maxima of the flow velocity. Thus, split trees (capturing maxima) are considered for this ensemble. The associated ground-truth classification assigns each time step to one of the three key phases (formation, drift, landfall) of the hurricane simulation. The corresponding classification task therefore consists in identifying, for each member, to which key phase it belongs. The ground-truth classification is as follows:

- **Class 1** (4 members): 2, 3, 4, 5
- **Class 2** (4 members): 30, 31, 32, 33
- **Class 3** (4 members): 45, 46, 47, 48

Starting Vortex

This ensemble is composed of 12 members, given as 2D regular grids (sampled at 1500×1000, implicitly triangulated by TTK). It has been generated with the Gerris flow solver [Pop06] and was provided in previous work [FFST18, VBT20]. It is available at the following address: https://github.com/julesvidal/wasserstein-pd-barycenter. The data models flow turbulence behind a wing, for two ranges of wing inclination angles. The considered scalar field is the orthogonal component of the curl of the flow velocity. This ensemble corresponds to a parameter study (in this case, studying the effect of the wing configuration on turbulence), which is a typical task in numerical simulation. In this application, salient extrema are typically considered as reliable estimations of the center of vortices. Thus, each run is represented by two merge trees (the join tree – capturing minima, and the split tree, capturing maxima), which are processed independently by our algorithms. The associated ground-truth classification assigns members computed with similar inclination angles to the same class. The corresponding classification task therefore consists in identifying, for a given ensemble member, its correct wing configuration class. The ground-truth classification is as follows:

- **Class 1** (6 members): Angle=2, Angle=3, Angle=4, Angle=5, Angle=6, Angle=8
- **Class 2** (6 members): Angle=38, Angle=39, Angle=40, Angle=41, Angle=42, Angle=43

Sea Surface Height

This ensemble is composed of 48 members, given as 2D regular grids (sampled at 1440×720, implicitly triangulated by TTK). Each member corresponds to an observation of the sea surface height at the surface of the Earth, taken in January, April, July and October 2012. The original data can be found at the following address: https://ecco.jpl.nasa.gov/products/all/. This ensemble has been used in previous work [FFST18, VBT20] and the corresponding VTK files are available at the following address: https://github.com/julesvidal/wasserstein-pd-barycenter. In this application, the features of interest are the center of eddies, which can be reliably estimated with height extrema. Thus, each observation is represented by two merge trees (the join tree – capturing minima, and the split

tree, capturing maxima), which are processed independently by our algorithms. The associated ground-truth classification assigns observations acquired in the same month to the same class. The corresponding classification task therefore consists in identifying, for each observation (taken at a specified date), the season in which it has been acquired. The ground-truth classification is as follows:

- **Class 1** (12 members): 20120111, 20120115, 20120116, 20120117, 20120118, 20120119, 20120120, 20120121, 20120123, 20120128, 20120129
- **Class 2** (12 members): 20120419, 20120420, 20120421, 20120422, 20120423, 20120424, 20120425, 20120426, 20120427, 20120428, 20120429, 20120430
- **Class 3** (12 members): 20120711, 20120712, 20120713, 20120714, 20120715, 20120716, 20120717, 20120718, 20120719, 20120720, 20120721, 20120722
- **Class 4** (12 members): 20121008, 20121009, 20121010, 20121011, 20121012, 20121016, 20121017, 20121018, 20121019, 20121020, 20121022, 20121023

Vortex Street

This ensemble is composed of 45 members, given as 2D regular grids (sampled at 300×100, implicitly triangulated by TTK). It has been generated with the Gerris flow solver [Pop06] and was provided in previous work [FFST18, VBT20]. It is available at the following address: https://github.com/julesvidal/wasserstein-pd-barycenter. The data models flow turbulence behind an obstacle. The considered scalar field is the orthogonal component of the curl of the flow velocity, for 5 fluids of different viscosity. This ensemble corresponds to a parameter study (in this case, studying the effect of viscosity on turbulence), which is a typical task in numerical simulation. In this application, salient extrema are typically considered as reliable estimations of the center of vortices. Thus, each run is represented by two merge trees (the join tree – capturing minima, and the split tree, capturing maxima), which are processed independently by our algorithms. The associated ground-truth classification assigns members computed with similar viscosities to the same class. The corresponding classification task therefore consists in identifying, for a given ensem-

ble member, its correct viscosity class. The ground-truth classification is as follows:

- **Class 1** (9 members): Viscosity=100.0, Viscosity=100.1, Viscosity=100.2, Viscosity=100.3, Viscosity=100.4, Viscosity=100.5, Viscosity=100.6, Viscosity=100.7, Viscosity=100.9
- **Class 2** (9 members): Viscosity=160.0, Viscosity=160.1, Viscosity=160.2, Viscosity=160.3, Viscosity=160.4, Viscosity=160.5, Viscosity=160.6, Viscosity=160.7, Viscosity=160.8
- **Class 3** (9 members): Viscosity=200.0, Viscosity=200.1, Viscosity=200.2, Viscosity=200.3, Viscosity=200.4, Viscosity=200.5, Viscosity=200.6, Viscosity=200.7, Viscosity=200.8
- **Class 4** (9 members): Viscosity=50.0, Viscosity=50.1, Viscosity=50.2, Viscosity=50.3, Viscosity=50.5, Viscosity=50.6, Viscosity=50.7, Viscosity=50.8, Viscosity=50.9
- **Class 5** (9 members): Viscosity=60.1, Viscosity=60.2, Viscosity=60.3, Viscosity=60.4, Viscosity=60.5, Viscosity=60.6, Viscosity=60.7, Viscosity=60.8, Viscosity=60.9

Appendix: Parameter analysis

B

In this appendix, we study the practical effect of the pre-processing parameters of our approaches. In particular, we extend the empirical stability evaluation of our metric (Section 3.3) with regard to all the pre-processing parameters, we illustrate their effect on geodesic computation (Section 3.4) and on the specific task of MT-PGA computation (Section 4.2).

B.1 Interpretation

The first pre-processing parameter of our approaches is $\epsilon_1 \in [0, 1]$. It dictates the merge of saddles in the input trees, to mitigate saddle swap instabilities, as previously documented by Sridharamurthy et al. [SMKN20]. Adjacent saddles in the input trees are merged if their relative difference in scalar value (relative to the largest function difference between adjacent saddles) is *smaller* than ϵ_1. For $\epsilon_1 = 0$, no saddle merge is performed whereas for $\epsilon_1 = 1$, all saddles are merged and $W_2^{\mathcal{T}}$ becomes equivalent to the L^2 Wasserstein distance between persistence diagrams, noted $W_2^{\mathcal{D}}$.

The local normalization step of our framework (Section 3.4.2) guarantees the topological consistency of the interpolated merge trees (Figure 3.7). However, this normalization shrinks the birth/death values of all the input branches to the interval $[0, 1]$, irrespective of their original persistence. To mitigate this effect, the input BDTs are pre-processed, so that branches with small initial persistence (i.e. *small branches*) are not given too much importance in the metric. In particular, small branches are moved up the input BDT if their persistence relative to their parent is *larger* than $\epsilon_2 \in [0, 1]$. When $\epsilon_2 = 0$, all branches are moved up to the root of the BDT and again, $W_2^{\mathcal{T}}$ becomes equivalent to $W_2^{\mathcal{D}}$. When $\epsilon_2 = 1$, no branch is moved up the BDT and ϵ_2 has no effect on the outcome (i.e. the input BDT is left unchanged). In practice, we recommend the default

value $\epsilon_2 = 0.95$: if a branch b has a nearly identical persistence to that of its parent b', it is moved higher in the BDT, so that its normalized persistence becomes nearly identical to that of its parent b' (instead of being artificially larger due to the local normalization).

The parameter $\epsilon_3 \in [0,1]$ further restricts the application of the above BDT pre-processing, by only considering (for displacement up the BDT) the branches with a relative persistence (with respect to the overall data range) smaller than ϵ_3. When $\epsilon_3 = 1$, all branches are subject to the above pre-processing and ϵ_2 fully dictates the BDT pre-processing. When $\epsilon_3 = 0$, no branch is moved up the BDT and the two parameters ϵ_2 and ϵ_3 have no effect on the outcome. In practice, we recommend the default value $\epsilon_3 = 0.9$, which prevents the most persistent branches from moving up the BDT.

Overall, when the parameters $(\epsilon_1, \epsilon_2, \epsilon_3)$ are set to the values $(0,1,1)$, the input trees are *not* pre-processed by the above procedures (i.e. they are left unchanged) and their structure has a strong impact on $W_2^{\mathcal{T}}$. When $\epsilon_1 = 1$ or when $\epsilon_2 = 0$, $W_2^{\mathcal{T}}$ becomes equivalent to $W_2^{\mathcal{D}}$ and the structure of the input trees has no impact anymore on the metric. In-between values balance the importance of the structure of the trees on the metric. We recommend the default values $(0.05, 0.95, 0.9)$, which provides an acceptable stability with regard to saddle swaps (mitigated by ϵ_1) and which gives a reasonable importance to small branches in the metric (controlled by ϵ_2 and ϵ_3, which are dependent parameters).

B.2 Metric Stability

Figure 3.13 provides an empirical stability evaluation of our new metric $W_2^{\mathcal{T}}$, as a function of an input perturbation, modeled by a random noise of amplitude ϵ. In particular, this experiment is achieved for several values of ϵ_1. The conclusion of this experiment is that $W_2^{\mathcal{T}}$ is not stable when $\epsilon_1 = 0$ (sudden increase in $W_2^{\mathcal{T}}$ for small values of ϵ) and that it is stable when $\epsilon_1 = 1$ (as anticipated [TMMH14]). For in-between values, $W_2^{\mathcal{T}}$ is stable until a transition point (colored dots in Figure 3.13), located at increasing noise levels (ϵ) for increasing values of ϵ_1. In particular, for the recommended default value $\epsilon_1 = 0.05$, $W_2^{\mathcal{T}}$ is stable up to a perturbation noise of amplitude 16% (of the overall data range).

In the following, we perform the same study for the other parameters of our approach, ϵ_2 and ϵ_3. Figure B.1 studies the practical stability of $W_2^{\mathcal{T}}$, for several values of ϵ_2. For this experiment, ϵ_3 has been set to 1 (then,

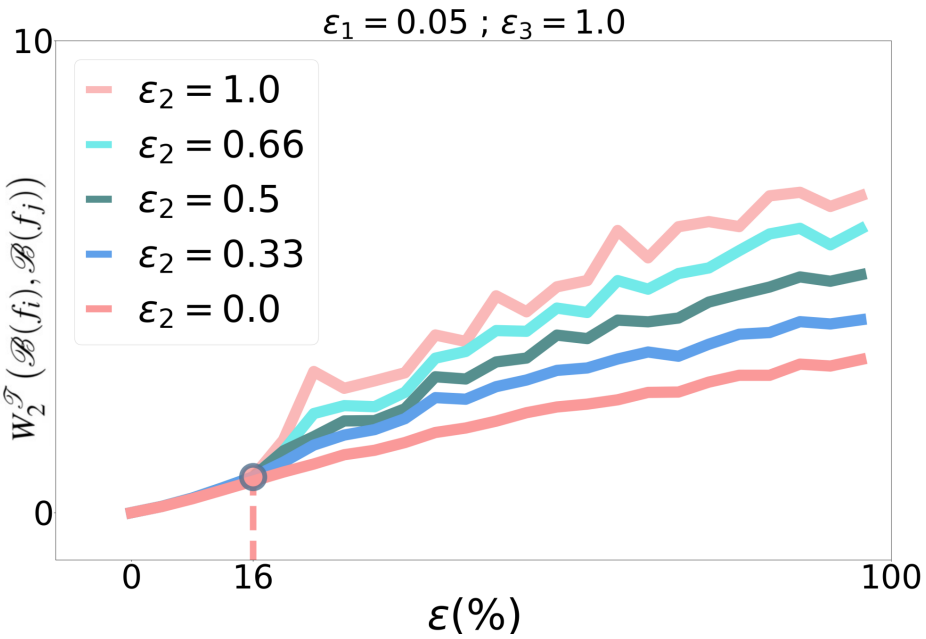


Figure B.1 – *Empirical stability evaluation with regard to ϵ_2. Given an input scalar field f_i, a noisy version f_j is created by inserting a random noise of increasing amplitude ϵ (cf. Figure 3.13). The evolution of $W_2^{\mathcal{T}}(\mathcal{B}(f_i), \mathcal{B}(f_j))$ with ϵ is reported for varying values of ϵ_2.*

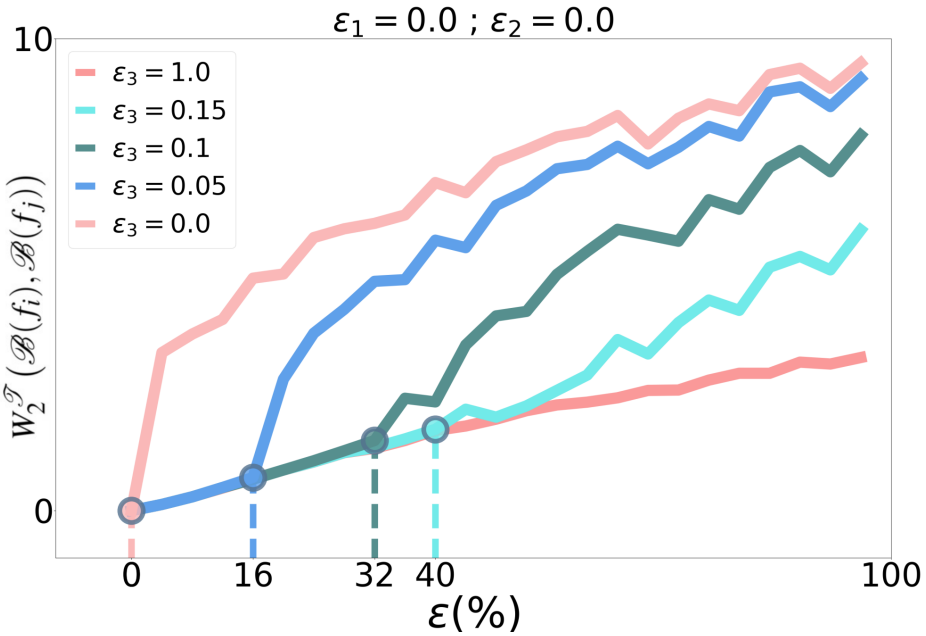


Figure B.2 – *Empirical stability evaluation with regard to ϵ_3. Given an input scalar field f_i, a noisy version f_j is created by inserting a random noise of increasing amplitude ϵ (cf. Figure 3.13). The evolution of $W_2^{\mathcal{T}}(\mathcal{B}(f_i), \mathcal{B}(f_j))$ with ϵ is reported for varying values of ϵ_3.*

only ϵ_2 has an impact on the BDT pre-processing described in the previous section). Moreover, ϵ_1 has been set to its recommended value, 0.05. Several curves are reported, one per ϵ_2 values. For $\epsilon_2 = 0$, all branches are moved up the BDT (irrespective of ϵ_1) and $W_2^{\mathcal{T}}$ becomes equivalent to $W_2^{\mathcal{D}}$ and the corresponding curve (red) exactly coincides with the light blue curve of Figure 3.13 ($\epsilon_1 = 1$). For $\epsilon_2 = 1$, the input BDT is not pre-processed at all and the corresponding curve (pink) exactly coincides with the cyan curve of Figure 3.13 (obtained for the default value $\epsilon_1 = 0.05$). In-between values of ϵ_2 result in continuous transitions between these two extreme cases (blue, green and cyan curves).

Figure B.2 studies the practical stability of $W_2^{\mathcal{T}}$, for several values of ϵ_3. For this experiment, we set $\epsilon_1 = 0$ and $\epsilon_2 = 0$, to better isolate the effect of ϵ_3. When $\epsilon_3 = 1$, all the branches of the input BDTs are subject to the BDT pre-processing. Since $\epsilon_2 = 0$, all branches are moved up to the root and $W_2^{\mathcal{T}}$ becomes equivalent to $W_2^{\mathcal{D}}$ and the corresponding curve (red) exactly coincides with the light blue curve of Figure 3.13 ($\epsilon_1 = 1$). When $\epsilon_3 = 0$, no branch is moved up in the input BDTs and the corresponding curve (pink) exactly coincides with the grey curve of Figure 3.13 ($\epsilon_1 = 0$). In-between values of ϵ_3 result in transitions between these two extreme cases (blue, green and cyan curves), with transition points (similar to Figure 3.13), before which $W_2^{\mathcal{T}}$ is stable. Note however, that since it is dependent on ϵ_2 (default value: 0.95), ϵ_3 has only a very mild practical impact on the metric.

B.3 Geodesic Analysis

Figures B.3, B.4 and B.5 respectively illustrate the effect of the parameters ϵ_1, ϵ_2 and ϵ_3 on the geodesics between merge trees. In particular, each figure shows, on the left, the geodesic obtained with a disabling value of the parameter (no effect on the computation). In contrast, the right side of each figure shows the geodesic obtained with the recommended default value of the parameter, to clearly visualize its impact.

Overall, as discussed in the detailed captions, these three parameters have the effect of moving branches up the input BDTs, hence reducing the structural impact of the trees on the metric, but also improving its stability (as discussed in Section B.2). In the data, moving a branch up the BDT corresponds to only slight modifications, which consist in reconnecting maxima to distinct saddles. For each parameter, the resulting pre-processing addresses cases where nearby saddles have very close function values,

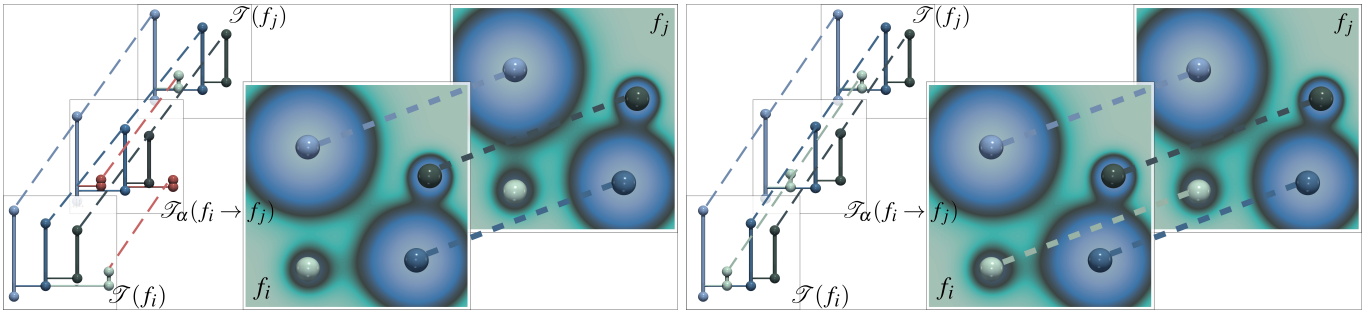


Figure B.3 – *Impact of the parameter ϵ_1 on geodesic computation (left: $\epsilon_1 = 0$, right: $\epsilon_1 = 0.05$). In this example (left), the white branch in $\mathcal{T}(f_i)$ is not matched to the white branch in $\mathcal{T}(f_j)$ as they have distinct depths in the corresponding BDTs (2 versus 1). However, these features are visually similar in the data (Gaussians with the white maximum in f_i and f_j, bottom left corner of the domain). With $\epsilon_1 = 0.05$ (right), the saddle of the white branch in $\mathcal{T}(f_i)$ gets merged with its ancestor saddle (whose f_i value was less than ϵ_1 away). Consequently, the white branch gets moved up the BDT (the white branch is attached to the main light blue branch in $\mathcal{T}(f_i)$, right). Since they now have identical depths in the corresponding BDTs, the white branches of $\mathcal{T}(f_i)$ and $\mathcal{T}(f_j)$ can now be matched together (right), which results in an overall matching (and geodesic) between these two trees which better conveys the resemblance between the two scalar fields f_i and f_j. Equivalently, one can interpret this procedure of saddle merge in the input trees as a modification of the input scalar field, turning f_i into f_j. In particular, this field modification disconnects the Gaussian with the white maximum from the Gaussian with the dark blue maximum (f_i) and reconnects it to the Gaussian with the light blue maximum (f_j).*

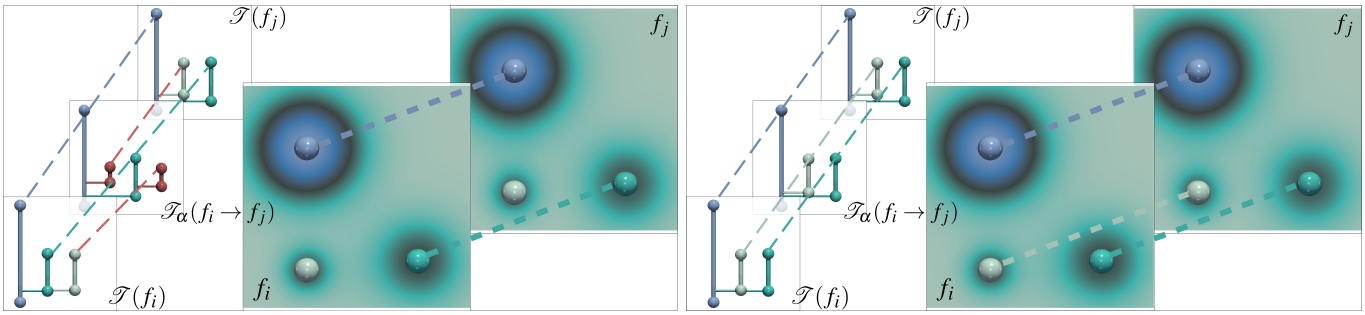


Figure B.4 – *Impact of the parameter ϵ_2 on geodesic computation (left: $\epsilon_2 = 1$, right: $\epsilon_2 = 0.95$). In this example (left), the white branch in $\mathcal{T}(f_i)$ is not matched to the white branch in $\mathcal{T}(f_j)$ as they have distinct depths in the corresponding BDTs (2 versus 1). Moreover, given the function difference between the white branch's saddle and its ancestor, that branch cannot be moved up the BDT under the effect of the ϵ_1 procedure (above). The white branch in $\mathcal{T}(f_i)$ has a persistence nearly identical to its parent (cyan). Thus, after local normalization (necessary to guarantee the topological consistency of the interpolated trees), its normalized persistence would become artificially high, which can have an undesirable effect on the metric. The BDT pre-processing addresses this issue and moves up the BDT branches with a relative persistence to their parent larger than ϵ_2 (recommended default value: 0.95). In this example (right), the white branch in $\mathcal{T}(f_i)$ moves up the BDT and becomes adjacent to the main light blue branch in $\mathcal{T}(f_i)$. Since they now have identical depths in the corresponding BDTs, the white branches of $\mathcal{T}(f_i)$ and $\mathcal{T}(f_j)$ can now be matched together (right), which better conveys the resemblance between the two scalar fields f_i and f_j. Equivalently, one can interpret this procedure of BDT pre-processing as a modification of the input scalar field, turning f_i into f_j. In particular, this field modification disconnects the Gaussian with the white maximum from the Gaussian with the cyan maximum (f_i) and reconnects it to the Gaussian with the light blue maximum (f_j).*

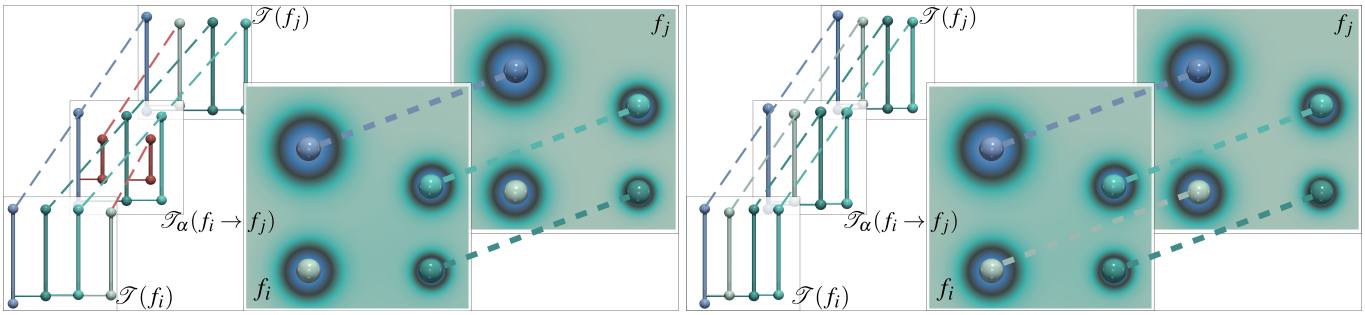


Figure B.5 – *Effect of the parameter ϵ_3 on geodesic computation (left: $\epsilon_3 = 0$, right: $\epsilon_3 = 0.9$). In this example (left), the white branch in $\mathcal{T}(f_i)$ is not matched to the white branch in $\mathcal{T}(f_j)$ as they have distinct depths in the corresponding BDTs (3 versus 1). Applying the above BDT pre-processing (ϵ_2) to all branches would move the cyan branch in $\mathcal{T}(f_i)$ up the BDT, which would prevent it to match to the cyan branch in $\mathcal{T}(f_j)$. The parameter ϵ_3 restricts the application of the above BDT pre-processing and prevents the movement of the most persistent branches (relative persistence larger than ϵ_3, default: 0.9). In this example (right), the white branch in $\mathcal{T}(f_i)$ moves up the BDT and becomes adjacent to the main light blue branch in $\mathcal{T}(f_i)$. Since they now have identical depths in the corresponding BDTs, the white branches of $\mathcal{T}(f_i)$ and $\mathcal{T}(f_j)$ can now be matched together (right), which results in an overall matching (and geodesic) between these two trees which better conveys the resemblance between the two scalar fields f_i and f_j. Equivalently, one can interpret this procedure on the BDTs as a modification of the input scalar field, turning f_i into f_j. In particular, this field modification disconnects the Gaussian with the white maximum from the Gaussian with the dark green maximum (f_i) and reconnects it to the Gaussian with the light blue maximum (f_j).*

which impacts the stability of the metric. Similarly to Sridharamurthy et al. [SMKN20], we mitigate this effect with ϵ_1, but we also introduce ϵ_2 and ϵ_3 to specifically limit the importance in the metric of branches with persistence close to that of their parent.

B.4 MT-PGA Analysis

In this section, we replicate the same experimental protocol, but this time for the specific task of MT-PGA computation. Specifically, Figures B.6, B.7 and B.8 respectively illustrate the effect of the parameters ϵ_1, ϵ_2, and ϵ_3 on the MT-PGA basis.

As described in Chapter 3, in the data, moving a branch up the BDT corresponds to only slight modifications, which consists in reconnecting maxima to distinct saddles. For each parameter (ϵ_1, ϵ_2, and ϵ_3), the resulting pre-processing addresses cases where nearby saddles have close function values, which impacts the stability of the metric. Similarly to Sridaharamurthy et al. [SMKN20], we mitigate this effect with ϵ_1, but we also introduce ϵ_2 and ϵ_3 to specifically limit the importance in the metric of branches with a persistence close to that of their parents Section 3.4.2.

For each figure, we study an ensemble consisting of two main clusters:

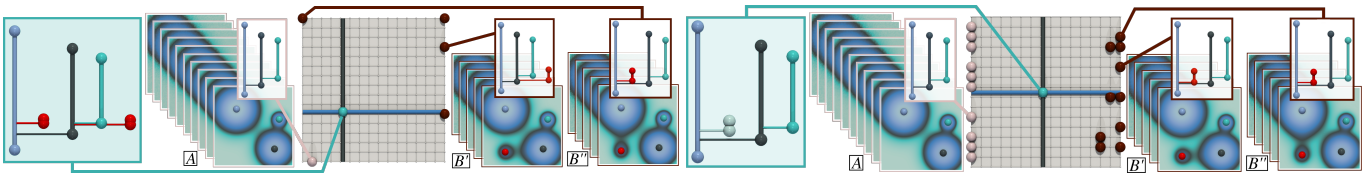


Figure B.6 – *Impact of the parameter ϵ_1 on MT-PGA computation (in this example, left: $\epsilon_1 = 0$, right: $\epsilon_1 = 0.3$). Initially (left), the red branch in the cluster B' is not matched to the red branch in the cluster B'' as they have distinct depths in the corresponding BDTs (2 versus 1). However, these features are visually similar in the data (Gaussians with the red maximum in B' and B'', bottom left corner of the domain). After the ϵ_1 pre-processing (right), the saddle of the red branch in B'' gets merged with its ancestor saddle (whose scalar value was less than ϵ_1 away). Consequently, the red branch gets moved up the BDT (i.e. the red branch is attached to the main light blue branch in B'', right). Since they now have identical depths in the corresponding BDTs, the red branches of the sub-clusters B' and B'' can now be matched together (right), which results in an overall matching between these two trees which better conveys the resemblance between the two sub-clusters B' and B''. Equivalently, one can interpret this procedure of saddle merge in the input trees as a modification of the input scalar field, turning the Gaussian mixture B' into B''. In particular, this field modification disconnects the Gaussian with the red maximum from the Gaussian with the black maximum (B') and reconnects it to the Gaussian with the light blue maximum (B''). Overall, after the ϵ_1 pre-processing (right), the MT-PGA better distinguishes the cluster A (pink spheres, left of the planar layout) from the cluster B (dark red spheres, right of the planar layout).*

A (pink spheres, left of Figures B.6, B.7 and B.8) and B (dark red spheres, right of Figures B.6, B.7 and B.8). These clusters have been synthesized by considering Gaussian mixtures (such that B has one more prominent feature than A) and by generating the other members of the ensemble with variants of these two patterns, by inserting a random additive noise. Specifically, the cluster B is synthesized out of 2 slightly distinct Gaussian mixtures, yielding two *artificial* sub-clusters B' and B'', such that, in each case, the red branch directly connects to a different branch in the sub-clusters B' and B''. Then each figure visualizes the impact of each parameter ϵ_1, ϵ_2, and ϵ_3 on the displacement of the red branch, and hence on the resulting MT-PGA basis.

Overall, as discussed in the detailed captions, these three parameters have the effect of moving branches up the input BDTs, hence reducing the structural impact of these branches on the metric, but also improving its stability. In Figures B.6, B.7 and B.8, by moving the red branch up, a larger section of the BDTs of the sub-clusters B' and B'' become isomorphic, and thus, the two sub-patterns of the cluster B become closer to each other in $\mathbb{B}$ and the clusters A and B become better differentiated in the MT-PGA basis (cluster A on the left of the planar layout, pink spheres, cluster B on the right of the planar layout, dark red spheres).

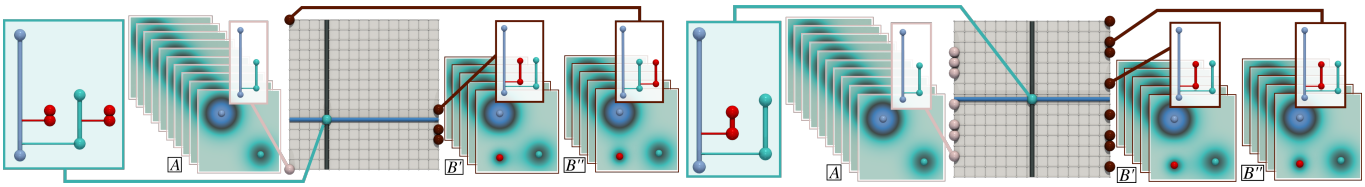


Figure B.7 – *Impact of the parameter ϵ_2 on MT-PGA computation (in this example, left: $\epsilon_2 = 1$, right: $\epsilon_2 = 0.8$). Initially (left), the red branch in B' is not matched to the red branch in B'' as they have distinct depths in the corresponding BDTs (1 versus 2). The red branch in B'' has a persistence nearly identical to its parent (cyan). Thus, after local normalization (necessary to guarantee the topological consistency of the interpolated trees), its normalized persistence would become artificially high, which can have an undesirable effect on the metric. The BDT pre-processing addresses this issue and moves up the BDT branches with a relative persistence to their parent larger than ϵ_2. After the ϵ_2 pre-processing (right), the red branch in B'' moves up the BDT and becomes adjacent to the main light blue branch. Since they now have identical depths in the corresponding BDTs, the red branches of B' and B'' can now be matched together (right), which better conveys the resemblance between the two sub-clusters B' and B''. Equivalently, one can interpret this procedure of BDT pre-processing as a modification of the input scalar field, turning the Gaussian mixture B' into B''. In particular, this field modification disconnects the Gaussian with the red maximum from the Gaussian with the cyan maximum (B'') and reconnects it to the Gaussian with the light blue maximum (B'). Overall, after the ϵ_2 pre-processing (right), the MT-PGA better distinguishes the cluster A (pink spheres, left of the planar layout) from the cluster B (dark red spheres, right of the planar layout).*

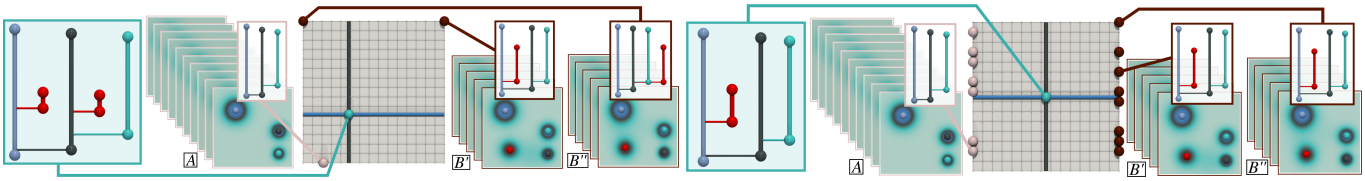


Figure B.8 – *Impact of the parameter ϵ_3 on MT-PGA computation (in this example, left: $\epsilon_3 = 0$, right: $\epsilon_3 = 0.7$). Initially (left), the red branch in B'' is not matched to the red branch in B' as they have distinct depths in the corresponding BDTs (1 versus 3). The parameter ϵ_3 restricts the application of the above BDT pre-processing (ϵ_2) and prevents the movement of the most persistent branches (relative persistence larger than ϵ_3). After the ϵ_3 pre-processing (right), the red branch in B'' moves up the BDT and becomes adjacent to the main light blue branch. Since they now have identical depths in the corresponding BDTs, the red branches of B' and B'' can now be matched together (right), which results in an overall matching between these two trees which better conveys the resemblance between the two sub-clusters B' and B''. Equivalently, one can interpret this procedure on the BDTs as a modification of the input scalar field, turning the Gaussian mixture B' into B''. In particular, this field modification disconnects the Gaussian with the red maximum from the Gaussian with the cyan maximum (B'') and reconnects it to the Gaussian with the light blue maximum (B'). Overall, after the ϵ_3 pre-processing (right), the MT-PGA better distinguishes the cluster A (pink spheres, left of the planar layout) from the cluster B (dark red spheres, right of the planar layout).*

Bibliography

[AAPW18] Keri Anderson, Jeffrey Anderson, Sourabh Palande, and Bei Wang. Topological data analysis of functional MRI connectivity in time and space domains. In *MICCAI Workshop on Connectomics in NeuroImaging*, 2018.

[ABG+15] Utkarsh Ayachit, Andrew C. Bauer, Berk Geveci, Patrick O'Leary, Kenneth Moreland, Nathan Fabian, and Jeffrey Mauldin. ParaView Catalyst: Enabling In Situ Data Analysis and Visualization. In *ISAV*, 2015.

[ACE+17] Henry Adams, Sofya Chepushtanova, Tegan Emerson, Eric Hanson, Michael Kirby, Francis Motta, Rachel Neville, Chris Peterson, Patrick Shipman, and Lori Ziegelmeier. Persistence Images: A Stable Vector Representation of Persistent Homology. *Journal of Machine Learning Research*, 18(8):1–35, 2017.

[AE13] Tushar Athawale and Alireza Entezari. Uncertainty quantification in linear interpolation for isosurface extraction. *IEEE Transactions on Visualization and Computer Graphics*, (12):2723–2732, 2013.

[AGL05] James Ahrens, Berk Geveci, and Charles Law. ParaView: An End-User Tool for Large-Data Visualization. *The Visualization Handbook*, pages 717–731, 2005.

[AJ19] T. Athawale and C. R. Johnson. Probabilistic Asymptotic Decider for Topological Ambiguity Resolution in Level-Set Extraction for Uncertain 2D Data. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1163–1172, 2019.

[AMJ+19] Tushar M. Athawale, Dan Maljovec, Chris R. Johnson, Valerio Pascucci, and Bei Wang. Uncertainty Visualization of 2D Morse Complex Ensembles Using Statistical Summary Maps. *CoRR*, abs/1912.06341, 2019.

[AN15] Aditya Acharya and Vijay Natarajan. A parallel and memory efficient algorithm for constructing the contour tree. In *IEEE PacificViz*, 2015.

[ASE16] T. Athawale, E. Sakhaee, and A. Entezari. Isosurface visualization of data with nonparametric models for uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):777–786, 2016.

[AVRT16] Rushil Anirudh, Vinay Venkataraman, Karthikeyan Natesan Ramamurthy, and Pavan K. Turaga. A Riemannian Framework for Statistical Analysis of Topological Persistence Diagrams. In *IEEE CVPR Workshops*, 2016.

[BAA+16] Andrew C. Bauer, H. Abbasi, J. Ahrens, H. Childs, B. Geveci, S. Klasky, K. Moreland, Patrick O'Leary, V. Vishwanath, B. Whitlock, and E W. Bethel. In-situ methods, infrastructures, and applications on high performance computing platforms. *Computer Graphics Forum*, 2016.

[Bab88] Charles Babbage. The Analytical Engine. *Proceedings of the British Association. Bath*, 1888.

[Ban67] T. F. Banchoff. Critical points and curvature for embedded polyhedral surfaces. *The American Mathematical Monthly*, 45(1):245–256, 1967.

[BCLM21] Brian Bollen, Erin Chambers, Joshua A. Levine, and Elizabeth Munch. Reeb Graph Metrics from the Ground Up. *CoRR*, 2021.

[BDSS18] Alexander Bock, Harish Doraiswamy, Adam Summers, and Cláudio T. Silva. TopoAngler: Interactive Topology-Based Extraction of Fishes. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):812–821, 2018.

[BEHP03] Peer-Timo Bremer, Herbert Edelsbrunner, Bernd Hamann, and Valerio Pascucci. A Multi-Resolution Data Structure for 2-Dimensional Morse Functions. In *Proc. of IEEE VIS*, 2003.

[Ber81] Dimitri P. Bertsekas. A new algorithm for the assignment problem. *Mathematical Programming*, 21(1):152–171, 1981.

[BGL+18] Harsh Bhatia, Attila G. Gyulassy, Vincenzo Lordi, John E. Pask, Valerio Pascucci, and Peer-Timo Bremer. Topoms: Comprehensive topological exploration for molecular and condensed-matter systems. *J. of Computational Chemistry*, 39(16):936–952, 2018.

[BGSF08] S. Biasotti, D. Giorgio, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 392(1-3):5–22, 2008.

[BGW14] U. Bauer, X. Ge, and Y. Wang. Measuring distance between Reeb graphs. In *Symposium on Computational Geometry*, 2014.

[BHJ+14] Georges Pierre Bonneau, Hans Christian Hege, Chris R. Johnson, Manuel M. Oliveira, Kristin Potter, Penny Rheingans, and Thomas Schultz". Overview and state-of-the-art of uncertainty visualization. *Mathematics and Visualization*, 37:3–27, 2014.

[BJB+12] H. Bhatia, S. Jadhav, P. Bremer, G. Chen, J. A. Levine, L. G. Nonato, and V. Pascucci. Flow visualization with quantified spatial and temporal errors using edge maps. *IEEE Transactions on Visualization and Computer Graphics*, 18(9):1383–1396, 2012.

[BK88] H Bourlard and Y Kamp. Auto-Association by Multilayer Perceptrons and Singular Value Decomposition. *Biological Cybernetics*, 59, 1988.

[BKR14] Ulrich Bauer, Michael Kerber, and Jan Reininghaus. Distributed computation of persistent homology. In *Algorithm Engineering and Experiments*, 2014.

[BMBF+19] Talha Bin Masood, Joseph Budin, Martin Falk, Guillaume Favelier, Christoph Garth, Charles Gueunet, Pierre Guillou, Lutz Hofmann, Petar Hristov, Adhitya Kamakshidasan, Christopher Kappe, Pavol Klacansky, Patrick Laurin, Joshua Levine, Jonas Lukasczyk, Daisuke Sakurai, Maxime Soler, Peter Steneteg, Julien Tierny, Will Usher, Jules Vidal, and Michal Wozniak. An Overview of the Topology ToolKit. In *TopoInVis*, 2019.

[BNP+21] Nick Brown, Rupert Nash, Piero Poletti, Giorgio Guzzetta, Mattia Manica, Agnese Zardini, Markus Flatken, Jules Vidal, Charles Gueunet, Evgenij Belikov, Julien Tierny, Artur Podobas, Wei Der Chien, Stefano Markidis, and Andreas Gerndt. Utilising urgent computing to tackle the spread of mosquito-borne diseases. In *IEEE/ACM UrgentHPC@SC*, 2021.

[BRPP15] Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and Radon Wasserstein Barycenters of Measures. *Journal of Mathematical Imaging and Vision*, 1(51):22–45, 2015.

[Bub15] Peter Bubenik. Statistical topological data analysis using persistence landscapes. *J. Mach. Learn. Res.*, 16:77–102, 2015.

[BWT+11] 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*, 17(9):1307–1324, 2011.

[BYM+14] Kenes Beketayev, Damir Yeliussizov, Dmitriy Morozov, Gunther H. Weber, and Bernd Hamann. Measuring the distance between merge trees. In *TopoInVis*. 2014.

[CCF+13] J. Chen, Alok Choudhary, S. Feldman, B. Hendrickson, C. R. Johnson, R. Mount, V. Sarkar, V. White, and D. Williams. *Synergistic Challenges in Data-Intensive Science and Exascale Computing: DOE ASCAC Data Subcommittee Report*. Department of Energy Office of Science, March 2013. Type: Report.

[CD14] Marco Cuturi and Arnaud Doucet. Fast computation of wasserstein barycenters. In *ICML*, 2014.

[CEH05] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. In *Symposium on Computational Geometry*, 2005.

[CKV13] M. Emre Celebi, Hassan A. Kingravi, and Patricio A. Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Syst. Appl.*, 2013.

[CSA00] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. In *Symp. on Dis. Alg.*, 2000.

[CSB+18] Elsa Cazelles, Vivien Seguy, Jérémie Bigot, Marco Cuturi, and Nicolas Papadakis. Geodesic PCA versus Log-PCA of Histograms in the Wasserstein Space. *SIAM J. Sci. Comput.*, 40(2), 2018.

[CSvdP04] Hamish A. Carr, Jack Snoeyink, and Michiel van de Panne. Simplifying Flexible Isosurfaces Using Local Geometric Measures. In *IEEE VIS*, 2004.

[Cut13] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NIPS*. 2013.

[CWSA16] Hamish Carr, Gunther Weber, Christopher Sewell, and James Ahrens. Parallel peak pruning for scalable SMP contour tree computation. In *IEEE LDAV*, 2016.

[DAWL23] Thomas Davies, Jack Aspinall, Bryan Wilder, and Tran-Thanh Long. Fuzzy c-means clustering in persistence diagram space for deep learning model selection. In Sophia Sanborn, Christian Shewmake, Simone Azeglio, Arianna Di Bernardo, and Nina Miolane, editors, *Proceedings of the 1st NeurIPS Workshop on Symmetry and Geometry in Neural Representations*, volume 197 of *Proceedings of Machine Learning Research*, pages 137–157. PMLR, 03 Dec 2023.

[DFFIM15] Leila De Floriani, Ulderico Fugacci, Federico Iuricich, and Paola Magillo. Morse complexes for shape segmentation and homological analysis: discrete models and algorithms. *Computer Graphics Forum*, 34(2):761–785, 2015.

[DG18] Bartlomiej Dudek and Pawel Gawrychowski. Edit distance between unrooted trees in cubic time. In *International Colloquium on Automata, Languages and Programming*, 2018.

[DHLZ02] Peter Diggle, Patrick Heagerty, K.-Y. Liang, and Scott Zeger. *The Analysis of Longitudinal Data*. Oxford University Press, 2002.

[DN13] Harish Doraiswamy and Vijay Natarajan. Computing Reeb Graphs as a Union of Contour Trees. *IEEE Transactions on Visualization and Computer Graphics*, 19(2):249–262, 2013.

[EH09] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2009.

[EHNP03] Herbert Edelsbrunner, John Harer, Vijay Natarajan, and Valerio Pascucci. Morse-Smale complexes for piecewise linear 3-manifolds. In *Symposium on Computational Geometry*, 2003.

[EHZ01] Herbert Edelsbrunner, John Harer, and Afra Zomorodian. Hierarchical morse complexes for piecewise linear 2-manifolds. In *Symposium on Computational Geometry*, 2001.

[Elk03] C. Elkan. Using the triangle inequality to accelerate k-means. In *ICML*, 2003.

[ELZ02] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological Persistence and Simplification. *Discrete Computational Geometry*, 28(4):511–533, 2002.

[EM90] Herbert Edelsbrunner and Ernst P Mucke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, 1990.

[FBW16] F. Ferstl, K. Bürger, and R. Westermann. Streamline variability plots for characterizing the uncertainty in vector field ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):767–776, 2016.

[FFST18] Guillaume Favelier, Noura Faraj, Brian Summa, and Julien Tierny. Persistence Atlas for Critical Point Variability in Ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1152–1162, 2018.

[FGT16] G. Favelier, C. Gueunet, and J. Tierny. Visualizing ensembles of viscous fingers. In *IEEE SciVis Contest*, 2016.

[FHG+14] David Feng, Bernd Hentschel, Sabine Griessbach, Lars Hoffmann, and Marc von Hobe. The IEEE SciVis Contest. http://sciviscontest.ieeevis.org/2014/, 2014.

[FKRW16] F. Ferstl, M. Kanzler, M. Rautenhaus, and R. Westermann. Visual analysis of spatial variability and global correlations in ensembles of iso-contours. *Computer Graphics Forum*, 35(3):221–230, 2016.

[FLPJ04] P. Thomas Fletcher, Conglin Lu, Stephen M. Pizer, and Sarang C. Joshi. Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE TMI*, 23(8):995–1005, 2004.

[For98] Robin Forman. A User's Guide to Discrete Morse Theory. *AM*, 1998.

[GABCG+14] D. Guenther, R. Alvarez-Boto, J. Contreras-Garcia, J.-P. Piquemal, and J. Tierny. Characterizing Molecular Interactions in Chemical Systems. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2476–2485, 2014.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Prses, 2016.

[GBG+14] A. Gyulassy, P.T. Bremer, R. Grout, H. Kolla, J. Chen, and V. Pascucci. Stability of dissipation elements: A case study in combustion. *Computer Graphics Forum*, 33(3):51–60, 2014.

[GBP19] Attila Gyulassy, Peer-Timo Bremer, and Valerio Pascucci. Shared-Memory Parallel Computation of Morse-Smale Complexes with Improved Accuracy. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1183–1192, 2019.

[GDN+07] Attila Gyulassy, Mark A. Duchaineau, Vijay Natarajan, Valerio Pascucci, Eduardo Bringa, Andrew Higginbotham, and Bernd Hamann. Topologically Clean Distance Fields. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1432–1439, 2007.

[GFJT17] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Task-based Augmented Merge Trees with Fibonacci Heaps,. In *IEEE LDAV*, 2017.

[GFJT19a] Charles Gueunet, Pierre Fortin, Julien Jomier, and Julien Tierny. Task-Based Augmented Contour Trees with Fibonacci Heaps. *IEEE TPDS*, 30(8):1889–1905, 2019.

[GFJT19b] Charles Gueunet, Pierre Fortin, Julien Jomier, and Julien Tierny. Task-based Augmented Reeb Graphs with Dynamic ST-Trees. In *EGPGV*, 2019.

[GGH+16] Christoph Garth, Berk Geveci, Bernd Hentschel, Jorg Kuhnert, Isabel Michel, Theresa-Marie Rhyne, and Simon SchrÃ¶der. The IEEE SciVis Contest. http://sciviscontest.ieeevis.org/2016/, 2016.

[GKL+16] A. Gyulassy, A. Knoll, K.C. Lau, B. Wang, P.T. Bremer, M.E. Papka, L. A. Curtiss, and V. Pascucci. Interstitial and Interlayer Ion Diffusion Geometry Extraction in Graphitic Nanosphere Battery Materials. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):916–925, 2016.

[GKWZ08] Alexander N Gorban, B Kegl, D C Wunsch, and A Zinovyev. *Principal Manifolds for Data Visualization and Dimension Reduction*. Springer, 2008.

[GMO+19] Ellen Gasparovic, Elizabeth Munch, Steve Oudot, Katharine Turner, Bei Wang, and Yusu Wang. Intrinsic interleaving distance for merge trees. *CoRR*, 1908.00063, 2019.

[GST14] David Günther, Joseph Salmon, and Julien Tierny. Mandatory critical points of 2D uncertain scalar fields. *Computer Graphics Forum*, 33(3):31–40, 2014.

[GZ09] A N Gorban and A Y Zinovyev. Principal Graphs and Manifolds. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods and Techniques*. 2009.

[HGTS15] Bernd Hentschel, Berk Geveci, Matthew Turk, and Samuel Skillman. The IEEE SciVis Contest. http://sciviscontest.ieeevis.org/2015/, 2015.

[HLH+16] C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. De Floriani, G. Scheuermann, H. Hagen, and C. Garth. A survey of topology-based methods in visualization. *Computer Graphics Forum*, 35(3):643–667, 2016.

[HMR21] Felix Hensel, Michael Moor, and Bastian Rieck. A survey of topological machine learning methods. *Frontiers Artif. Intell.*, 2021.

[HOGJ13] M. Hummel, H. Obermaier, C. Garth, and K. I. Joy. Comparative visual analysis of lagrangian transport in CFD ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2743–2752, 2013.

[HSKK01] Masaki Hilaga, Yoshihisa Shinagawa, Taku Komura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *ACM SIGGRAPH*, 2001.

[JS03] C. R. Johnson and A. R. Sanderson. A next step: Visualizing errors and uncertainty. *IEEE Computer Graphics and Applications*, 23(5):6–10, 2003.

[Kan42] Leonid Kantorovich. On the translocation of masses. *AS USSR*, 1942.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[kme90] *Partitioning Around Medoids (Program PAM)*, chapter 2, pages 68–125. John Wiley & Sons, Ltd, 1990.

[KMN17] Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams. *ACM J. of Experimental Algorithmics*, 22, 2017.

[Kra10] Martin Kraus. Visualization of uncertain contour trees. In *IVTA*, 2010.

[KRHH11] J. Kasten, J. Reininghaus, I. Hotz, and H.C. Hege. Two-dimensional time-dependent vortex regions based on the acceleration magnitude. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2080–2087, 2011.

[KW78] J B Kruskal and M Wish. Multidimensional Scaling. In *SUPS*, 1978.

[LBM+06] D. E. 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*, 12(5):1053–1060, 2006.

[LCO18] Théo Lacombe, Marco Cuturi, and Steve Oudot. Large Scale computation of Means and Clusters for Persistence Diagrams using Optimal Transport. In *NIPS*, 2018.

[Llo82] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 1982.

[LPW21] Mingzhe Li, Sourabh Palande, and Bei Wang. Sketching merge trees. *CoRR*, 2021.

[LS16] T. Liebmann and G. Scheuermann. Critical Points of Gaussian-Distributed Scalar Fields on Simplicial Grids. *Computer Graphics Forum*, 35(3):361–370, 2016.

[LWL+20] Anna Pia Lohfink, Florian Wetzels, Jonas Lukasczyk, Gunther H. Weber, and Christoph Garth. Fuzzy contour trees: Alignment and joint layout of multiple contour trees. *Computer Graphics Forum*, 39(3):343–355, 2020.

[MAH+05] Alan Maceachren, A.Robinson, Susan Hopper, Steven Gardner, Robert Murray, Mark Gahegan, and Elisabeth Hetzler. Visualizing geospatial information uncertainty: What we know and what we need to know. *CGIS*, 2005.

[MBW14] Dmitriy Morozov, Kenes Beketayev, and Gunther H. Weber. Interleaving distance between merge trees. In *TopoInVis*. 2014.

[MDN12] Senthilnathan Maadasamy, Harish Doraiswamy, and Vijay Natarajan. A hybrid parallel algorithm for computing and tracking level set topology. In *HiPC*, 2012.

[MHRB20] Michael Moor, Max Horn, Bastian Rieck, and Karsten M. Borgwardt. Topological autoencoders. In *ICML*, 2020.

[MKZ+19] Cheng Meng, Yuan Ke, Jingyi Zhang, Mengrui Zhang, Wenxuan Zhong, and Ping Ma. *Large-Scale Optimal Transport Map Estimation Using Projection Pursuit*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[Mon81] Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Académie Royale des Sciences de Paris*, 1781.

[Mun57] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[MWK14] Mahsa Mirzargar, Ross T. Whitaker, and Robert M. Kirby. Curve boxplot: Generalization of boxplot for ensembles of curves. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2654–2663, 2014.

[MWR+16] Daniel Maljovec, Bei Wang, Paul Rosen, Andrea Alfonsi, Giovanni Pastore, Cristian Rabiti, and Valerio Pascucci. Topology-inspired partition-based sensitivity analysis and visualization of nuclear simulations. In *IEEE PacificViz*, 2016.

[ODM+06] Kim Olsen, Steve Day, Bernard Minster, Reagan Moore, Yifeng Cui, Amit Chourasia, Marcus Thiebaux, Hunter Francoeur, Philip Maechling, Steve Cutchin, and Ken Nunes. The IEEE SciVis Contest. http://sciviscontest.ieeevis.org/2006/, 2006.

[OGHT10] M. Otto, T. Germer, H.-C. Hege, and H. Theisel. Uncertain 2D vector Field Topology. *Computer Graphics Forum*, 29(2):347–356, 2010.

[OGT11] M. Otto, T. Germer, and H. Theisel. Uncertain topology of 3D vector fields. *IEEE PacificViz*, 2011.

[OGT19] Malgorzata Olejniczak, André Severo Pereira Gomes, and Julien Tierny. A Topological Data Analysis Perspective on Non-Covalent Interactions in Relativistic Calculations. *International Journal of Quantum Chemistry*, 120(8):e26133, 2019.

[Org04] Organizers. The IEEE SciVis Contest. http://sciviscontest.ieeevis.org/, 2004.

[Par12] Salman Parsa. A deterministic *o(m log m)* time algorithm for the reeb graph. In *Symposium on Computational Geometry*, 2012.

[PCMS04] Valerio Pascucci, Kree Cole-McLaughlin, and Giorgio Scorzelli. Multi-resolution computation and presentation of contour trees. In *IASTED*, 2004.

[Pea95] Karl Pearson. Notes on regression and inheritance in the case of two parents. In *Proceedings of the Royal Society of London*, 1895.

[Pea01] K Pearson. On Lines and Plances of Closest Fit to Sytems of Points in Space. *Philosophical Magazine*, 2:559–572, 1901.

[PG18] John Patchett and Galen Ross Gisler. The IEEE SciVis Contest. http://sciviscontest.ieeevis.org/2018/, 2018.

[PGA13] K. Potter, S. Gerber, and E. W. Anderson. Visualization of uncertainty without a mean. *IEEE Computer Graphics and Applications*, 33(1):75–79, 2013.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019. https://pytorch.org/cppdocs/.

[PH11] K. Pöthkow and H.-C. Hege. Positional Uncertainty of Isocontours: Condition Analysis and Probabilistic Measures. *IEEE Transactions on Visualization and Computer Graphics*, 17(10):1393–1406, 2011.

[PH13] Kai Pöthkow and Hans-Christian Hege. Nonparametric models for uncertainty visualization. *Computer Graphics Forum*, 32(3):131–140, 2013.

[PMW13] T. Pfaffelmoser, M. Mihai, and R. Westermann. Visualizing the variability of gradients in uncertain 2D scalar fields. *IEEE Transactions on Visualization and Computer Graphics*, 19(11):1948–1961, 2013.

[Popo6] Stéphane Popinet. Gerris Flow Solver. http://gfs.sourceforge.net/wiki/index.php/Main_Page, 2006.

[PPH12] Christoph Petz, Kai Pöthkow, and Hans-Christian Hege. Probabilistic local features in uncertain vector fields with spatial correlation. *Computer Graphics Forum*, 31(3pt2):1045–1054, 2012.

[PPH13] Kai Pöthkow, Christoph Petz, and Hans-Christian Hege. Approximate level-crossing probabilities for interactive visualization of uncertain isocontours. *Int. J. Uncert. Quantif.*, 3(2):101–117, 2013.

[PRJ12] Kristi Potter, Paul Rosen, and Christ R Johnson. From quantification to visualization: A taxonomy of uncertainty visualization approaches. *IFIP AICT*, 2012.

[PRW11] Tobias Pfaffelmoser, Matthias Reitinger, and Rüdiger Westermann. Visualizing the positional and geometrical variability of isosurfaces in uncertain scalar fields. *Computer Graphics Forum*, 30(3):951–960, 2011.

[PSBM07] V Pascucci, G Scorzelli, P T Bremer, and A Mascarenhas. Robust on-line computation of Reeb graphs: simplicity and speed. *ACM Transactions on Graphics*, 26(3):58, 2007.

[PVDT22] Mathieu Pont, Jules Vidal, Julie Delon, and Julien Tierny. Wasserstein Distances, Geodesics and Barycenters of Merge Trees. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):291–301, 2022.

[PVT23] Mathieu Pont, Jules Vidal, and Julien Tierny. Principal Geodesic Analysis of Merge Trees (and Persistence Diagrams). *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[PW12] Tobias Pfaffelmoser and Rüdiger Westermann. Visualization of global correlation structures in uncertain 2D scalar fields. *Computer Graphics Forum*, 31(3):1025–1034, 2012.

[PWB+09] K. Potter, A. Wilson, P. Bremer, D. Williams, C. Doutriaux, V. Pascucci, and C. R. Johnson. Ensemble-vis: A framework for the statistical visualization of ensemble data. In *2009 IEEE ICDM*, 2009.

[PWH11] Kai Pöthkow, Britta Weber, and Hans-Christian Hege. Probabilistic Marching Cubes. *Computer Graphics Forum*, 30(3):931–940, 2011.

[PWL97] A. T. Pang, C. M. Wittenbrink, and S. K. Lodha. Approaches to uncertainty visualization. *The Visual Computer*, (8):370–390, 1997.

[RPDB12] Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein barycenter and its application to texture mixing. In *Scale Space and Variational Methods in Computer Vision*,

pages 435–446, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[RT16] Vanessa Robins and Katharine Turner. Principal Component Analysis of Persistent Homology Rank Functions with case studies of Spatial Point Patterns, Sphere Packing and Colloids. *Physica D: Nonlinear Phenomena*, 334:99–117, 2016.

[RWS11] Vanessa Robins, Peter John Wood, and Adrian P. Sheppard. Theory and Algorithms for Constructing Discrete Morse Complexes from Grayscale Digital Images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(8):1646–1658, 2011.

[SC15] Vivien Seguy and Marco Cuturi. Principal Geodesic Analysis for Probability Measures under the Optimal Transport Metric. In *NeurIPS*, 2015.

[SDT23] Keanu Sisouk, Julie Delon, and Julien Tierny. Wasserstein Dictionaries of Persistence Diagrams. *CoRR*, 2023.

[She68] D. Shepard. A two-dimensioanl interpolation function for irregularly-spaced data. In *ACM National Conference*, 1968.

[Sin67] R. Sinkhorn. Diagonal equivalence to matrices with prescribed row and column sums. *American Mathematical Monthly*, 45(2):195–198, 1967.

[SKS12] S. Schlegel, N. Korn, and G. Scheuermann. On the interpolation of data with normally distributed uncertainty for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2305–2314, 2012.

[SMKN20] Raghavendra Sridharamurthy, Talha Bin Masood, Adhitya Kamakshidasan, and Vijay Natarajan. Edit Distance between Merge Trees. *IEEE Transactions on Visualization and Computer Graphics*, 26(3):1518–1531, 2020.

[SN12] Nithin Shivashankar and Vijay Natarajan. Parallel Computation of 3D Morse-Smale Complexes. *Computer Graphics Forum*, 31(3):965–974, 2012.

[Sou11] T. Sousbie. The Persistent Cosmic Web and its Filamentary Structure: Theory and Implementations. *Royal Astronomical Society*, 414:384–403, 2011.

[SPCT18] Maxime Soler, Melanie Plainchault, Bruno Conche, and Julien Tierny. Lifted Wasserstein matcher for fast and robust topology tracking. In *IEEE LDAV*, 2018.

[SPN+16] Nithin Shivashankar, Pratyush Pranav, Vijay Natarajan, Rien van de Weygaert, EG Patrick Bos, and Steven Rieder. Felix: A topology based framework for visual exploration of cosmic filaments. *IEEE Transactions on Visualization and Computer Graphics*, 22(6):1745–1759, 2016.

[SSW14] Himangshu Saikia, Hans-Peter Seidel, and Tino Weinkauf. Extended Branch Decomposition Graphs: Structural Comparison of Scalar Data. *Computer Graphics Forum*, 33(3):41–50, 2014.

[SZD+10] J. Sanyal, S. Zhang, J. Dyer, A. Mercer, P. Amburn, and R. Moorhead. Noodles: A tool for visualization of numerical weather model ensemble uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1421–1430, 2010.

[Szy13] A. Szymczak. Hierarchy of stable Morse decompositions. *IEEE Transactions on Visualization and Computer Graphics*, 19(5):799–810, 2013.

[TBGS18] Ilya O. Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schölkopf. Wasserstein auto-encoders. In *ICLR*, 2018.

[TCWN08] Russell Taylor, Amit Chourasia, Daniel Whalen, and Michael L. Norman. The IEEE SciVis Contest. http://sciviscontest.ieeevis.org/2008/, 2008.

[TFL+17] Julien Tierny, Guillaume Favelier, Joshua A. Levine, Charles Gueunet, and Michael Michaux. The Topology ToolKit. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):832–842, 2017. https://topology-tool-kit.github.io/.

[TGSP09] Julien Tierny, Attila Gyulassy, Eddie Simon, and Valerio Pascucci. Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1177–1184, 2009.

[Tie18] Julien Tierny. *Topological Data Analysis for Scientific Visualization*. Springer, 2018.

[TMMH14] Katharine Turner, Yuriy Mileyko, Sayan Mukherjee, and John Harer. Fréchet Means for Distributions of Persistence Diagrams. *Discrete Computational Geometry*, 52(1):44–70, 2014.

[TN13] Dilip Mathew Thomas and Vijay Natarajan. Detecting symmetry in scalar fields using augmented extremum graphs. *IEEE Transactions on Visualization and Computer Graphics*, 2013.

[TV98] S. Tarasov and M. Vyali. Construction of contour trees in 3D in O(n log n) steps. In *Symposium on Computational Geometry*, 1998.

[unc08] ISO/IEC Guide 98-3:2008 uncertainty of measurement - part 3: Guide to the expression of uncertainty in measurement (GUM). 2008.

[VBT20] Jules Vidal, Joseph Budin, and Julien Tierny. Progressive Wasserstein Barycenters of Persistence Diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):151–161, 2020.

[vdMH08] L.J. P. van der Maaten and G.E. Hinton. Visualizing Data Using t-SNE. *JMLR*, 9(86):2579–2605, 2008.

[WBKS04] Wei Wang, Cindy Bruyere, Bill Kuo, and Tim Scheitlin. The IEEE SciVis Contest. http://sciviscontest.ieeevis.org/2004/, 2004.

[WCG+17] Thomas Wischgoll, Amit Chourasia, Ksenia Gorges, Matthias Bruck, and Niklas Rober. The IEEE SciVis Contest. http://sciviscontest.ieeevis.org/2017/, 2017.

[WG22] Florian Wetzels and Christoph Garth. A Deformation-based Edit Distance for Merge Trees. In *TopoInVis*, 2022.

[WLG22] Florian Wetzels, Heike Leitte, and Christoph Garth. Branch Decomposition-Independent Edit Distances for Merge Trees. *Computer Graphics Forum*, 2022.

[WMK13] R. T. Whitaker, M. Mirzargar, and R. M. Kirby. Contour boxplots: A method for characterizing uncertainty in feature sets from simulation ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2713–2722, 2013.

[Woo14] David P. Woodruff. *Sketching as a Tool for Numerical Linear Algebra*. Now Publishers, 2014.

[WZ13] Keqin Wu and Song Zhang. A contour tree based visualization for exploring data with uncertainty. *IJUQ*, 2013.

[YMS+21] Lin Yan, Talha Bin Masood, Raghavendra Sridharamurthy, Farhan Rasheed, Vijay Natarajan, Ingrid Hotz, and Bei Wang. Scalar field comparison with topological descriptors: Properties and applications for scientific visualization. *Computer Graphics Forum*, 40(3):599–633, 2021.

[YWM+19a] Lin Yan, Yusu Wang, Elizabeth Munch, Ellen Gasparovic, and Bei Wang. Source Code for a Structural Average of Labeled Merge Trees for Uncertainty Visualization. https://github.com/tdavislab/amt, 2019.

[YWM+19b] Lin Yan, Yusu Wang, Elizabeth Munch, Ellen Gasparovic, and Bei Wang. A structural average of labeled merge trees for uncertainty visualization. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):832–842, 2019.

[Zha96] K. Zhang. A Constrained Edit Distance Between Unordered Labeled Trees. *Algorithmica*, 15(3):205–222, 1996.

[Zom10] Afra J. Zomorodian. Topology for computing. In Mikhail J. Atallah and Marina Blanton, editors, *Algorithms and Theory of Computation Handbook (Second Edition)*, chapter 3, pages 82–112. CRC Press, 2010.

[ZSS92] Kaizhong Zhang, Rick Statman, and Dennis Shasha. On the editing distance between unordered labeled trees. *Information Processing Letters*, 42(3):133–139, 1992.

Analysis of Ensembles of Topological Descriptors

Topological Data Analysis (TDA) forms a collection of tools to generically, robustly and efficiently reveal implicit structural patterns hidden in complex datasets. These tools allow to compute a topological representation for each member of an ensemble of datasets by encoding its main features of interest in a concise and informative manner. A major challenge consists then in designing analysis tools for such ensembles of topological descriptors. Several tools have been well studied for persistence diagrams, one of the most used descriptor. However, they suffer from a lack of specificity, which can yield identical data representations for significantly distinct datasets. In this thesis, we aimed at developing more advanced analysis tools for ensembles of topological descriptors, capable of tackling the lack of discriminability of persistence diagrams and going beyond what was already available for these objects. First, we adapt to merge trees, descriptors having a better specificity, the tools already available for persistence diagrams such as distances, geodesics and barycenters. Then, we want to go beyond this notion of average being the barycenter in order to study the variability within an ensemble of topological descriptors. We then adapt the Principal Component Analysis framework to persistence diagrams and merge trees, resulting in a dimensionality reduction method that indicates which structures in the ensemble are most responsible for the variability. However, this framework allows only to detect linear patterns of variability in the ensemble. To tackle this we propose to generalize this framework to Auto-Encoder in order to detect non-linear, i.e. more complex, patterns in an ensemble of merge trees or persistence diagrams. Specifically, we propose a new neural network layer capable of processing natively these objects. We present applications of all this work in feature tracking in a time-varying ensemble, data reduction to compress an ensemble of topological descriptors, clustering to form homogeneous groups in an ensemble, and dimensionality reduction to create a visual *map* indicating how the data are organized regarding each other in the ensemble.

Analyse d'Ensembles de Descripteurs Topologiques

L'analyse topologique de données forme un ensemble d'outils visant à révéler de manière générique, robuste et efficace les caractéristiques structurelles implicites cachées dans des ensembles de données complexes. Ces outils permettent de calculer une représentation topologique pour chaque membre d'un ensemble de données en encodant ses principales caractéristiques d'intérêt de manière concise et informative. Un défi majeur consiste ensuite à concevoir des outils d'analyse pour de tels ensembles de descripteurs topologiques. Plusieurs outils ont été bien étudiées pour les diagrammes de persistance, l'un des descripteurs les plus utilisés. Cependant, ils souffrent d'un manque de spécificité, pouvant donner des représentations de données identiques pour des données significativement différentes. Dans cette thèse, nous avons cherché à développer des outils d'analyse plus avancés pour des ensembles de descripteurs topologiques, capables de résoudre le problème de discriminabilité des diagrammes de persistance et d'aller au-delà de ce qui était déjà disponible pour ces objets. Tout d'abord nous adaptons aux arbres de fusion, descripteurs ayant une meilleur spécificité, les outils déjà disponibles pour les diagrammes de persistance tels que le calcul de distances, géodésiques et barycentres. Ensuite, nous souhaitons aller au-delà de cette simple notion de moyenne qu'est le barycentre pour étudier la variabilité au sein d'un ensemble de descripteurs topologiques. Nous adaptons alors le cadre de l'Analyse en Composantes Principales aux diagrammes de persistance et les arbres de fusion, résultant une méthode de réduction de dimensions qui indique quelles structures dans l'ensemble sont les plus responsables de la variabilité. Cependant, ce cadre permet uniquement de détecter des tendances linéaires de variabilité dans l'ensemble. Pour résoudre ce problème, nous proposons de généraliser ce cadre aux Auto-Encodeurs afin de détecter des motifs non linéaires, i.e. plus complexes, dans un ensembles d'arbres de fusions ou de diagrammes de persistance. Plus précisément, nous proposons une nouvelle couche de réseau de neurones capable de traiter nativement ces objets. Nous présentons des applications de ces travaux pour le suivi de structures dans un ensemble de données variant dans le temps pour la réduction de données pour compresser un ensemble de descripteurs topologiques, dans le partitionnement pour former des groupes homogènes dans un ensemble, et dans la réduction de dimensions pour créer une *carte* visuelle indiquant comment les données sont organisées les unes par rapport aux autres dans l'ensemble.