HAL
open science

Progressivity in topological data analysis

Jules Vidal

To cite this version:

Jules Vidal. Progressivity in topological data analysis. Computational Geometry [cs.CG]. Sorbonne Université, 2021. English. NNT : 2021SORUS398 . tel-04107109v1

HAL Id: tel-04107109

https://theses.hal.science/tel-04107109v1

Submitted on 14 Feb 2023 (v1), last revised 26 May 2023 (v2)

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

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

DOCTORAL THESIS
SORBONNE UNIVERSITÉ

Speciality: **COMPUTER SCIENCE**

Presented by

JULES VIDAL

to obtain the degree of

Ph.D. of Sorbonne Université

Progressivity in Topological Data Analysis

To be defended on December 8th, 2021, before the committee:

Michaël Aupetit	Senior Scientist	Qatar Computing Research Institute	**Reviewer**
Frédéric Chazal	Senior Scientist	INRIA	**Reviewer**
Isabelle Bloch	Professor	Sorbonne Université	**Examiner**
David Coeurjolly	Senior Scientist	CNRS	**Examiner**
Jean-Daniel Fekete	Senior Scientist	INRIA	**Examiner**
Gabriel Peyré	Senior Scientist	CNRS	**Examiner**
Vanessa Robins	Senior Scientist	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

Acknowledgments
Remerciements

I would like to warmly thank Michaël Aupetit and Frédéric Chazal for accepting to take the time to read and review this thesis. I greatly appreciated your feedback. Thank you also to the other examiners, Isabelle Bloch, David Coeurjolly, Jean-Daniel Fekete, Gabriel Peyré and Vanessa Robins. I am honored that you accepted to be part of my defense committee and to become acquainted with my work.

Un grand merci à toi Julien, d'avoir été pour moi un professeur et un encadrant de thèse exceptionnels. Merci pour tes conseils, pour tout ce que tu m'as appris, et surtout pour ta bienveillance constante. Ta motivation et ta bonne humeur sont contagieuses, et j'aurais eu du mal à arriver jusqu'ici sans l'une et l'autre.

Merci aux (ex-)membres de l'équipe APR et du LIP6 pour ces chouettes années passées ensemble. Merci en particulier Guillaume F., Charles, Maxime, Matthieu J., Marwan, David, Alice, Steven, Clément, Matthieu D., Boubacar, pour votre exceptionnelle gentillesse, l'accueil chaleureux que vous m'avez offert, et tous vos conseils. Merci Ghiles, Vincent et Frédéric pour m'avoir accueilli dans votre gang à roulettes. Merci Martin, Raphaël et Yi-Ting, c'était un vrai plaisir de traverser cette expérience à vos côtés de bout en bout. Mention spéciale à Pierre, un grand merci pour toute l'aide que tu m'as apportée, m'avoir appris à utiliser `git` (ou au moins essayé) et les nombreuses séries auxquelles tu m'as initié. Un grand merci aussi Mat(t)ieu, Hugo, Keanu, Francesco, Ada, Guillaume B., et Eve, pour les moments toujours agréables passés ensemble. Vous formez une belle équipe !

A big thanks to the members of the VESTEC project, who were all really welcoming and really kind. It has been a great and maturing experience working with you.

Merci à Mélanie, Fabien, Jonathan, et aux autres membres de la communauté visu que j'ai rencontrés en conférence, et toujours recroisés avec un grand plaisir.

À tous mes ami·e·s qui ont été présent·e·s à mes côtés, m'ont encouragé et soutenu, m'ont souvent écouté et ont parfois enduré les moments difficiles avec moi, un immense merci. J'ai de la chance de vous avoir autour de moi, et la chance que vous soyiez assez nombreux et nombreuses pour que j'hésite franchement à vous nommer toustes. Mais puisque je vous dois bien ça: merci Iris, Romain, Mathieu, Matthieu, Juliette, Olivier, Jérémy, Cyrielle, Raphaël, Antoine, Alex, Arno, Roulia, Laura, Pierre E., Sofia, Franz, Ilham, Gwendal, Claire, Pierre B., Agathe, Estelle, Magali, Thomas, Glen, Fanny, Timothée, Mark, et Rémy. Sans ordre particulier. Un grand merci aussi à toi, Naomi, d'être là avec moi.

Enfin, merci à ma mère Catherine, mon père Frédéric, ma sœur Rose, ma sœur Eva, mon frère Sam, Marie, mes grand-parents, et le reste de ma famille, qui ont partagé mes réussites et mes tracas, et qui se sont efforcé·e·s de comprendre et d'apprécier mon travail. Votre soutien a beaucoup compté.

Publications

First-author publications

Journal Papers

A Progressive Approach to Scalar Field Topology
Jules Vidal, Pierre Guillou, Julien Tierny
IEEE Transactions on Visualization and Computer Graphics, 2021
To be presented at IEEE VIS 2021

Progressive Wasserstein Barycenters of Persistence Diagrams
Jules Vidal, Joseph Budin, Julien Tierny
IEEE Transactions on Visualization and Computer Graphics
Proc. of IEEE VIS 2019
Best Paper Honorable Mention Award

Conference Papers

Fast Approximations of Persistence Diagrams with Guarantees
Jules Vidal and Julien Tierny
Proc. of IEEE Symposium on Large Data Analysis and Visualization (LDAV) 2021

Other Publications

Journal Papers

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

Conference Papers

Statistical Parameter Selection for Clustering Persistence Diagrams
Max Kontak, Jules Vidal, Julien Tierny
Proc. of Super Computing workshop on Urgent HPC 2019

An Overview of the Topology Toolkit
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 A. Levine, Jonas Lukasczyk, Daisuke Sakurai, Maxime Soler, Peter Steneteg, Julien Tierny, Will Usher, Jules Vidal and Michal Wozniak
Proc. of TopoInVis 2019

Tutorials

Topological Analysis of Ensemble Scalar Data with TTK
Christoph Garth, Charles Gueunet, Pierre Guillou, Lutz Hofmann, Joshua A Levine, Jonas Lukasczyk, Julien Tierny, Jules Vidal, Bei Wang, Florian Wetzels
IEEE VIS Tutorials 2021

Topological Data Analysis Made Easy with the Topology ToolKit, What is New?
Martin Falk, Christoph Garth, Charles Gueunet, Pierre Guillou, Attila Gyulassy, Lutz Hofmann, Christopher Kappe, Joshua A Levine, Jonas Lukasczyk, Julien Tierny, Jules Vidal
IEEE VIS Tutorials 2020

Topological Data Analysis Made Easy with the Topology ToolKit, A Sequel
Martin Falk, Christoph Garth, Charles Gueunet, Joshua A Levine, Jonas Lukasczyk, Julien Tierny, Jules Vidal
IEEE VIS Tutorials 2019

Contents

Acknowledgments – Remerciements iii

Publications v

Contents vii

Notations xi

1 Introduction 1

1.1 General Context 2

1.1.1 Data Analysis and Visualization 2

1.1.2 The Topology ToolKit (TTK) 3

1.1.3 The VESTEC Project 4

1.2 Motivations 4

1.2.1 Topological data representation 5

1.2.2 Progressive computation 5

1.3 Problem Formulation 7

1.3.1 Data Reduction 7

1.3.2 Analysis of Reduced representations 7

1.3.3 Degraded/progressive computation 8

1.4 Contributions 8

1.5 Outline 10

2 Theoretical Background 11

2.1 Input data representation 13

2.1.1 Domain representation 13

2.1.2 Topological invariants 16

2.1.3 Scalar field representation 18

2.2 Critical Points 21

2.3 Persistence Diagrams 23

2.3.1 The case of the **0**-th Betti number 23

2.3.2 Notions of persistent homology 24

2.3.3 Topological persistence 26

2.4 Metrics on the space of Persistence Diagrams 27
2.4.1 Wasserstein distance between diagrams 28
2.4.2 Bottleneck distance between diagrams 30
2.5 Other topological abstractions 30

3 A Progressive Approach to Scalar Field Topology 33
Our contribution in one image 35
3.1 Context 36
3.1.1 Related Work 36
3.1.2 Contributions 37
3.2 Progressive Data Representation 38
3.2.1 Edge-Nested Triangulation Hierarchy 38
3.2.2 Edge-Nested Triangulations of Regular Grids 40
3.2.3 Topologically Invariant Vertices 42
3.3 Progressive Critical Points 46
3.3.1 Initialization and Updates 47
3.3.2 Computation Shortcuts 49
3.3.3 Parallelism 49
3.3.4 Extremum Lifetime 49
3.4 Progressive Persistence Diagrams 50
3.4.1 Persistence Diagram from Critical Points 52
3.4.2 Progressive Strategy 54
3.4.3 Parallelism 54
3.5 Results 55
3.5.1 Progressive Data Representation 55
3.5.2 Time Performance 58
3.5.3 Stress Cases 61
3.5.4 Progressive Topological Visualization and Analysis 62
3.6 Limitations and Discussion 65
3.7 Summary 66

4 Approximation of Persistence Diagrams with Guarantees 69
Our contribution in one image 71
4.1 Overview 72
4.2 Topology Approximation 72
4.2.1 Hierarchy Processing 73
4.2.2 Vertex Folding 74
4.2.3 Bottleneck Error Control 76

4.2.4 Monotony offsets 77
4.2.5 Parallelism 79
4.2.6 Uncertainty 79
4.3 Results 80
4.3.1 Time Performance 80
4.3.2 Approximation Accuracy 83
4.3.3 Qualitative Analysis 85
4.4 Limitations and Discussion 87
4.5 Summary 88

5 Progressive Wasserstein Barycenters of Persistence Diagrams **89**
Our contribution in one image 93
5.1 Context 94
5.1.1 Related Work 94
5.1.2 Contributions 97
5.2 Background 98
5.2.1 Efficient Wasserstein distance computation by Auction . . 98
5.2.2 Wasserstein barycenters of Persistence diagrams 100
5.3 Overview 101
5.4 Progressive Barycenters 101
5.4.1 Auctions with Price Memorization 101
5.4.2 Accuracy-driven progressivity 102
5.4.3 Persistence-driven progressivity 103
5.4.4 Parallelism 104
5.4.5 Computation time constraints 104
5.5 Application to Ensemble Topological Clustering 107
5.6 Results 108
5.6.1 Time performance 109
5.6.2 Barycenter quality 111
5.6.3 Ensemble visual analysis with Topological Clustering . . 114
5.7 Limitations 116
5.8 Overall time-constrained pipeline 118
5.9 Summary 119

6 An application use case **121**
6.1 The VESTEC project 123
6.1.1 Purpose 123
6.1.2 Numerical simulations for urgent decision-making 124

6.1.3 Use cases 125
6.1.4 Challenges 128
6.2 Topological Data Analysis in VESTEC 129
6.2.1 Persistence diagrams for Data Reduction 129
6.2.2 In-Situ Computation 130
6.2.3 Statistical Analysis 131
6.3 Results: The Space Weather Use Case 131
6.3.1 *In-Situ* Computation of Persistence Diagrams 131
6.3.2 Wasserstein distances between diagrams 132
6.3.3 Topological Clustering 133
6.4 Conclusion 135

7 Conclusion 137
7.1 Summary of contributions 137
7.2 Discussion 139
7.3 Perspectives 141
7.4 Final Word 142

A Appendix: Data List 143

Bibliography 147

Notations

$\mathscr{M}$	Manifold
$\mathcal{G}$	Regular grid
$\mathcal{M}$	Piecewise linear manifold
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^n$	Euclidean space of dimension n
σ	Simplex
$\mathcal{M}_i$	Set of i-simplices of $\mathcal{M}$
τ	Face of a simplex
v	Vertex
$St(\sigma), \quad Lk(\sigma)$	Star and link of a simplex σ
$Lk^-(\sigma), \quad Lk^+(\sigma)$	Lower link and upper link of a simplex σ
$f : \mathcal{M} \to \mathbb{R}$	Piecewise linear scalar field
w	Isovalue
$f^{-1}(w)$	Level set
$f^{-1}_{-\infty}(w)$	Sub-level set
$\mathcal{D}(f)$	Persistence diagram of f
W_2	L_2-Wasserstein distance
W_∞	Bottleneck distance
$\mathcal{D}^*$	Wasserstein barycenter of a set of persistence diagrams
$\mathcal{H}$	Edge-nested triangulation hierarchy
$\mathcal{M}^i$	i-th level of $\mathcal{H}$
o	An *old* vertex in $\mathcal{M}^i$
n	A *new* vertex in $\mathcal{M}^i$
$Lk(v)^i, \quad Lk^+(v)^i, \quad Lk^-(v)^i$	Link, upper link and lower link of v in $\mathcal{M}^i$
$\widehat{v}$	*Folded* vertex
$\widehat{f}$	Approximation of f by vertex folding
$f^{\hookleftarrow}$	Staircase approximation of f
$\mathbb{M}$	Monotony offsets
$\mathbb{O}$	Offsets

1 Introduction

Data is an intangible concept that has taken an increasing place in our everyday life. A general idea is that data usually contains *information*, in a raw state. In contrast to information, however, data needs to be scrutinized, analyzed, and interpreted in order to be given meaning. Etymologically, the word *data* comes from Latin *data*, plural of *datum*, « that is given ». It represents a premise, a given or granted fact upon which a reasoning or a calculation can be built. This thesis deals with data analysis and visualization, in particular for the interpretation of scientific data, i.e. relating to the fields of natural sciences and engineering. We seek ways to extract and visualize information from the raw data, so as to ease its interpretation and offer more understanding of a phenomenon. In particular, our work focuses on topological methods for data analysis and visualization. This recent field of research relies on the idea that data has a structure, an intrinsic shape that characterizes the phenomenon that it describes. Extracting this shape is of great interest to understand what is happening behind the data. A great analogy[1] can be made with maps. A raw photography of a aerial view of a city is a good description of the real world. However its use is limited to localize and guide oneself. In contrast, a map encoding the main structures of the data (main roads, main buildings) would be of greater help. Once this new representation is created, it can further be used to compute even more interesting information, such as optimal itineraries and travel times. Topological Data Analysis enables the same thing with more general data, it helps to encode the main features into simpler, practical reduced representations.

[1] I first heard this analogy from Attila Gyulassi, at the occasion of the first Topology ToolKit tutorial at IEEE VIS 2018 (my first ever attendance at a Visualization conference).

1.1 General Context

1.1.1 Data Analysis and Visualization

In modern sciences, data is historically present in the form of measurement results and observations. They are produced with the help of measuring tools: thermometers, telescopes, manometers, and later more advanced sensors such as an electronic microscope or an MRI scanner, for instance. Natural sciences moved forward with the development of new scientific instruments, which continuously produced data. The analysis of such data has been a key component of the hypothetico-deductive method, broadly used in natural sciences, which consists in formulating hypotheses, later tested against observable data. Data is also the support for inductive reasoning in sciences, which consists in the formulation of a hypothesis based on the analysis of data. A famous example of a so-called *data-driven* scientific discovery was the enunciation by Kepler of his third law of planetary motion in 1619 [Kep19], from the analysis of astronomical data gathered since the Antiquity, notably by the astronomers Ptolemy and Tycho Brahe. The kind of data aforementioned, produced by observational and measuring instruments, is said to be *acquired data*.

In contrast, scientific data can also be the result of calculations realized with a model. The field of numerical simulations is concerned with the solving of these calculations with a computer, in order to simulate the unfolding of a real phenomenon. Simulations are broadly used in all scientific domains to analyze the implications of a model. They are also used in scientific applications to try and predict the reaction of a system in specific conditions. Some examples of applications include weather forecasting, molecular dynamics simulations to estimate the efficiency of a medicinal drug, or crash test simulations in the car industry to avoid the cost and hassle of performing real ones. In practice, large *ensembles* of simulations are run in order to explore the influence of a large variety of input parameters for a model and quantify the related uncertainties. The colossal progress made in the last decades in computing power as well as the development of advanced numerical methods have opened the gate to an unprecedented production of data through numerical simulations. Figure 1.1 gives examples of both acquired and simulated scientific data.

Technological progress induces a continuous increase in the size and complexity of data produced by acquisition and simulation methods, that has to be interpreted. The analysis of data is a key component of the

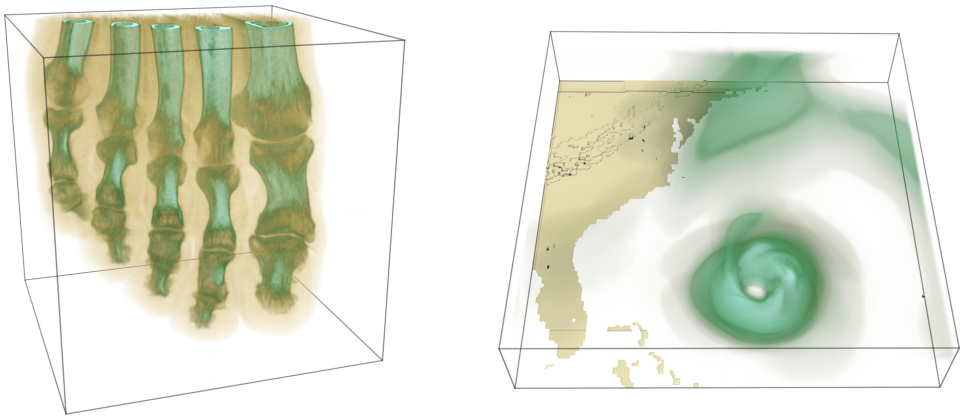


Figure 1.1 – *Examples of scientific data. On the left, an* acquired *data set: a Computerized Tomography scan (CT scan) of a foot, the density of the foot is represented. On the right, a* simulated *data set: the result of the numerical simulation of the Isabel hurricane, that hit the coast of Florida in 2003. The magnitude of the wind velocity is represented. The data is visualized using a colormap: green and cyan denote higher values of the data, while beige and brown correspond to lower values.*

interpretation process. Scientific visualization, in particular, deals with the graphical representation of raw scientific data with the aim of gaining insights about the data and facilitating its interpretation, for instance by visually highlighting trends that might be overlooked otherwise. Scientific visualization combines data analysis and computer graphics, and is concerned with the interactivity, performance and interpretability of the produced visualizations.

1.1.2 The Topology ToolKit (TTK)

The Topology ToolKit (TTK, https://topology-tool-kit.github.io) [TFL+17, BMBF+19] is a software platform designed for the topological analysis of scalar data for scientific visualization. It is developed in C++ and tightly integrated in ParaView [AGL05], which makes it easily accessible to end users, as ParaView is a widely-used, open-source visualization software that is developed around the Visualization ToolKit (VTK) library. TTK aims at providing efficient and robust implementations of the standard tools of Topological Data Analysis (introduced in Section 1.2.1). Its modularity also facilitates the development of new analysis tools, while leveraging the visualization ecosystem provided by VTK and ParaView. As such it is also a versatile research tool. TTK is an open-source software, it is mainly developed at Sorbonne Université under the direction of Julien Tierny, but has over 30 different contributors [TTK20] from over 16 academic and industrial institutions. TTK is a distributed software and re-

cently became an official plugin of ParaView (as of version 5.10), which increased its accessibility.

All the research work that is presented in this thesis was developed and implemented in TTK, and officially integrated in the library upon publication. Therefore it is publicly available, and accessible.

1.1.3 The VESTEC Project

This thesis takes place in the context of a European project called VESTEC (Visual Exploration and Sampling Toolkit for Extreme Computing). The purpose of the project is to develop and build a flexible tool chain to support decision makers in the event of catastrophic situations (such as wildfires, earthquakes, disease outbreaks, ...), thanks to interactive supercomputing and interactive data analysis and visualization. The project regrouped partners both in academics and industrial fields, with expertise in various domains: high-performance computing, data analysis and visualization, numerical simulation and specific application fields. The work on data analysis and visualization that was realized during this thesis was integrated into the project, in interaction with its other parts such as high performance computing environments and immersive visualization applications. As such, this gave us the opportunity to work in the context of the development of a complete pipeline of simulation and data analysis and visualization, to experiment on real data and to exchange with experts in different related fields.

The extent of the project and the obtained results are described in detail in Chapter 6 as an application use case of our work.

1.2 Motivations

The growing power of acquisition tools and computer resources induces a never-ending increase in the size and complexity of the data produced in engineering and sciences, which constitutes a challenge for their analysis and interpretation. In order to address these issues, advanced data analysis tools are designed to efficiently capture the main features of interest in large data sets, in order to support interactive visualization and analysis tasks.

1.2.1 Topological data representation

Topological Data Analysis (TDA) is a collection of techniques born from the application of the concepts of computational topology to data analysis. Briefly presented, topology describes geometrical objects based on properties that are invariant through continuous deformations. Principles such as connectedness, number of handles or voids, orientability, are used to characterize objects independently of their geometrical shape. As such, the tools developed in Topological Data Analysis (TDA) enable a generic, robust, and efficient extraction of structural features in data [EH09]. Over the last years, many data analysis and visualization methods have been built around these concepts [HLH+16], with applications to a large spectrum of domains, including astrophysics [Sou11, SPN+16], biological imaging [CSvdP04, BDSS18, AAPW18], chemistry [BGL+18, GABCG+14, OGT19], fluid dynamics [KRHH11], material sciences [GDN+07, GKL+15, SPD+19], or turbulent combustion [LBM+06, BWT+11, GBG+14]. In the case of scalar data, TDA algorithms rely on established topological data abstractions, such as contour trees [BR63, DBvK97, TV98, CSA00, SM17, GFJT19], Reeb graphs [Ree46, BGSF08, PSBM07] or Morse-Smale complexes [DFFIM15, GBHP08, RWS11, GBP18]. In particular, the Persistence diagram [ELZ02] is a concise data representation, which visually summarizes the population of features of interest in a data set, as a function of a measure of importance called *topological persistence*. Its conciseness, combined with its stability, made it increasingly popular in machine learning [CCO17, RHBK15, RSL17] and in interactive data analysis, where it quickly provides visual hints regarding the number and importance of the features in the data.

Although the core algorithms in TDA have practicable asymptotic complexities (usually from linear to quadratic time), the construction of the above topological abstractions can still require significant computation times for real-life data sets. Thus, when they are integrated into larger interactive systems, TDA algorithms can become a serious time bottleneck. This is a concern in data exploration scenarios, where users may need to wait between seconds and minutes to get a feedback when they adjust the parameters of the topological analysis.

1.2.2 Progressive computation

In his seminal paper on response times of interactive systems, Miller [Mil68] studied the impact of response time on the ability of users to

maintain focus on a given task. For response times below a second (*continuity preserving latency*), the system appears fully responsive to user adjustments, allowing truly interactive sessions. For response times below a few seconds (*flow preserving latency*), users still manage to maintain their focus but the pauses in the exploration, due to the computation, challenge user interpretation skills. Above ten seconds (*attention preserving latency* threshold), users tend to disengage from the task to pursue other activities in parallel, which is highly detrimental to the interpretation process. Unfortunately, for real-life data, existing TDA algorithms correspond to the latter category of response times.

To address the discontinuities in user experience implied by excessive computation times, the notion of *progressive* data analysis has been explored by several authors in information visualization [WM04, FP16, ZGC+17, JSF20]. In this context, a progressive algorithm is a technique capable of providing an interpretable output upon interruption requests, and of refining it otherwise, progressively converging towards the final solution. In an interactive setup, progressive algorithms can improve user experience in two ways. First, users can let such algorithms refine progressively their outputs, while receiving continuously visual feedbacks, and stop them when the outputs are deemed satisfactory. Second, users can also define a priori an upper limit on the computation time, after which the computation is interrupted. This enables to design interactive systems with guaranteed response time. This is particularly useful for algorithms whose actual execution times are difficult to anticipate, such as in TDA, where many popular algorithms, although with known time complexity, may have an output-sensitive computation time in practice. Progressive algorithms can also be beneficial to non-interactive setups, such as batch-mode computations on high performance computers, where the allocation of computing resources often needs to be finely controlled. In this context, progressivity enables the assignment of precise computation time budget to data analysis programs. This is relevant for time-critical applications such as numerical simulation in support of urgent decision making in crisis management (wildfires, floods, outbreaks, etc), *i.e* the context of the VESTEC project. This is also relevant to other applications, towards the control of the power consumption of high performance computers, which becomes an increasingly important societal issue.

1.3 Problem Formulation

In this thesis, we try and tackle the issue of dealing with the analysis and visualization of large scalar data. Our goal is to provide generic and efficient methods to describe and analyze data, in order to incorporate them into larger analysis pipeline such as the one developed in the scope of the VESTEC project. In this context, we turn to Topological Data Analysis to first perform data reduction through the generic extraction of topological features of scalar data and their encoding into efficient reduced data structures. In a second step, we try and develop statistical tools to provide a coherent analysis of the reduced data. The third axis of our work resides in the application of the principles of progressivity to the whole approach.

1.3.1 Data Reduction

The persistence diagram [ELZ02] constitutes our topological abstraction of choice for the concise representation of important features in the data. The persistence diagram is formally introduced in Section 2.3. It is one of the most simple and most studied representations of the repartition of topological features of a scalar field. The diagram encodes the *topological persistence* of features created and destroyed at the location of critical points in the data, and displays them as bars in a 2D plot, which makes it a very light and concise representation. Additionally, it is robust to noise, as seen in Section 2.3. More specifically, we focus in our work on the saddle-extrema persistence diagram, as the relevance of the topological features described by saddle-saddle persistence pairs is usually of moderate interest in our applications. Well-defined metrics (Section 2.4) have been proposed to describe the space of persistence diagrams, and extensively studied, which further motivates the choice of this topological abstraction for our work.

1.3.2 Analysis of Reduced representations

The important size of modern datasets challenges the efficiency of traditional statistical data analysis techniques, such as a clustering approach using a simple pointwise L_2-metric. Large ensemble of data sets are indeed usually too large to fit in memory at once, which motivates the use of data reduction techniques. Additionally, an analysis performed on adequately reduced data is often more informative, a consequence of the *curse of dimensionality* [Bel66]. Once a scalar field is represented by a persistence

diagram, further analysis can be run directly on the diagram. Following the idea of analyzing large quantities of data, we focused on the analysis of *ensembles* of scalar fields using their topology. As the data is reduced through the computation of a persistence diagram for each scalar field, the problem now resides in the analysis and visualization of an ensemble of persistence diagrams.

In this context, we investigated ways to efficiently estimate a diagram that is *representative* of the set, as such a representative diagram could visually convey the global trends in the ensemble in terms of features of interest. To address this issue, a promising idea consists in considering the *barycenter* of a set of diagrams, given a distance metric between them, such as the so-called Wasserstein metric presented in Chapter 2. Once distances and barycenters can efficiently be computed in the space of persistence diagrams, it is only a short step to the computation of a clustering of an ensemble of persistence diagrams. The clustering of ensemble members based on their topology is an operation of great interest for the purpose of performing trend analysis in a large ensemble of simulation results.

1.3.3 Degraded/progressive computation

In this work we investigate the two axes listed above, data reduction for one part and analysis of the reduced representations for the other, in the light of degraded and progressive computation. As mentioned before, our goal is to develop techniques that are able to provide an *approximate* but *exploitable* result within reduced computation times. This kind of degraded computation favors an interactive exploration of the data, and is useful to save computing resources such as time or power. When it is possible, we seek to develop methods that progressively refine approximate results towards an exact result. This kind of progressive computation enables continuous feedback along the way, and is by nature interruptible and resumable. It enables the production of approximate results within a constraint of computation time or precision on the result, that can be refined upon user requests.

1.4 Contributions

In this thesis, we present the following contributions to the development of degraded and progressive computation techniques in Topological Data Analysis:

Progressive Topological Data Reduction

We introduce a progressive approach for the topological analysis of scalar data. Our method relies on a hierarchical representation of the input data and the new notion of *topological invariant vertices*. This representation enables the development of new progressive algorithms for the extraction of critical points and the computation of the saddle-extremum persistence diagram of a scalar field. Our method computes results in a *coarse-to-fine* manner, progressively refining them. Thanks to efficient update mechanisms, our algorithms are overall even faster that non-progressive reference approaches. Upon interruption, they provide interpretable results that are similar to the exact, final outputs and which empirically quickly converge towards them.

Degraded Persistence Diagram Computation

We introduce a novel algorithm for the specific problem of computing an estimation of the persistence diagram of a scalar field, with guarantees on the error committed. Although not progressive, our method allows the fast and accurate approximation of a diagram, within a user-controlled Bottleneck distance to the exact result. Our approach provides a scalar field that corresponds to the approximation of the diagram, and we demonstrate ways to visualize the uncertainty related to the approximation directly in the diagrams.

Progressive Analysis of Reduced Data

On the problem of computing a persistence diagram that is representative of an ensemble of input diagrams, we introduce a novel algorithm for the progressive computation of Wasserstein barycenters of persistence diagrams. We revisit existing algorithms for the efficient computation of distances between diagrams to extend previous work by Turner et al. [TMMH14] on the estimation of barycenters. Our progressive approach produces barycenters that are explicit diagrams and are representative of the repartition of features in the ensemble. The approach provides an order of magnitude speedup over the fastest combination of existing methods, and can be interrupted to produce barycenters accounting for the main features of the data within interactive times. Additionally, we revisit the *k-means* algorithm and extend our strategy to introduce an interrupt-

ible clustering algorithm for an ensemble of persistence diagrams, with applications to the visual analysis of global feature trends in the data.

1.5 Outline

The rest of this manuscript is organized as follows:

- In Chapter 2, we present theoretical prerequisites on Topological Data Analysis and a general review of the state-of-the-art.
- Chapter 3 describes our progressive approach for the topological analysis of scalar data, with applications to the progressive extraction of critical points and the progressive persistence diagram computation.
- Chapter 4 presents our method for the fast approximation of persistence diagrams with guarantees on the resulting bottleneck error.
- Chapter 5 describes our approach for the progressive computation of Wasserstein barycenters of ensembles of persistence diagrams, as well as its extension to a progressive clustering algorithm of persistence diagrams.
- Chapter 6 details an application use case of our work. We describe in detail the VESTEC project in which this thesis takes place, and show how our work was integrated in the project, as well as preliminary results on a specific application use case from the project.
- Finally, we summarize in Chapter 7 the contributions of this thesis and elaborate on current limitations and open perspectives.

Theoretical Background

2

Contents

2.1 Input data representation 13
2.1.1 Domain representation 13
2.1.2 Topological invariants 16
2.1.3 Scalar field representation 18
2.2 Critical Points 21
2.3 Persistence Diagrams 23
2.3.1 The case of the **0**-th Betti number 23
2.3.2 Notions of persistent homology 24
2.3.3 Topological persistence 26
2.4 Metrics on the space of Persistence Diagrams 27
2.4.1 Wasserstein distance between diagrams 28
2.4.2 Bottleneck distance between diagrams 30
2.5 Other topological abstractions 30

This chapter presents theoretical preliminaries on topology and Topological Data Analysis, which constitutes the basis of our work. We formalize the representation of the input data that will be used in the remainder of our work, namely the notion of piecewise linear scalar field defined on piecewise linear manifolds. We also introduce useful topological representations of the features of a scalar field, such as its *critical points* and its *persistence diagram*. In order to define these objects, we summarize some concepts from computational topology, notably the notions of *homology* and *persistent homology*.

We tried in this chapter to provide a sound but concise basis to our work, and to give some intuition about the objects that we will handle

in this thesis. This chapter contains definitions adapted from [Tie18] and [EH09]. We refer to the textbook by Edelsbrunner and Harer for a formal and detailed introduction to computational topology [EH09].

2.1 Input data representation

This section formalizes the notion of *data* in our framework, that constitutes the input for the data analysis and visualization process. In a word, we consider a piecewise linear (PL) *scalar* field $f : \mathcal{M} \to \mathbb{R}$ defined on a PL d-manifold $\mathcal{M}$, with d equals 2 or 3 in our applications. The purpose of this section is to define these terms and introduce considerations about the topology of these objects. It contains a substantial amount of formal definitions, that are needed in our opinion to rigorously present the objects that we will handle in the remainder of this work.

2.1.1 Domain representation

We introduce hereafter the notion of piecewise linear manifold, that constitutes the domain on which the input scalar data is defined. We start by presenting the general notion of *topological space* and *homeomorphisms*, which is instrumental to the definition of manifolds.

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

- $\emptyset$ and $\mathbb{X}$ are in T;
- Any union of elements of T is in T;
- Any *finite* intersection of elements of T is in T.

T is then said to be a *topology* on $\mathbb{X}$. The elements of T are called the *open sets* of $\mathbb{X}$.

Definition 2.2 (*Homeomorphism*) A *homeomorphism* between two topological spaces is a bijective continuous function whose inverse function is continuous. These spaces are then said to be homeomorphic.

Definition 2.3 (*Manifold*) A topological space $\mathscr{M}$ is a d-manifold (without boundary) if every point of $\mathscr{M}$ has an open neighborhood that is homeomorphic to an open Euclidean ball of dimension d.

This definition intuitively means that manifolds are shapes in d dimensions that can locally be assimilated to a part of an Euclidean space $\mathbb{R}^d$ of the same dimension d, although their global structure might be more complex. For instance, a sphere locally looks like a 2-dimensional plane.

Manifolds with boundaries can be defined by authorizing *boundary points*, i.e. points which have a neighborhood that is homeomorphic to a

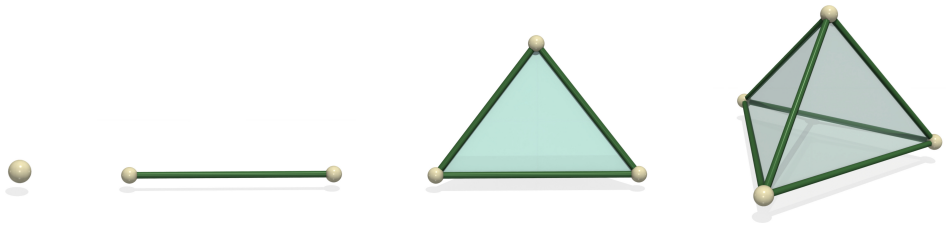


Figure 2.1 – *Simplices in low dimension ($d \leq 3$), with their faces. From left to right, a vertex, an edge, a triangle and a tetrahedron.*

Euclidean half-ball. The boundary of a manifold $\mathscr{M}$, noted $\partial\mathscr{M}$, is then defined as the set of boundary points. For instance, the boundary of the solid 3-dimensional ball is a sphere, which in return has an empty boundary. In the following, we only consider manifolds in low dimensions, with $d \leq 3$.

Definition 2.4 (*Convexity*) A set $\mathbb{C}$ of an Euclidean space $\mathbb{R}^n$ is *convex* if, for any two points p_0, p_1 of $\mathbb{C}$, the point $tp_0 + (1-t)p_1$ is also in $\mathbb{C}$, for all $t \in [0,1]$. The *convex hull* of a set $\mathcal{P}$ of points of an Euclidean space $\mathbb{R}^n$ is then the unique minimal convex set containing all points of $\mathcal{P}$, *e.g.* the intersection of all convex sets containing all points of $\mathcal{P}$.

Definition 2.5 (*Simplex*) A d-simplex is the convex hull of $d+1$ affinely independent points of an Euclidean space $\mathbb{R}^n$, with $0 \leq d \leq n$. d denotes the dimension of the simplex. In our applications, with $d \leq 3$, the simplices are classified in the following (Figure 2.1):

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

A d-simplex can be seen as the simplest combinatorial brick of dimension d that can be used to represent a d-dimensional domain in $\mathbb{R}^n$ with $d \leq n$.

Definition 2.6 (*Face*) A *face* τ of a d-simplex σ is a simplex that is defined by a non-empty, strict subset of the vertices of σ. The simplex σ is said to be a *co-face* of τ. We note $\tau < \sigma$.

As an example, a tetrahedron has fourteen faces: four triangles, six edges and four vertices, as seen in Figure 2.1.

Definition 2.7 (*Simplicial Complex*) A simplicial complex $\mathcal{K}$ of dimension d is a finite set of i-simplices, with $i \leq d$ such that:

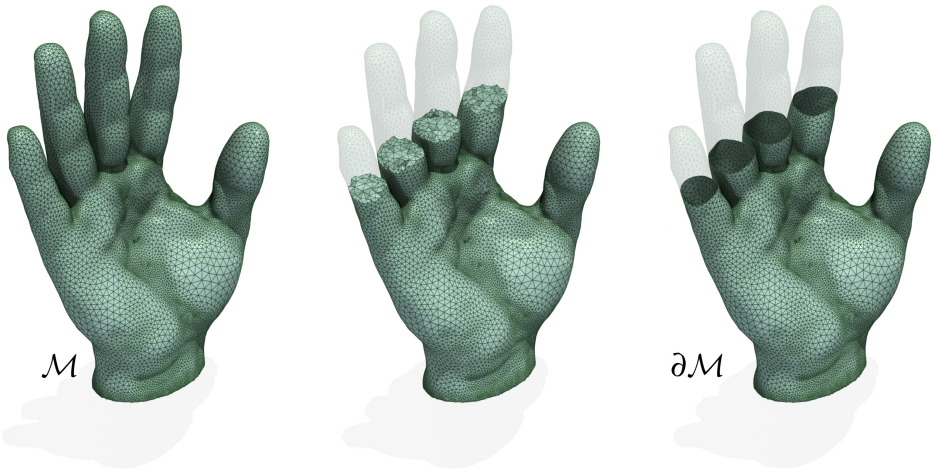


Figure 2.2 – *Example of a piecewise linear (PL) 3-manifold $\mathcal{M}$, i.e. a triangulation of dimension $d = 3$ (left). A clip view (center) reveals the solid interior of $\mathcal{M}$, constituted of tetrahedra. The boundary $\partial\mathcal{M}$ of $\mathcal{M}$ (right), is a PL 2-manifold without boundary. The clip view reveals that it is "hollow", constituted only of triangles.*

- for all $\sigma \in \mathcal{K}$, each face of σ is also in $\mathcal{K}$;
- for all $\sigma_0, \sigma_1 \in \mathcal{K}$, the intersection $\sigma_0 \cap \sigma_1$ is either empty or a common face of σ_0 and σ_1.

Definition 2.8 (*Star*) Given a simplicial complex $\mathcal{K}$, the *star* of a simplex σ of $\mathcal{K}$, noted $St(\sigma)$, is the set of the co-faces of σ that belong to $\mathcal{K}$:

$$St(\sigma) = \{\sigma' \in \mathcal{K} \mid \sigma < \sigma'\}$$

Intuitively, the star of a simplex σ is a small combinatorial neighborhood including σ.

Definition 2.9 (*Link*) Given a simplicial complex $\mathcal{K}$, the *link* of a simplex σ of $\mathcal{K}$, noted $Lk(\sigma)$, is the set of faces of the simplices of $St(\sigma)$ that have an empty intersection with σ:

$$Lk(\sigma) = \{\tau \in \mathcal{K} \mid \tau < \sigma',\ \sigma' \in St(\sigma),\ \tau \cap \sigma = \emptyset\}$$

Intuitively, the link of a simplex σ is the boundary of a small combinatorial neighborhood including σ.

Definition 2.10 (*Triangulation*) Let $\mathbb{X}$ be a topological space and $\mathcal{K}$ a simplicial complex. If the union of all simplices of $\mathcal{K}$ is homeomorphic to $\mathbb{X}$, $\mathcal{K}$ is called a triangulation of $\mathbb{X}$, and noted $\mathcal{T}$.

Definition 2.11 (*Piecewise Linear Manifold*) The triangulation of a manifold $\mathscr{M}$ is called a piecewise linear manifold, noted $\mathcal{M}$.

Therefore, a PL manifold (illustrated in Figure 2.2) is a topologically accurate representation of a manifold using a triangulation, *i.e.* an arrangement of combinatorial bricks called simplices. A PL manifold can be efficiently represented in memory through the list and position of its vertices, and the list of the high-dimensional simplices (triangles in 2d, tetrahedra in 3d) defined using these vertices.

In this thesis, a special emphasis is given to PL-manifolds generated from regular grids. A 2-dimensional regular grid is a regular tessellation of a rectangle, using identical squares, called the *cells*. In 3d, we consider a regular tessellation of a rectangular cuboid using cubes as cells. A regular grid can be easily triangulated into a valid simplicial complex by considering the Kuhn triangulation [Kuh60] of the cells, which subdivides each 2-cell into two triangles and each 3-cell into six tetrahedras. This is illustrated later on, in Figure 3.3 (page 40). As this triangulation scheme is the same for each cell, the resulting triangulation can be encoded implicitly, which makes its representation in memory extremely efficient in practice.

2.1.2 Topological invariants

This section is an introduction to some topological concepts, that are used to characterize the properties of the domains presented above.

Definition 2.12 (*Topological invariant*) A topological invariant of a topological space is a property which is preserved under the application of homeomorphisms (definition 2.2).

Definition 2.13 (*Connectedness*) A PL manifold $\mathcal{M}$ is *connected* if there exists a *path* between any pairs of points p_0, p_1 of $\mathcal{M}$, *i.e.* a continuous function $\phi : [0,1] \to \mathcal{M}$ with $\phi(0) = p_0$ and $\phi(1) = p_1$.

Definition 2.14 (*Connected component*) The largest connected subsets of $\mathcal{M}$ are called its *connected components*. The connected components of a PL manifold are themselves PL manifolds.

The connectedness of a topological space, and its number of connected components, are topological invariants. For instance, a PL manifold $\mathcal{M}$, which is a triangulation of a manifold $\mathscr{M}$, has the same number of connected components than $\mathscr{M}$. Other topological invariants are used to describe topological spaces that are homeomorphic, independently of their

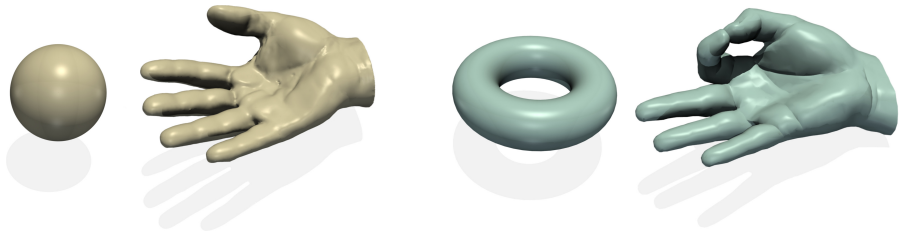


Figure 2.3 – *Examples of 2-manifolds with different Betti numbers. The sphere and the leftmost hand, in beige, are homeomorphic. Their Betti numbers are* $\beta_0 = 1$, $\beta_1 = 0$ *and* $\beta_2 = 1$. *They are not homeomorphic to the torus or the other hand, in cyan. For those,* $\beta_0 = 1$, $\beta_1 = 1$ *and* $\beta_2 = 1$, *as a non-collapsible cycle can be drawn around the handles.*

geometrical shape. In particular, the *Betti numbers* are the topological invariants that are mostly used in our work, and that are defined through the notion of *homology*.

In a few words, homology is a description of topological spaces based on the number and type of "holes" that they contain. For instance, a disk is a 2-dimensional manifold, and so is a sphere. However the shape of the sphere is intrinsically different, as the sphere has no boundary, and is hollow: it contains a void. Like the sphere, the torus is a 2-dimensional manifold with no boundary, and it also contains a void. However the shape of the torus fundamentally differs from the shape of the sphere, as it presents another kind of "hole", that resembles more a handle than a void. Those three manifolds are not homeomorphic.

Definition 2.15 (*Betti numbers*) The homology of a PL d-manifold $\mathcal{M}$ (with $d \leq 3$) of $\mathbb{R}^3$ is described by its Betti numbers $\beta_0, \beta_1, \ldots \beta_{d-1} \in \mathbb{N}$. β_0 indicates the number of connected components of $\mathcal{M}$. β_1 corresponds to its number of handles (or the number of independent 1-cycles in the case of 2-manifolds [EH09]) and β_2 indicates its number of 3-dimensional voids.

Figure 2.3 illustrates the use of Betti numbers to characterize the topological shape of PL manifolds. In this example, all four shapes are closed surfaces, *i.e.* 2-manifolds without boundary (the hands are hollow). Manifolds of same colors are homeomorphic. The sphere and the first hand, in beige, both verify $\beta_0 = 1$, $\beta_1 = 0$ and $\beta_2 = 1$. In contrast, the torus and the second hand exhibit a hole, which is described by the value of their 1-st Betti number, $\beta_1 = 1$. In particular, the two hands are not homeomorphic: one has the topological shape of a sphere, and the other of a torus.

Betti numbers can be defined in any dimension. Formally, the p-th Betti number β_p of $\mathcal{M}$ is defined as the rank of the p-th homology group $H_p(\mathcal{M})$. $H_p(\mathcal{M})$ is the quotient group $\ker \partial_p / \mathrm{Im}\ \partial_{p+1}$, where ∂_p is the

boundary operator on the group of *chains* of p-simplices. This contruction induces an equivalence relation on the group of so-called p-cycles: two cycles are equivalent, or *homologous*, if they can be continuously transformed into each other, through the addition (with modulo 2 coefficients) of p-simplices, without being reduced to a point. As such, β_p represents the number of non-equivalent cycles of p-simplices in $\mathcal{M}$. For more details about the construction of simplicial homology, we refer to [EH09].

2.1.3 Scalar field representation

Recall that we focus in this work on *scalar* data, that is formalized hereafter:

Definition 2.16 (*Barycentric coordinates*) Let p be a point of $\mathbb{R}^d$ and σ a d-simplex. p can be expressed with a linear combination of the 0-faces $\{v_i\}_{0 \leq i \leq d}$ of σ: $p = \sum_{i=0}^{d} \alpha_i v_i$, where $\alpha_i \in \mathbb{R}$ and $\sum_{i=0}^{d} \alpha_i = 1$.

The $\{\alpha_i\}_{0 \leq i \leq d}$ are the *barycentric coordinates* of p relatively to σ. The point p belongs to σ if and only if $\alpha_i \in [0, 1]$ for all i.

Definition 2.17 (*Piecewise Linear Scalar Field*) Let $\mathcal{T}$ be a triangulation and h be a function that maps the vertices of $\mathcal{T}$ to $\mathbb{R}$. Let f be the function mapping any point p of a d-simplex σ of $\mathcal{T}$ to a value $f(p) = \sum_{i=0}^{d} \alpha_i h(v_i)$, where $\{\alpha_i\}_{0 \leq i \leq d}$ are the barycentric coordinates of p relatively to σ. We say that f is linearly interpolated from h on σ. f is called a piecewise linear (PL) scalar field on $\mathcal{T}$.

The notion of PL scalar field is illustrated in Figure 2.4. The field is characterized by its values on the vertices of $\mathcal{M}$, and linearly interpolated on the simplices of higher dimension. This interpolation is efficiently computed on modern hardware, and is a key operation in scientific visualization, as it allows to visually represent the different values of the considered field thanks to an interpolation between colors, resulting in a colormap.

In scientific visualization, some PL scalar fields that we will handle are for instance the electronic density for molecular datasets (Figure 3.1), the viscosity in fluid dynamics simulation results (Figure 5.9), material density in acquired CT scans (Figure 3.17, Figure 4.1), or the magnitude of a vector field (wind velocity in Figure 5.2, magnetic field in Figure 6.4).

Definition 2.18 (*Lower and Upper links*) Let f be a PL scalar field on $\mathcal{T}$. The lower link of a simplex $\sigma \in \mathcal{T}$, noted $Lk^-(\sigma)$, is the subset of the link of σ whose vertices have a strictly lower value than the vertices of σ. Conversely, the upper

link $Lk^+(\sigma)$ is the subset of $Lk(\sigma)$ whose vertices have a strictly higher value than the vertices of σ.

In the remainder of this thesis, we only consider PL scalar fields that are injective on the set of vertices of $\mathcal{M}$, *i.e.* that verify $f(v_0) \neq f(v_1)$ for two different vertices v_0, v_1 of $\mathcal{M}$. This requirement is easy to enforce in practice with a local perturbation of f, on the model of *simulation of simplicity* [EM90]. In our application, we apply this perturbation symbolically: we simply consider an additional injective offset field $\mathbb{O}$ on the set of vertices, which we usually set to the index of vertices in the triangulation. In case of equality of f values between different vertices, we disambiguate the inequality using the field $\mathbb{O}$. This point is illustrated in Section 4.2.4, when it arises in our work.

To summarize, our input data is represented by PL scalar fields defined on PL manifolds, which are objects for which we can use topology to characterize their shape. In the following sections, we will apply topological considerations to geometrical constructions that can be defined for PL scalar fields: the *level sets* and *sub-level sets*, illustrated in Figure 2.4.

Definition 2.19 (*Level set*) The level set $f^{-1}(w)$ of an isovalue $w \in \mathbb{R}$ with respect to a PL scalar field $f : \mathcal{M} \to \mathbb{R}$ is defined as the pre-image by f of w:

$$f^{-1}(w) = \{p \in \mathcal{M} \mid f(p) = w\}$$

Definition 2.20 (*Sub-level set*) The sub-level set $f^{-1}_{-\infty}(w)$ of an isovalue $w \in \mathbb{R}$ with respect to a PL scalar field $f : \mathcal{M} \to \mathbb{R}$ is defined as the pre-image by f of $(-\infty, w)$:

$$f^{-1}_{-\infty}(w) = \{p \in \mathcal{M} \mid f(p) < w\}$$

The study of the topology of the sub-level set of f for various isovalues enables to characterize features of interest in the data. In Sections 2.2 and 2.3, we will present how this characterization takes place and how it allows to encode the main features of a PL scalar field in reduced representations, what we generally call *topological abstractions*. We will particularly focus on the notion of *critical points* and *persistence diagram*.

In this manuscript, scalar fields will be visualized with the help of a *colormap* (i.e. a correspondence between f values and colors). For visual consistency purposes, we will use a unique, divergent color palette throughout the manuscript, as illustrated in Figure 2.4. This palette has been generated from a painting by Bruce Marsh, using the method by Samsel et al. [SBB18]. This choice is motivated by the fact that a good amount of datasets from diverse application fields is presented in this

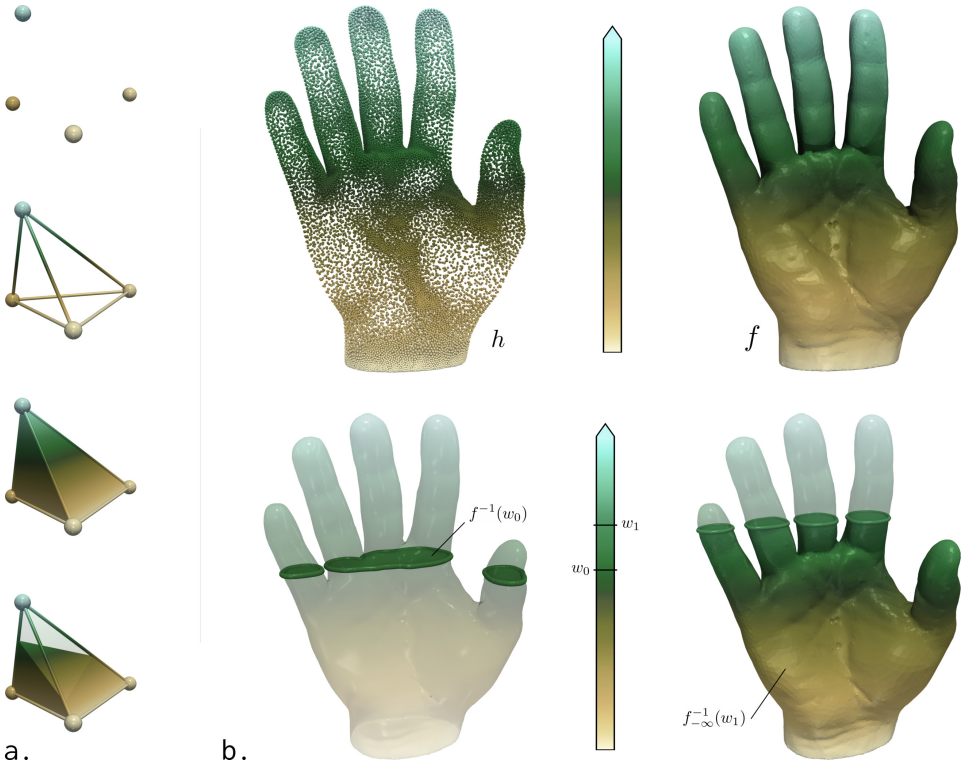


Figure 2.4 – *Piecewise linear (PL) scalar field. The values of the field are given at the vertices of the domain (b, top left) and linearly interpolated on the simplices of higher dimensions using barycentric coordinates (a), resulting in a PL scalar field (b, top right). The level set (b, bottom left, opaque surface) and sub-level set (b, bottom right, opaque volume) of f are geometrical constructions that can be computed on the field.*

thesis. In consequence, selecting the same color palette enforces visual consistency, which helps interpret the numerous figures present in this thesis. The narrow saturation and hue ranges of this palette provide a good perceived contrast [SBB18], adapted to the communication of complex information. Additionally, we find this palette aesthetically pleasing. Figure 2.4 shows the colormap as a vertical, colored arrow (center), from light brown (low f values, bottom) to light green (high f values, top), transiting through dark brown (middle f values, center). The precise correspondence between the f values and the palette often needs to be manually adjusted on a per application basis, for instance, to fine tune the location of the dark brown level sets to relevant isovalues. Since the same color palette is used throughout the manuscript, we will omit the representation of the colormap (arrows in Figure 2.4) in the remainder.

2.2 Critical Points

When considering a scalar field f, as one continuously increases an isovalue w, the topology of the sub-level set $f_{-\infty}^{-1}(w)$ (described by their Betti numbers, in 3D the number of their connected components, handles and voids) only changes at specific locations, named the *critical points* of f [Mil63]. In the PL setting (when considering PL scalar field on a PL manifold $\mathcal{M}$), critical points are bound to correspond to vertices of $\mathcal{M}$. Banchoff [Ban70] introduced a local characterization of critical points, defined as follows.

Definition 2.21 (*Critical points*) Let f be a PL scalar field on a PL manifold $\mathcal{M}$, and v a vertex of $\mathcal{M}$. If $Lk^+(v)$ and $Lk^-(v)$ are simply connected, v is a *regular* point. Otherwise, v is a *critical point* of f, and $f(v)$ is called a critical isovalue.

Definition 2.22 (*Extremum*) Let v be a critical point of a PL scalar field f. If $Lk^+(v)$ is empty, then v is a *local minimum* of f. If $Lk^-(v)$ is empty, then v is a *local maximum* of f. In both cases, v is called a *local extremum* of f.

Definition 2.23 (*Saddle*) Let v be a critical point of a PL scalar field f. If v is neither a local minimum nor a local maximum of f, then v is a *saddle point* of f.

For regular vertices, the sub-level sets enter the neighborhood of v, $St(v)$, through the lower part of the neighborhood boundary, $Lk^-(v)$, and exit through its upper part, $Lk^+(v)$, *without* changing their topology. For critical points, a change in the topology of $f_{-\infty}^{-1}(w)$ is observed, which is concomitant to a change in the topology of the level set in $St(v)$. At the location of local minima, a connected component of level set is created. At the location of local maxima, a connected component of level set disappears. In the case of PL scalar fields defined on 3-manifolds, the level sets locally merge at the location of some saddles, denoted 1-saddles, and locally split at the location of other saddles, called 2-saddles. The local characterization of the *criticality* of vertices (whether they are regular, an extremum or a saddle) is illustrated in Figure 2.5, along with the topological changes of level sets at their locations.

The critical points can be classified with regard to their *index* $\mathcal{I}(v)$, which intuitively corresponds to the number of independent directions of

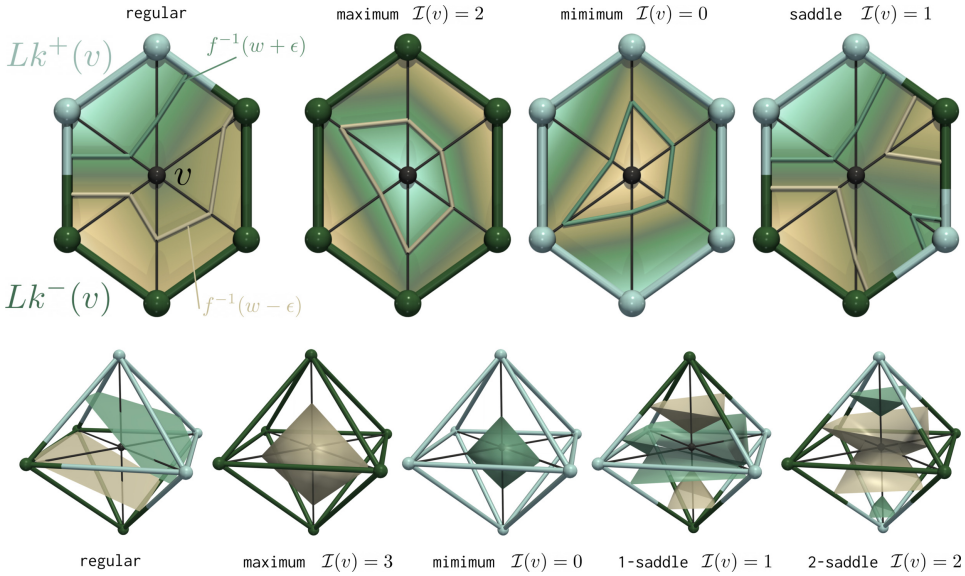


Figure 2.5 – *Critical points and regular points with their star, in 2d (top) and 3d (bottom). The criticality of a vertex v (black sphere) of value w is characterized by the connectedness of its upper link $Lk^+(v)$ (spheres and tubes in cyan), and its lower link $Lk^-(v)$ (spheres and tubes in dark green). In each case, a level set of $St(v)$, taken above the value $w = f(v)$ is shown in light green, and a level set taken below w in beige. For critical points, the level sets undergo topological changes as they pass w.*

decreasing f values around v. It is equal to 0 for local minima ($Lk^-(v) = \emptyset$), to d for local maxima ($Lk^+(v) = \emptyset$) and otherwise to i for i-saddles ($0 < i < d$). The critical index of a critical point v can be formally defined with the notion of *reduced Betti numbers* of $Lk^-(v)$ [EH09].

Adjacency relations between critical points can be captured with the notion of *integral line*.

Definition 2.24 (*Integral lines*) Given a vertex v, its *forward* integral line, noted $\mathcal{L}^+(v)$, is a path along the edges of $\mathcal{M}$, initiated in v, such that each edge of $\mathcal{L}^+(v)$ connects a vertex v' to its highest neighbor v''. Then, forward integral lines are guaranteed to terminate in local maxima of f. A *backward* integral line, noted $\mathcal{L}^-(v)$ is defined symmetrically (i.e. integrating downwards towards minima).

When encountering a saddle s, we say that an integral line *forks*: it yields one new integral line per connected component of $Lk^+(s)$. Note that several integral lines can *merge* (and possibly fork later).

Critical points play a central role in Topological Data Analysis, as they often correspond to features of interest in various applications: centers of vortices in fluid dynamics [KRHH11], atoms in chemistry [BGL+18, GABCG+14, OGT19] or clusters of galaxies in astrophysics

[Sou11, SPN+16]. However in real-world data, in practice, critical points can appear in large and impractical amounts, due to the presence of numerous small features in the data. This can be the consequence of noise in the acquisition process, or numerical noise in the simulations for instance. Depending on the application, this noise may need to be discarded in the analysis, in order to identify critical points describing actual features of interest in the data. This is the purpose of *persistent homology*, introduced in Section 2.3: it provides an importance measure on critical points, which is derived from topological considerations.

In the remainder of this work, we represent critical points using spheres at the localization of the critical points in $\mathcal{M}$, whether $\mathcal{M}$ is 2D or 3D. This simple representation enforces visual consistency across all figures, which eases their interpretation. It is also consistent with the preferred representation of critical points in the Topology ToolKit [TTK20], and more generally ParaView, where vertices are often represented as spheres. We encode the criticality of critical points with our chosen colormap (Section 2.1.3). Extrema are represented with the extreme colors of the palette (light brown for minima, light green for maxima). Saddles are represented with in between colors, as illustrated in Figure 2.6.

2.3 Persistence Diagrams

As seen in the previous section, the topology of the sub-level sets of f evolves at the location of critical points. Specifically, changes occur in the Betti numbers of the sub-level sets.

2.3.1 The case of the 0-th Betti number

Let f be PL scalar field defined on a connected PL manifold $\mathcal{M}$. Let us consider the 0-th Betti number of $f_{-\infty}^{-1}(w)$, that indicates its number of connected components. A new connected component of $f_{-\infty}^{-1}(w)$ is created as w passes the value of each local minimum of f. The *Elder rule* [EH09] indicates that if two connected components, created at the minima m_0 and m_1 with $f(m_0) < f(m_1)$, meet at a given 1-saddle s_0, the *youngest* of the two components (the one created at m_1) *dies* in favor of the *oldest* one (created at m_0). The connected components of $f_{-\infty}^{-1}(w)$ are thus born at the values of the local minima, and die at the values of the 1-saddles. When w reaches the maximum value of f, we have $f_{-\infty}^{-1}(w) = \mathcal{M}$, which only has one connected component: the oldest one, created at the global minimum

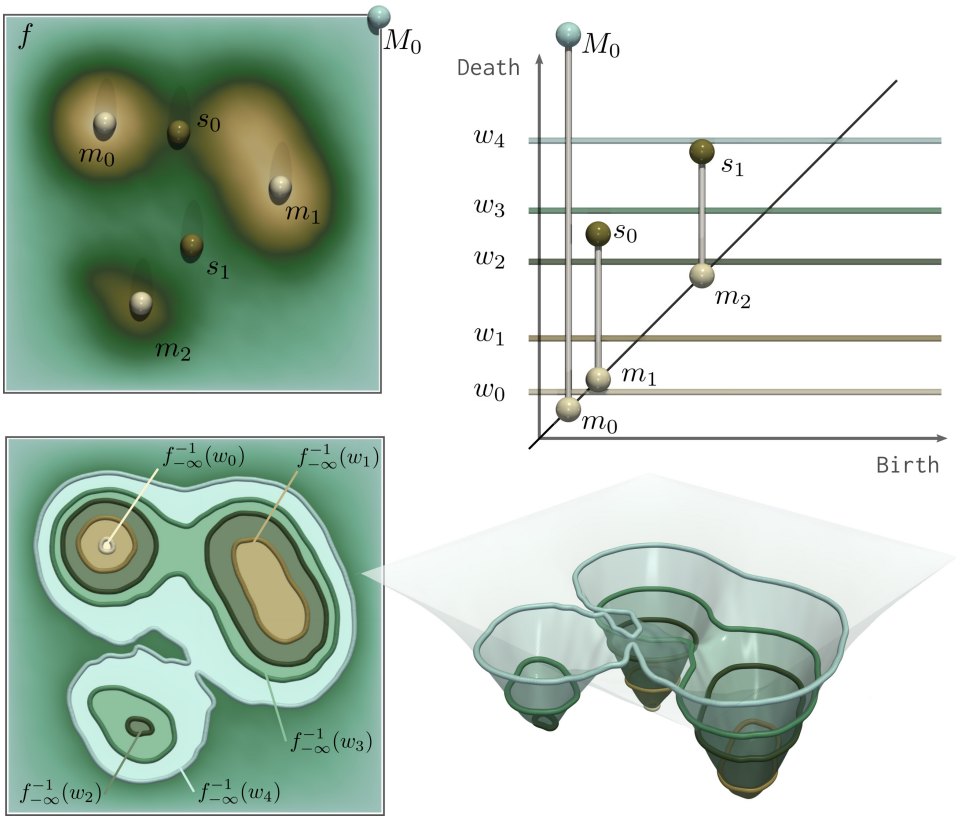


Figure 2.6 – *Minimum/saddle persistence diagram (top right) of a PL scalar field f defined on a 2d domain (top left). The spheres denote the critical points of f (light beige: minima, brown: saddles, cyan: maximum). The diagram tracks the evolution of the connected components of sub-level sets of f (bottom). As the isovalue increases, new connected components are created in m_0, then m_1 and m_2. Then the connected component born in m_1* dies *at s_0, as it merges with the older one created in m_0. This results in the pair (m_1, s_0) in the diagram. Finally, the component created in m_2 dies at s_1, it is represented in the diagram by the pair (m_2, s_1). By convention, the global minimum m_0 is paired with the global maximum M_0.*

of f. This means that every other connected component can be uniquely associated with a so-called *persistence* pair (m, s) of a local minimum m and a 1-saddle s that denote its birth and death locations. The quantity $p = f(s) - f(m)$ is called the *topological persistence* of the pair, and denotes the lifetime of the associated connected component of $f^{-1}_{-\infty}(w)$ in terms of scalar values. This pairing process is illustrated in Figure 2.6

2.3.2 Notions of persistent homology

This example can be generalized to other Betti numbers and arbitrary dimension with the help of *Persistent Homology*, which formulates the Elder rule for homology groups. For different isovalues $w_0 \leq w_1 \leq \ldots \leq w_j$, the

sequence of consecutive sub-level sets verifies:

$$f_{-\infty}^{-1}(w_0) \subseteq f_{-\infty}^{-1}(w_1) \subseteq \ldots \subseteq f_{-\infty}^{-1}(w_j)$$

This nested sequence can be formalized in a nested sequence of simplicial sub-complexes of $\mathcal{M}$, using the concept of *filtration*.

Definition 2.25 (*Filtration*) Let f be an injective scalar field defined on a simplicial complex $\mathcal{K}$, such that $f(\tau) < f(\sigma)$ for each face τ of each $\sigma \in \mathcal{K}$. Let n be the number of simplices of $\mathcal{K}$ and $\mathcal{K}_i$ be the sub-level set of f by the i-th value in the sorted set of simplices values. The nested sequence of sub-complexes $\mathcal{K}_0 \subseteq \mathcal{K}_1 \subseteq \ldots \subseteq \mathcal{K}_{n-1} = \mathcal{K}$ is called the *filtration* of f.

Note that this definition requires to associate a scalar value to simplices with a dimension higher than 0, such as $f(\tau) < f(\sigma)$ for each face τ of each simplex σ. This requirement enforces that $\mathcal{K}_i$ is indeed a simplicial complex. In the case of PL scalar fields, the particular notion of *lower star* filtration is used in practice [EH09].

Recall that we are interested in the evolution of the homology in the filtration, namely in the number of classes in $H_p(\mathcal{K}_i)$ for each dimension p. Since the filtration is a *nested* sequence of complexes, the classes in $H_p(\mathcal{K}_i)$ can be tracked in $H_p(\mathcal{K}_j)$. Formally, that is because the filtration induces a sequence of *homomorphisms* between the homology groups of the sub-complexes of the filtration. A homomorphism is a map between groups that commutes with the group operation. In the case of the homology groups, the group operation is a formal sum of sets (*chains*) of p-simplices with modulo 2 coefficients [EH09]. These maps enable to define the *persistent homology group* $H_p^{i,j}$ as the image of the homomorphism between $H_p(\mathcal{K}_i)$ and $H_p(\mathcal{K}_j)$.

In a few words, as we progress up the filtration, new homology classes (the different connected components for $p = 0$, cycles for $p = 1$, or voids for $p = 2$) can arise in $H_p(\mathcal{K}_i)$. Conversely, classes can disappear if they become trivial or merge with one another. In case of a merger, the Elder rule is applied to state that the youngest one *dies* at the benefit of the oldest, created earlier in the filtration. The p-th persistent homology group $H_p^{i,j}$ regroups the homology classes that were present in $H_p(\mathcal{K}_i)$ and that still *persist* up to $H_p(\mathcal{K}_j)$. The p-th *persistent* Betti number $\beta_p^{i,j}$ is defined as the rank of $H_p^{i,j}$ and indicates the number of these classes.

We once again refer to [EH09] for a formal, more detailed introduction to persistent homology.

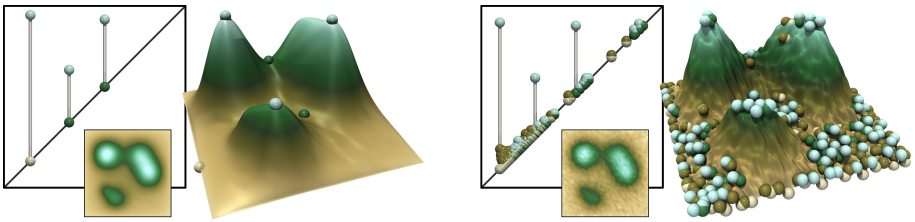


Figure 2.7 – *Saddle/maximum persistence diagrams of a clean (left) and noisy (right) scalar field (light brown spheres: minima, cyan: maxima, others: saddles). The main three hills are clearly apparent in the diagrams (high persistence pairs), whereas small pairs near the diagonal indicate noisy features.*

2.3.3 Topological persistence

We mentioned in Section 2.2 that changes in the homology classes of $f_{-\infty}^{-1}(w)$ occur at the locations of the critical points [Mil63]. Thus each topological feature is characterized by a pair of critical points denoting its birth and death, and whose difference in value indicated its lifespan in the data, called *topological persistence*.

As exemplified in Section 2.3.1, the persistence of connected components of $f_{-\infty}^{-1}(w)$, described by the 0-th homology group, is characterized by minimum/1-saddle critical pairs. For PL scalar fields defined on 3-manifolds, the 2-saddles are paired with maxima and characterize the persistence of the voids of $f_{-\infty}^{-1}(w)$, while 1-saddle/2-saddle pairs describe the persistence of its handles. In the case of 2-manifolds, the 1-saddle/maximum pairs describe the persistence of the independent cycles of $f_{-\infty}^{-1}(w)$.

As a result, critical points can be unambiguously matched in persistence pairs (c_0, c_1), with $f(c_0) < f(c_1)$ and $\mathcal{I}(c_0) = \mathcal{I}(c_1) - 1$, of topological persistence $p = f(c_0) - f(c_1)$. By convention, we pair the global minimum with the global maximum of f, resulting in a persistence pair that spans the whole scalar range of f, although the corresponding connected component of $f_{-\infty}^{-1}(w)$ technically never dies.

The topological persistence of each critical pair gives a measure of importance on the corresponding critical points of the field, that has been shown to be reliable to distinguish between noise and important features. The critical pairs are usually visualized in the *Persistence diagram* [EH09] of f, noted $\mathcal{D}(f)$. The diagram provides a representation of the ensemble of features in the data, where each persistence pair (c_0, c_1) is embedded as a point in the 2D plane, at coordinates $\big(f(c_0), f(c_1)\big)$. The persistence of each pair can thus be read in the diagram as the height of the point to the diagonal. Consequently, each topological feature of $f_{-\infty}^{-1}(w)$ (the

homology classes of $f_{-\infty}^{-1}(w)$: connected components, cycles and voids) can be visualized in the diagram as a bar (Figure 2.6 and Figure 2.7), whose height indicates its importance in the data. Large bars corresponding to high persistence features stand out visually, while low persistence pairs, likely to be associated with noisy features, are represented by small bars in the vicinity of the diagonal. The persistence diagram is a concise visual depiction of the repartition of features in the data and has been shown to be a stable [CEH05, CCG+09] and useful tool for data summarization tasks. As seen in Figure 2.7, it encodes the number, ranges and salience of features of interest, and gives hints about the level of noise in the data.

In practical applications, features of interest are often characterized by the extrema of the field. Thus, in the remainder of this thesis, when considering persistence diagrams, we will focus on minimum/1-saddle pairs and $(d-1)$-saddle/maximum pairs. The interpretation of features characterized by 1-saddle/2-saddle pairs is usually more difficult in our applications.

We chose to represent persistence diagrams with spheres and tubes, as seen in Figure 2.7, which is the representation used in the Topology ToolKit [TTK20]. By using spheres, we emphasize that persistence pairs actually correspond to a pair of two critical points (represented themselves as spheres in the input data), and the relation to the spheres denoting the critical points in the domain is further highlighted by the usage of a common color map (encoding the criticality of the corresponding critical points). Although to our knowledge no user study was conducted to compare the effectiveness of different visualizations of persistence diagrams, we believe that using such 3D glyphs helps distinguish nearby pairs.

2.4 Metrics on the space of Persistence Diagrams

The definition of metrics between topological abstractions is an important topic, as it enables the comparison and similarity estimation of scalar fields based on their topology. For persistence diagrams, well-established metrics have been defined and intensively studied, such as the *Wasserstein* distance and the *Bottleneck* distance. The persistence diagram is *stable* under these two metrics [CEH05, CSEHM10], which intuitively means that a small variation in the scalar fields entails a small difference in the resulting distances.

2.4.1 Wasserstein distance between diagrams

The Wasserstein distance was originally defined in the context of Transportation theory [Kan42, Mon81]. For persistence diagrams, intuitively, this distance aims at optimizing a matching between the features of two diagrams to compare and penalizes mismatches between these diagrams.

Given two diagrams $\mathcal{D}(f)$ and $\mathcal{D}(g)$, a pointwise distance, noted d_q, inspired from the L^p norm, can be introduced in the 2D birth/death space between two points $a = (x_a, y_a) \in \mathcal{D}(f)$ and $b = (x_b, y_b) \in \mathcal{D}(g)$, with $q > 0$, as follows :

$$d_q(a,b) = \left(|x_b - x_a|^q + |y_b - y_a|^q\right)^{1/q} = \|a - b\|_q \tag{2.1}$$

By convention, $d_q(a,b)$ is set to zero if both a and b exactly lie on the diagonal ($x_a = y_a$ and $x_b = y_b$). The L_q-Wasserstein distance, noted W_q, between $\mathcal{D}(f)$ and $\mathcal{D}(g)$ can then be introduced as:

$$\boxed{W_q\big(\mathcal{D}(f), \mathcal{D}(g)\big) = \min_{\phi \in \Phi} \left(\sum_{a \in \mathcal{D}(f)} d_q\big(a, \phi(a)\big)^q \right)^{1/q} \qquad (2.2)}$$

where Φ is the set of all possible assignments ϕ mapping each point $a \in \mathcal{D}(f)$ (diagonal included) to a point $b \in \mathcal{D}(g)$ (diagonal included). Note that such a mapping can map a point $a \in \mathcal{D}(f)$ to its *diagonal projection*, $\Delta(a) = (\frac{x_a+y_a}{2}, \frac{x_a+y_a}{2})$, in $\mathcal{D}(g)$ (or reciprocally), which denotes the removal (or the insertion) of the corresponding feature from the assignment, with a cost $d_q\big(a, \Delta(a)\big)^q$ (Figure 2.8). The Wasserstein distance can be computed by solving an optimal assignment problem, for which existing algorithms [Mun57, Mor10] however often require a balanced setting. To address this, the input diagrams $\mathcal{D}(f)$ and $\mathcal{D}(g)$ are typically *augmented* into $\mathcal{D}'(f)$ and $\mathcal{D}'(g)$, which are obtained by injecting the diagonal projections of all the off-diagonal points of one diagram into the other:

$$\mathcal{D}'(f) = \mathcal{D}(f) \cup \{\Delta(b) \mid b \in \mathcal{D}(g), x_b \neq y_b\} \tag{2.3}$$

$$\mathcal{D}'(g) = \mathcal{D}(g) \cup \{\Delta(a) \mid a \in \mathcal{D}(f), x_a \neq y_a\} \tag{2.4}$$

In this way, the Wasserstein distance is guaranteed to be preserved by construction, $W_q\big(\mathcal{D}(f), \mathcal{D}(g)\big) = W_q\big(\mathcal{D}'(f), \mathcal{D}'(g)\big)$, while making the assignment problem balanced ($|\mathcal{D}'(f)| = |\mathcal{D}'(g)|$) and thus solvable with traditional assignment algorithms. In the following, we will focus on $q = 2$.

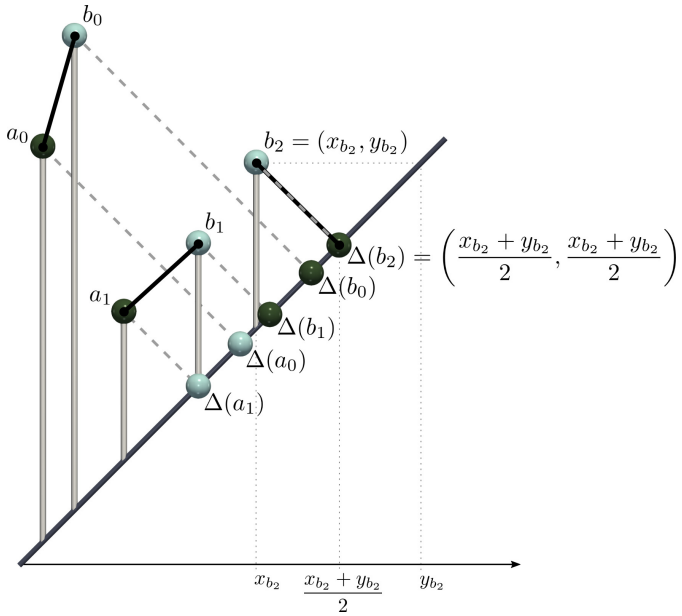


Figure 2.8 – *Example of a diagonal matching in the assignment between two diagrams $\mathcal{D}(f)$ and $\mathcal{D}(g)$ of different cardinality. The augmented diagrams $\mathcal{D}'(f)$ (green spheres) and $\mathcal{D}'(g)$ (cyan spheres) are constructed by including the orthogonal projections $\Delta(a)$ (grey dashed lines) of each point a onto the diagonal of the other diagram. The assignment problem thus becomes balanced (matchings are shown with black lines). This allows a persistence pair to be matched to a diagonal point (b_2 and $\Delta(b_2)$). Diagonal points of different diagrams can be matched together with zero cost.*

Geometrical lifting

In the applications, it can often be useful to geometrically lift the Wasserstein metric, by also taking into account the geometrical layout of critical points [SPCT18b]. Let (c, c') be the critical point pair corresponding to the point $a \in \mathcal{D}(f)$. Let $p_a^\lambda \in \mathbb{R}^d$ be their linear combination with coefficient $\lambda \in [0,1]$ in $\mathcal{M}$: $p_a^\lambda = \lambda c' + (1-\lambda)c$. Then, the geometrically lifted pointwise distance $\widehat{d_2}(a,b)$ can be given as:

$$\widehat{d_2}(a,b) = \sqrt{(1-\alpha)d_2(a,b)^2 + \alpha||p_a^\lambda - p_b^\lambda||_2^2} \tag{2.5}$$

The parameter $\alpha \in [0,1]$ quantifies the importance given to the geometry of critical points and it must be tuned on a per application basis. The parameter λ tunes the relative importance given to the geometrical location of each critical point of the persistence pairs. As in our applications, the features of interest are often captured by the extrema of the field, we recommend to set $\lambda = 1$ when considering saddle/maximum persistence pairs, and $\lambda = 0$ when dealing with minimum/saddle persistence pairs.

2.4.2 Bottleneck distance between diagrams

When q tends to infinity, the Wasserstein distance converges to the *Bottleneck* distance, another practical metric that measures the worst mismatch between $\mathcal{D}(f)$ and $\mathcal{D}(g)$:

$$W_\infty(\mathcal{D}(f), \mathcal{D}(g)) = \min_{\phi \in \Phi} \left(\max_{a \in \mathcal{D}(f)} \|a - \phi(a)\|_\infty \right) \tag{2.6}$$

At the difference of the Wasserstein distance, the Bottleneck distance does not account for the difference in the numbers of features between diagrams. As such, it is usually deemed less informative than the Wasserstein distance. However the Bottleneck distance between two diagrams has more explicit stability properties: it is bounded by the L_∞ distance between the corresponding scalar fields [CEH05]:

$$W_\infty(\mathcal{D}(f), \mathcal{D}(g)) \leq \|f - g\|_\infty \tag{2.7}$$

2.5 Other topological abstractions

Some applications call for the use of topological abstractions that are more discriminative than the critical points or the persistence diagram. Although they do not represent the core subject of our work, we give hereafter some examples for completeness, and to give a glance of the variety of tools that topological data analysis provides. We briefly introduce the notions of *merge* and *contour* trees, *Reeb graphs* and *Morse-Smale complexes*. We provide illustrations of these topological abstractions in Figure 2.9.

Merge and contour trees

The *merge tree* tracks the merge events of connected components of sub-level sets that were described in Section 2.3.1. If we continuously draw a path that tracks these components from their birth to their death, we are left with a graph (a 1-simplicial complex) that starts at a local minimum and merge at a saddle with another path corresponding to another connected component of sub-level set. As the connected components merge but cannot split, the whole graph is a tree, called the merge tree. If we build these paths in the ascending order of vertices values, we are left with monotone paths that visit every vertex. The corresponding tree is called

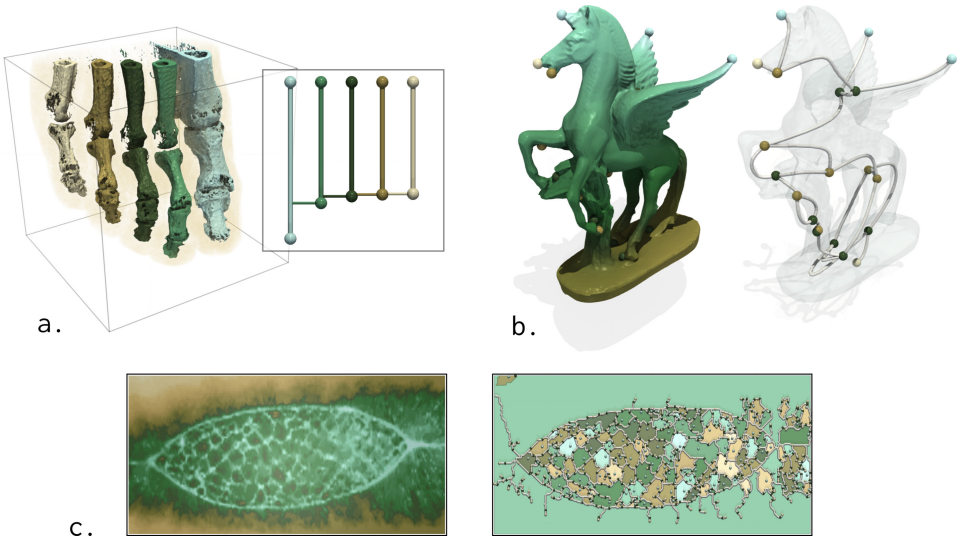


Figure 2.9 – *Examples of other topological abstractions. The merge tree (a, right) induces a segmentation of the domain (a, left) which enables to extract the bones of the five toes. The Reeb graph (b) enables the extraction of the skeleton of a complex shape. The Morse-Smale complex (c, right) provides the segmentation of the domain based on the gradient of the field. In this example, it enables the segmentation of the cells in this classical [EH09] acquired biological dataset (c, left). These examples and the corresponding data are available in the Topology ToolKit's public repository [TTK20].*

an *augmented merge tree* and provides a segmentation of the domain, that associates for each vertex the connected component of sub-level set that first reached it.

The *contour tree* is a generalization of the merge tree that tracks in addition the split events of the level sets, and induces a different segmentation of the domain.

The saddle/extremum persistence diagram can easily be deduced from the merge tree, for which efficient computation techniques have been developed, with optimal complexity in all dimensions [CSA00], and with important acceleration thanks to shared-memory parallelism [GFJT19]. This last method is the implementation available by default in the Topology ToolKit [TFL+17] for the computation of the saddle/extremum persistence diagram.

Reeb graphs

The Reeb graph [Ree46] is the generalization of the contour tree to domains that are not simply connected (*i.e.* that contains holes), which motivates its use for shape analysis applications [BGSF08]. Like the merge and contour trees, it provides a segmentation of the domain that is useful in

scientific visualization and data analysis, for feature extraction purpose, or to provide an indexing of contours, so as to efficiently extract level sets.

Morse-Smale complexes

An other widely used topological abstraction is the Morse-Smale complex [DFFIM15], which encodes the relations between critical points in terms of unique integral lines of the gradient field. These integral lines segment the domain into cells in which the gradient integrates to identical extremities. The Morse-Smale complexe provides a cellular partition of the domain which shows itself useful in scientific visualization, to capture features that coincide with the gradient.

3 A Progressive Approach to Scalar Field Topology

Contents

Our contribution in one image 35

3.1 Context 36

3.1.1 Related Work 36

3.1.2 Contributions 37

3.2 Progressive Data Representation 38

3.2.1 Edge-Nested Triangulation Hierarchy 38

3.2.2 Edge-Nested Triangulations of Regular Grids 40

3.2.3 Topologically Invariant Vertices 42

3.3 Progressive Critical Points 46

3.3.1 Initialization and Updates 47

3.3.2 Computation Shortcuts 49

3.3.3 Parallelism 49

3.3.4 Extremum Lifetime 49

3.4 Progressive Persistence Diagrams 50

3.4.1 Persistence Diagram from Critical Points 52

3.4.2 Progressive Strategy 54

3.4.3 Parallelism 54

3.5 Results 55

3.5.1 Progressive Data Representation 55

3.5.2 Time Performance 58

3.5.3 Stress Cases 61

3.5.4 Progressive Topological Visualization and Analysis 62

3.6 Limitations and Discussion 65

3.7 Summary 66

This chapter introduces progressive algorithms for the topological analysis of scalar data. Our approach is based on a hierarchical representation of the input data and the fast identification of *topologically invariant vertices*, which are vertices that have no impact on the topological description of the data and for which we show that no computation is required as they are introduced in the hierarchy. This enables the definition of efficient coarse-to-fine topological algorithms, which leverage fast update mechanisms for ordinary vertices and avoid computation for the topologically invariant ones. We demonstrate our approach with two examples of topological algorithms (critical point extraction and persistence diagram computation), which generate interpretable outputs upon interruption requests and which progressively refine them otherwise. Experiments on real-life datasets illustrate that our progressive strategy, in addition to the continuous visual feedback it provides, even improves run time performance with regard to non-progressive algorithms and we describe further accelerations with shared-memory parallelism. We illustrate the utility of our approach in batch-mode and interactive setups, where it respectively enables the control of the execution time of complete topological pipelines as well as previews of the topological features found in a dataset, with progressive updates delivered within interactive times.

The work presented in this chapter has been published in the journal IEEE Transactions on Visualization and Computer Graphics in 2021 [VGT21]. It was certified replicable by the Graphics Replicability Stamp Initiative (http://www.replicabilitystamp.org/). The code and data needed to reproduce the results presented in this chapter are available at https://github.com/julesvidal/progressive-scalar-topology. Additionally, our implementation has been integrated in the Topology ToolKit [TFL+17].

Our contribution in one image

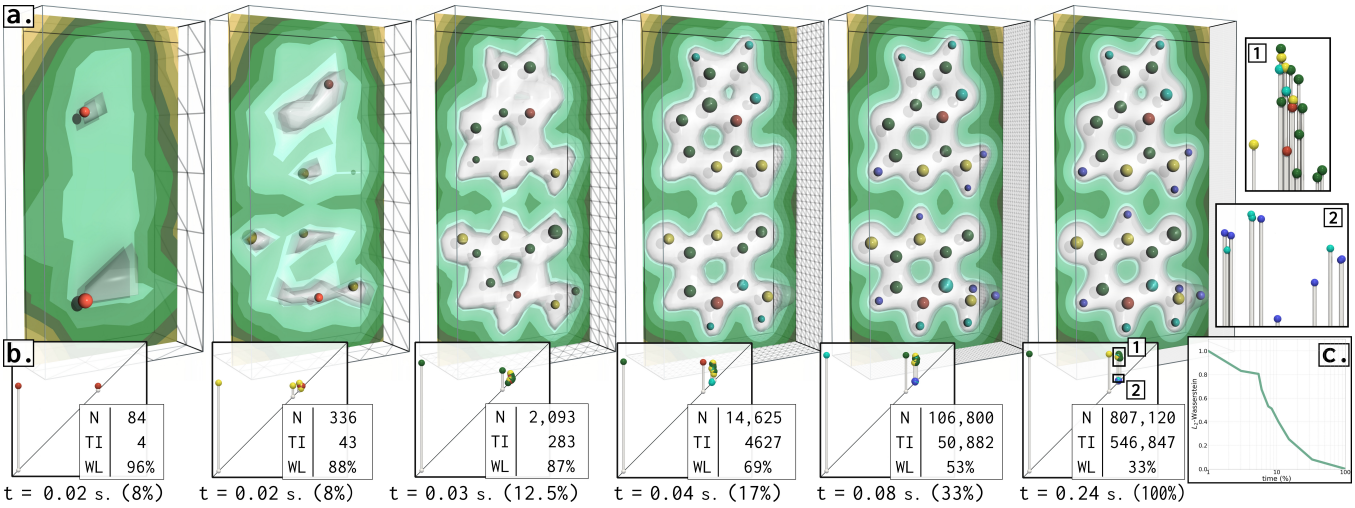


Figure 3.1 – *Progressive persistence diagrams (saddle-maximum pairs) of the electron density of the adenine-thymine (AT) molecular system (an isosurface shows the two molecules), for a few steps of the progressive computation. Our coarse-to-fine approach efficiently refines the persistence diagram by progressing down a hierarchical representation of the input (from left to right). Maxima (denoting the atoms) are shown in the domain (a) with spheres, scaled by topological persistence and colored by lifetime in the data hierarchy (from red to dark blue). In this example, the persistence diagram (b) progressively captures the main features of the data. As of 8% of the computation time (leftmost), two persistent maxima are captured, denoting the presence of two molecules. As the computation progresses, atoms are progressively captured, heavier atoms first: oxygens, then nitrogens and carbons, and finally hydrogens are respectively all captured as of 12.5%, 17% and 33% of the computation time. At this point the diagram is complete and its accuracy is then improved until the final, exact result (rightmost). This qualitative progression is confirmed quantitatively by the empirical convergence of the L_2-Wassserstein distance to the final output (c), which is monotonically decreasing: more computation time indeed yields more accuracy. Our algorithms leverage efficient update mechanisms and* topologically invariant vertices (TI)*, which can be quickly identified and for which we show that no computation is required, thus drastically reducing the workload* (WL) *of the algorithm with time. Overall, our progressive approach efficiently computes the persistence diagram of the data, while continuously providing relevant visual feedback.*

3.1 Context

3.1.1 Related Work

In this chapter, we introduce the first progressive algorithms for the computation of topological abstractions, namely critical points (Section 3.3) and persistence diagrams (for extremum-saddle pairs, Section 3.4). Our approach is based on a hierarchical representation of the data. Multiresolution hierarchies have been considered before, for the Reeb graph [HSKK01b], the contour tree [PCMS04] and the Morse-Smale complex [BEHP03, GRP+12, IF17], but the hierarchical aspect dealt with the *output* data structure, while the input was processed without multiresolution, with existing algorithms [BFS00, CSA00, EHZ01]. In contrast, in our work, the *input* data is represented as a multiresolution hierarchy and the output is efficiently, progressively updated in a coarse-to-fine manner, by iterating through the hierarchy levels.

Our progressive scheme relies on a hierarchical representation of the input data. In the visualization community, many types of hierarchies have been defined to encode and extract visual representations from volumetric data at different levels of details [GDL+02, WF09b, WF09a, PB00, LVLM04, GP00]. For example, Gerstner and Pajarola [GP00] introduce a method for the robust extraction of isosurfaces in multiresolution volume representations. For this, their algorithm extracts the critical points of the input scalar field, for each level of their hierarchical scheme. However, they use for this the standard, non-progressive, procedure [Ban70]. In contrast, our approach extracts the critical points for all of our hierarchy levels *progressively*, i.e. without recomputing from scratch critical points at each new hierarchy level, but instead by efficiently and minimally updating the information already computed at the previous levels. Generally, in our work, we focus on a specific scheme based on the so-called *red* subdivision [H. 42, R.E83, L0087, J. 95, S. 95] applied to regular grids [Kuh60, Bey98], in particular to investigate progressive and efficient *coarse-to-fine* computations, in contrast to the traditional fine-to-coarse hierarchical approaches found in the visualization literature.

The approaches which are the most related to our work are probably the streaming algorithms for computing the Reeb graph [PSBM07] and the merge tree [BWT+11]. These algorithms are capable of computing their output in a streaming fashion: the simplices of the input domain can be processed in arbitrary order and these algorithms maintain and itera-

tively complete their output data structure. However, while they can be interrupted, these algorithms are not, strictly speaking, *progressive*: upon interruption, they do not provide interpretable but *partial* results, which are very far in practice from the final result. For instance, the streaming Reeb graph [PSBM07] can typically count at a given time a very large number of loops (which will be iteratively filled as the algorithm progresses). In contrast, our coarse-to-fine algorithms provide interpretable results upon interruption, which are visually similar to the exact, final outputs and which empirically quickly converge towards them.

3.1.2 Contributions

This chapter presents the following new contributions:

1. *A progressive data representation (Section 3.2)* We present an approach for the progressive topological analysis of scalar data, to generate interpretable outputs upon interruption requests. Our approach relies on a hierarchical representation of the input data (derived from established triangulation subdivision schemes [H. 42, Kuh60, R.E83, Loo87, J. 95, S. 95, Bey98]) and the fast identification of the new notion of *topologically invariant vertices*, for which we show that no computation is required as they are introduced in the hierarchy.

2. *A progressive algorithm for critical point extraction (Section 3.3)* We introduce a progressive algorithm for critical point extraction. As it progresses down the data hierarchy, our algorithm leverages efficient update mechanisms for ordinary vertices and avoids computation for the topologically invariant ones. This enables a progressive output refinement, which results in even faster overall computations than non-progressive methods. We also introduce a fast heuristic to evaluate the lifetime of critical points in the data hierarchy.

3. *A progressive algorithm for persistence diagram computation (Section 3.4)* We introduce a progressive algorithm for the computation of persistence diagrams of extremum-saddle pairs, built on top of the above contributions. In practice, our algorithm tends to capture the main features of the data first, and then progressively improves its accuracy. This is confirmed quantitatively by the empirical convergence of the Wasserstein distance to the final output, which is monotonically decreasing (more computation time indeed yields more accuracy). Our approach enables

a continuous visual feedback, while being in practice even faster overall than non-progressive methods.

3.2 Progressive Data Representation

This section details our hierarchical scheme for the progressive representation of the input data, which relies on a hierarchy of triangulations $\mathcal{H}$ derived from established subdivision schemes [H. 42, R.E83, Loo87, J. 95, S. 95]. In particular, our goal is to define a hierarchical scheme that will enable efficient update mechanisms between hierarchy levels. This will avoid, at each new level of the hierarchy, the recomputation from scratch of the topological data representations presented in Sections 3.3 and 3.4, and this will instead enable their progressive update. After a generic description of the employed triangulation hierarchy (Section 3.2.1), we present for completeness an efficient implementation [Kuh60, Bey98] for the special case of triangulations of regular grids (Section 3.2.2), on which we focus in this thesis (Section 3.6 discusses generalizations). Next, we resume our generic description (Section 3.2.3) and show how to leverage the specific structure of the employed triangulation hierarchy to accelerate the topological analysis of the data. For this, we introduce the novel notion of *Topologically Invariant Vertices*, which is central to our work.

3.2.1 Edge-Nested Triangulation Hierarchy

Our progressive representation of the input data is based on a multiresolution hierarchy of the input PL-manifold $\mathcal{M}$, which relies on established subdivision schemes [H. 42, R.E83, Loo87, J. 95, S. 95]. Intuitively, our goal is to define a multiresolution hierachy that will enable the efficient update of the topological information computed at the previous levels, in order to avoid full re-computations (Section 3.3). In order to construct such a hierarchical scheme, as formalized next, we impose that, as one progresses down the hierarchy, new vertices are only inserted along pre-existing edges (exactly one new vertex per edge, typically at their center), and that the additional new edges only connect new vertices (Figure 3.2). This will have the beneficial effect of preserving, from one hierarchy level to the next, the *structure* of the local neighborhood around each pre-existing vertex (of its link, as discussed in Section 3.2.3), which will in turn effectively enable fast updates of the pre-existing local topological information (Section 3.3). We call such a hierarchy *edge-nested* and we for-

malize it in the following, to introduce the notations that will be used in the rest of the chapter. Let $\mathcal{H} = \{\mathcal{M}^0, \mathcal{M}^1, \ldots, \mathcal{M}^h\}$ be a hierarchy of PL d-manifolds, which respects the following key conditions.

1. **Old Vertex Condition:** Each vertex of $\mathcal{M}^i$ (the triangulation at level i) also belongs to the vertex set, noted $\mathcal{M}_0^{i+1}$, of $\mathcal{M}^{i+1}$:

$$\mathcal{M}_0^i \subset \mathcal{M}_0^{i+1} \tag{3.1}$$

   The vertices of $\mathcal{M}^{i+1}$ already present in $\mathcal{M}^i$ are called *old vertices* (black spheres in Figure 3.2).

2. **New Vertex Condition:** Each vertex of $\mathcal{M}^{i+1}$ not present in $\mathcal{M}^i$ has to be located on an edge (v_0, v_1) of $\mathcal{M}^i$ (typically at its center), as summarized below, where $\mathcal{M}_1^i$ stands for the edge set of $\mathcal{M}^i$:

$$\forall v \in \mathcal{M}_0^{i+1}, v \notin \mathcal{M}_0^i : \quad \exists (v_0, v_1) \in \mathcal{M}_1^i, \ v \in (v_0, v_1) \tag{3.2}$$

   The vertices of $\mathcal{M}^{i+1}$ not present in $\mathcal{M}^i$ are called *new vertices* (white spheres in Figure 3.2).

3. **Old Edge Condition:** Each edge (v_0, v_1) of $\mathcal{M}^i$ has to be subdivided at level $i+1$ at exactly one new vertex v of $\mathcal{M}^{i+1}$:

$$\begin{aligned} \forall (v_0, v_1) \in \mathcal{M}_1^i : \qquad & |\{v \in (v_0, v_1), \ v \notin \mathcal{M}_0^i, \ v \in \mathcal{M}_0^{i+1}\}| = 1 \\ & (v_0, v) \in \mathcal{M}_1^{i+1}, \quad (v, v_1) \in \mathcal{M}_1^{i+1} \\ & (v_0, v_1) \notin \mathcal{M}_1^{i+1} \end{aligned} \tag{3.3}$$

   The edges of $\mathcal{M}^{i+1}$ obtained by subdivision of an edge of $\mathcal{M}^i$ are called *old edges*, they connect old vertices to new vertices (gray cylinders in Figure 3.2).

4. **New Edge Condition:** Each edge of $\mathcal{M}^{i+1}$ which is not an old edge has to connect two new vertices, and it is called a *new edge* (white cylinders in Figure 3.2).

Figure 3.2 presents a simple example of 2D edge-nested triangulation hierarchy. Note that the Loop subdivision [Loo87] is compatible with the above formalization, which is more generally termed as *red* subdivision in the scientific computing literature, and which has been extensively studied for domains of two [R.E83], three [J. 95, S. 95] and arbitrary dimensions [H. 42]. An input PL manifold $\mathcal{M}$ admits an edge-nested triangulation hierarchy if there exists a hierarchy $\mathcal{H}$ for which $\mathcal{M}$ is the last element ($\mathcal{M} = \mathcal{M}^h$).

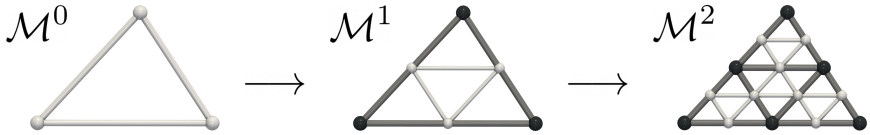


Figure 3.2 – *Edge-nested triangulation hierarchy for a simple 2D example. Old vertices/edges are shown in black/gray. New vertices and edges are shown in white.*

3.2.2 Edge-Nested Triangulations of Regular Grids

While the construction of an edge-nested triangulation hierarchy given an arbitrary input manifold $\mathcal{M}$ is an open question which we leave for future work (see Section 3.6), it can be shown that such a hierarchy exists for regular grids, and that it can be implemented very efficiently, as discussed by Bey [Bey98]. We describe this implementation in the following for the sake of completeness, by detailing how to efficiently retrieve an arbitrarily coarse version of the fine triangulation $\mathcal{M}^h$ from an input regular grid $\mathcal{G}^0$.

Let $\mathcal{G}^0$ be a d-dimensional regular grid, with d equal to 2 or 3 in our applications, of dimensions L_x^0, L_y^0, L_z^0 (*i.e.* of number of vertices $|\mathcal{G}_0^0| = (L_x^0+1) \times (L_y^0+1) \times (L_z^0+1)$, in 2D: $L_z^0 = 0$). We will first assume that L_x^0, L_y^0 and L_z^0 are all powers of 2. Let ϕ_0 be the *triangulation operator*, which transforms $\mathcal{G}^0$ into a valid triangulation $\mathcal{M}^h$, i.e. $\mathcal{M}^h = \phi_0(\mathcal{G}^0)$, by preserving vertex sets, i.e. $\mathcal{M}_0^h = \mathcal{G}_0^0$, and by inserting exactly one edge for each i-dimensional cell of $\mathcal{G}^0$ ($1 < i \leq d$), according to a unique pattern, which is *invariant by translation* along the cells of the grid, known as Kuhn's triangulation [Kuh60]. In 2D, each quadrilateral is subdivided into two triangles by inserting one edge always along the *same diagonal*. In 3D, each hexahedron is subdivided into six tetrahedra by always inserting the *same diagonal* edges (Figure 3.3).

Let Π_1 be the *decimation operator*, which transforms the regular grid $\mathcal{G}^0$ into a regular grid $\mathcal{G}^1$, i.e. $\mathcal{G}^1 = \Pi_1(\mathcal{G}^0)$, by selecting one vertex every

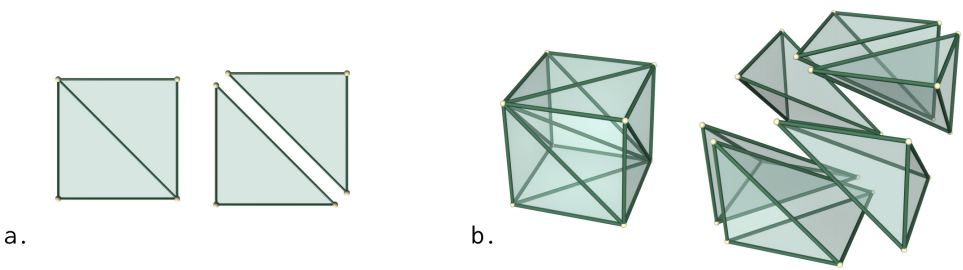


Figure 3.3 – *Translation invariant local triangulation pattern for the cells of a 2D and 3D regular grid. In 2D, quadrilaterals are subdivided into two triangles (a), always along the same diagonal. In 3D, the generalization of this pattern subdivides each hexahedron into six tetrahedra (b).*

$$\begin{tikzcd}
\mathcal{M}^0 \arrow[r] & \mathcal{M}^1 \arrow[r] & \dots \arrow[r] & \mathcal{M}^{h-1} \arrow[r] & \mathcal{M}^h \\
\mathcal{G}^h \arrow[u, "\phi_h"] & \mathcal{G}^{h-1} \arrow[l, "\Pi_h"] \arrow[u, "\phi_{h-1}"] & \dots \arrow[l, "\Pi_{h-1}"] \arrow[u] & \mathcal{G}^1 \arrow[l, "\Pi_2"] \arrow[u, "\phi_1"] & \mathcal{G}^0 \arrow[l, "\Pi_1"] \arrow[u, "\phi_0"]
\end{tikzcd}$$

Figure 3.4 – *Commutative diagram for the generation of an edge-nested triangulation hierarchy $\mathcal{H} = \{\mathcal{M}^0, \mathcal{M}^1, \dots, \mathcal{M}^h\}$ from a regular grid $\mathcal{G}^0$. The hierarchy can be obtained by a sequence of decimation operators Π_i, accompanied with triangulation operators ϕ_i.*

two vertices in each dimension. Let (i, j, k) be the grid coordinates of a vertex $v \in \mathcal{G}^0$. Then the grid $\mathcal{G}^1$ is obtained by only selecting the vertices with even grid coordinates (i, j, k) in $\mathcal{G}^0$. In 2D, each quadrilateral of $\mathcal{G}^1$ corresponds in the general case to four quadrilaterals of $\mathcal{G}^0$ and in 3D, each hexahedron of $\mathcal{G}^1$ corresponds to eight hexahedra of $\mathcal{G}^0$. Note that the decimation operator Π_1 induces a reciprocal *subdivision* operator, which, given $\mathcal{G}^1$, yields $\mathcal{G}^0$ by inserting a new vertex in the center of each i-dimensional cell of $\mathcal{G}^1$ $(0 < i \leq d)$.

We now introduce by recurrence a sequence of decimation operators Π_i (Figure 3.4), which decimate each grid $\mathcal{G}^{i-1}$ into a grid $\mathcal{G}^i$ by subsampling its vertices with even grid coordinates as described above. It follows that for a given level of decimation i, the dimensions of $\mathcal{G}^i$ are given by $L_x^i = L_x^0/2^i$, $L_y^i = L_y^0/2^i$, and $L_z^i = L_z^0/2^i$. Let us now consider the sequence of triangulation operators ϕ_i, which triangulate each grid $\mathcal{G}^i$ into a triangulation $\mathcal{M}^{h-i}$, i.e. $\mathcal{M}^{h-i} = \phi_i(\mathcal{G}^i)$, as illustrated by the commutative diagram of Figure 3.4. Then, it can be verified (Figure 3.5) that each condition of Section 3.2.1 is indeed satisfied by the sequence $\mathcal{H} = \{\mathcal{M}^0, \mathcal{M}^1, \dots, \mathcal{M}^h\}$ and that $\mathcal{H}$ is a valid edge-nested triangulation hierarchy. In particular, as described by Bey [Bey98], any triangulation $\mathcal{M}^i$ can be equivalently obtained either: *(i)* by applying the red subdivision scheme [R.E83, J. 95, S. 95] i times on $\mathcal{M}^0$ or *(ii)* by considering the Kuhn triangulation [Kuh60] of $\mathcal{G}^{h-i}$ (itself obtained by i regular subdivisions of $\mathcal{G}^h$). In other words, any triangulation $\mathcal{M}^i$ in the commutative diagram of Figure 3.4 can be obtained by starting either *(i)* from $\mathcal{M}^0$ or *(ii)* from $\mathcal{G}^h$. In our work, we exploit this equivalence property, but in *reverse*: we use it to efficiently retrieve an arbitrarily coarse version of the fine triangulation $\mathcal{M}^h$ of the input grid $\mathcal{G}^0$.

In particular, the edge-nested triangulation hierarchy $\mathcal{H}$ can be implemented very efficiently, by encoding the entire hierarchy implicitly, and

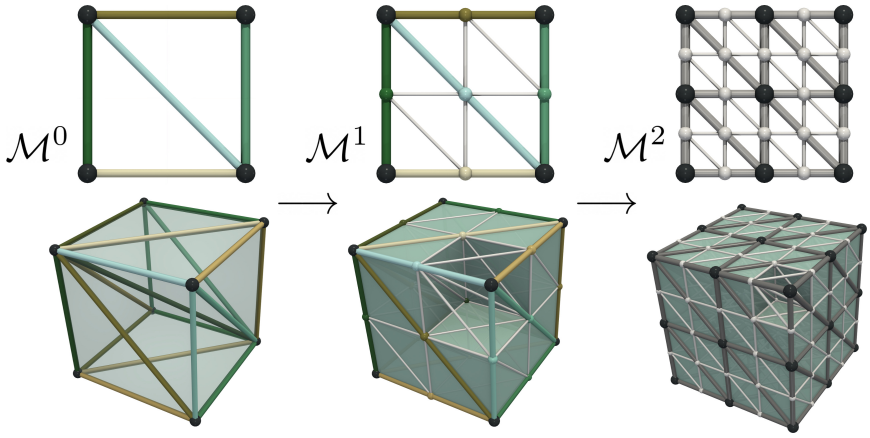


Figure 3.5 – *Edge-nested triangulation hierarchy generated from a regular grid. Old vertices/edges are shown in black/gray in $\mathcal{M}^2$. There is a one-to-one mapping (colors from $\mathcal{M}^0$ to $\mathcal{M}^1$) between the edges of $\mathcal{M}^0$ and the new vertices of $\mathcal{G}^{h-1}$, inserted in each i-dimensional cell of $\mathcal{G}^h$ ($0 < i \leq d$).*

by only maintaining the grid $\mathcal{G}^0$ in memory. At a given hierarchy level i, adjacency relations in $\mathcal{M}^i$ between two vertices v_0 and v_1 can be inferred based on their grid coordinates at level i, (i_0, j_0, k_0) and (i_1, j_1, k_1), and given the triangulation pattern shown in Figure 3.3. Then, the data values associated to the vertices v_0 and v_1 can be retrieved by mapping these vertices back to their original locations in $\mathcal{G}^0$, given by the grid coordinates $(i_0 \times 2^{h-i}, j_0 \times 2^{h-i}, k_0 \times 2^{h-i})$ and $(i_1 \times 2^{h-i}, j_1 \times 2^{h-i}, k_1 \times 2^{h-i})$. This approach is easily extended to support regular grids whose dimensions, L_x^0, L_y^0 or L_z^0 are not necessarily powers of 2. In particular, when considering the decimation operator Π_i, in case some of the dimensions L_x^{i-1}, L_y^{i-1} or L_z^{i-1} are not even, Π_i systematically adds the last vertex of $\mathcal{G}^{i-1}$ for each odd dimension. In our progressive algorithms (Sec. 3.3 and 3.4), these few extra vertices will require full recomputations. Below, we resume our generic description for arbitrary edge-nested triangulation hierarchies, not necessarily obtained from regular grids (Section 3.6 discusses generalizations).

3.2.3 Topologically Invariant Vertices

The input edge-nested triangulation hierarchy $\mathcal{H}$ yields a hierarchy of PL scalar fields $\{f^0, f^1, \ldots, f^h\}$, such that each old vertex v maintains by construction its scalar value: $f^i(v) = f^j(v) = f(v)$, $\forall j \ / \ i \leq j \leq h$. In the following, we show how the specific structure of edge-nested triangulation hierarchies described in Section 3.2.1 can be leveraged to efficiently update topological information while progressing down the hierarchy. First

we show that edge-nested triangulations preserve the topology of the link of vertices when progressing from one hierarchy level to the next. This enables the quick identification, discussed next, of vertices which do not change their criticality when progressing down the hierarchy. We call these vertices *topologically invariant old vertices*, as they will need no update during subsequent analyses (Section 3.3 and Section 3.4). Last, we show how to efficiently identify new vertices that are guaranteed by construction to be regular points of f^i, which we call *topologically invariant new vertices* and for which no computation will be required in subsequent analyses.

Link Topological Invariance:

A first insight is that the link $Lk(v)$ of a vertex v is topologically invariant throughout the hierarchy. This property is important because it will enable the fast identification of vertices which do not change their criticality (next paragraph). Let $Lk(v)^i$ be the link of v at level i, then there exists a one-to-one mapping Ψ_i (Figure 3.6) between the simplices of $Lk(v)^i$ and $Lk(v)^{i+1}$ – such that $Lk(v)^{i+1} = \Psi_i\big(Lk(v)^i\big)$ – which preserves the simplicial structure of $Lk(v)^i$ (*e.g.* the adjacencies). Indeed, *(i)* new vertices are only inserted on old edges (this maps the kth neighbor of v at level i to its kth new neighbor at level $i+1$, top red arrow in Figure 3.6) and *(ii)* new edges are only inserted between new vertices (this maps the kth edge of $Lk(v)^i$ to the kth new edge of $Lk(v)^{i+1}$, right red arrow in Figure 3.6). This mapping Ψ_i can be viewed as a combinatorially invariant *zoom* in the neighborhood of v as one progresses down the hierarchy.

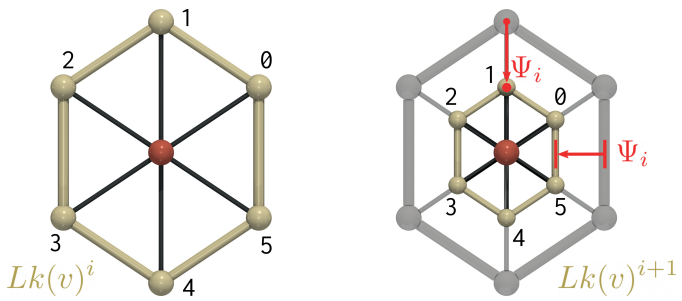


Figure 3.6 – *Link topological invariance of edge-nested triangulations. From one hierarchy level (i) to the next ($i+1$), edge-nested triangulations preserve the local structure of the link $Lk(v)^i$ of an old vertex v (red sphere). In particular, there exists a one-to-one mapping Ψ_i between the vertices and the edges (red arrows) of $Lk(v)^i$ and $Lk(v)^{i+1}$.*

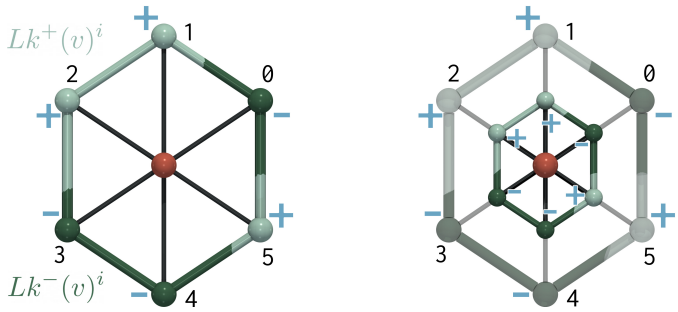


Figure 3.7 – *Topologically Invariant Old Vertex. The link invariance enables the fast identification of old vertices which do not change their criticality: these are old vertices (red sphere) for which the* polarity *(blue signs) remains unchanged from one hierarchy level (i) to the next ($i+1$) and for which, therefore, connected components of lower and upper links (green and blue components, respectively) do not change (thus, requiring no update). Such vertices are called* topologically invariant old vertices

Topologically Invariant Old Vertices

A second insight deals with the evolution of the data values on the link of an *old* vertex, as one progresses down the hierarchy and zooms with the above mapping Ψ_i. We define the *polarity* of $Lk(v)^i$, noted $\delta : Lk(v)^i \rightarrow \{-1, 1\}$ as the field which assigns to each neighbor n of v at level i the sign of its function difference with v: $\delta(n) = sgn\big(f(n) - f(v)\big)$. The polarity is positive in the upper link, negative in the lower link (Figure 3.7, blue signs). Let (v_0, v_1) be an edge at level i, which gets subdivided at level $i+1$ along a new vertex v_n. Assuming that $f(v_0) < f(v_1)$, we say that v_n is *monotonic* if $f(v_n) \in \big(f(v_0), f(v_1)\big)$. Otherwise, v_n is *non-monotonic*. In that case, if v_n's polarity in $Lk(v_0)^{i+1}$ is the opposite of v_1's polarity in $Lk(v_0)^i$, we say that v_0 is *impacted* by its neighbor v_n. Now, if an old vertex v is not *impacted* by any of its non-monotonic neighbors, its link polarity is maintained (i.e. the blue signs in Figure 3.7 remain unchanged when going from the hierarchy level i to $i+1$). This implies that v is therefore guaranteed to maintain its *criticality*: it maintains its critical index (i.e., $\mathcal{I}(v)^{i+1} = \mathcal{I}(v)^i$) or it remains regular. Indeed, each neighbor n which does not impact v maintains its classification as being upper or lower. Then, since there is a one-to-one mapping Ψ_i (see Figure 3.6) between $Lk(v)^i$ and $Lk(v)^{i+1}$ which preserves their simplicial structure, it follows that the complexes $Lk^-(v)^{i+1}$ and $Lk^+(v)^{i+1}$ are respectively identical to $Lk^-(v)^i$ and $Lk^+(v)^i$. Thus, the number of connected components of lower and upper links are maintained, preserving the criticality of v. Old vertices which are not impacted by their non-monotonic neighbors are called *topologically invariant old vertices*.

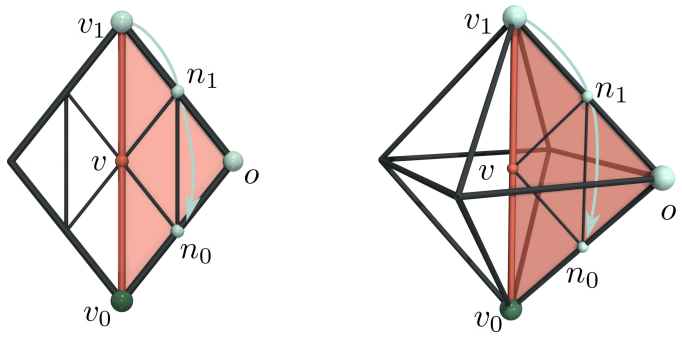


Figure 3.8 – *Topologically Invariant New Vertex, on a 2D (left) and a 3D (right) example. A new vertex v which is* monotonic *(i.e. $f(v_0) < f(v) < f(v_1)$, with v_0 and v_1 being respectively the lowest and highest vertex of the edge (v_0, v_1) where v is inserted) is guaranteed to be regular if all its adjacent new neighbors (in the figure, n_0 and n_1) are also* monotonic. *Such vertices are called* topologically invariant new vertices.

Topologically Invariant New Vertices

A third insight deals with the link of *new* vertices. Given a new monotonic vertex v (small red spheres in Figure 3.8) subdividing an edge (v_0, v_1) at level i (red cylinders in Figure 3.8), if its new neighbors are all monotonic as well, v is then called an *interpolating vertex* and it can be shown that v must be a regular vertex.

First, since v is monotonic, it cannot be an extremum, since by definition it is connected to one lower (v_0) and one upper (v_1) old vertex (large green and blue spheres in Figure 3.8). Note that v_0 and v_1 are the only old vertices adjacent to v. Second, to show that v is regular, we argue that $Lk^+(v)^i$ is necessarily connected (and so is $Lk^-(v)^i$, symmetrically). Let (v_0, v_1, o) be a triangle at level $i-1$ (red triangles in Figure 3.8). At level i, the edges (v_0, o) and (v_1, o) are subdivided along the new vertices n_0 and n_1 and the *new* edges (v, n_0), (v, n_1), and (n_0, n_1) are inserted to connect the new vertices.

Let us assume that $f(n_0) > f(v)$. n_0 is then an *upper* neighbor of v ($n_0 \in Lk^+(v)^i$). Since n_0 is monotonic, this means that the *outer* old vertex o (which is not in $Lk(v)^i$) must also be upper: $f(o) > f(n_0) > f(v)$. Since n_1 is monotonic as well, it follows that n_1 is upper too. Thus, there exists a path $\{v_1, n_1, n_0\} \in Lk(v)^i$ (blue arrow in Figure 3.8), which connects v_1 to n_0 and which is only composed of *upper* vertices. Thus n_0 and v_1 belong to the same connected component of $Lk^+(v)^i$. The same reasoning holds for all the new *upper* neighbors of v. It follows that $Lk^+(v)^i$ and $Lk^-(v)^i$ are both made of a single connected component, containing exactly one old vertex each, v_1 and v_0 respectively. Thus, v is regular. Note that this reasoning readily applies to 2D and 3D, as demonstrated in Figure 3.8.

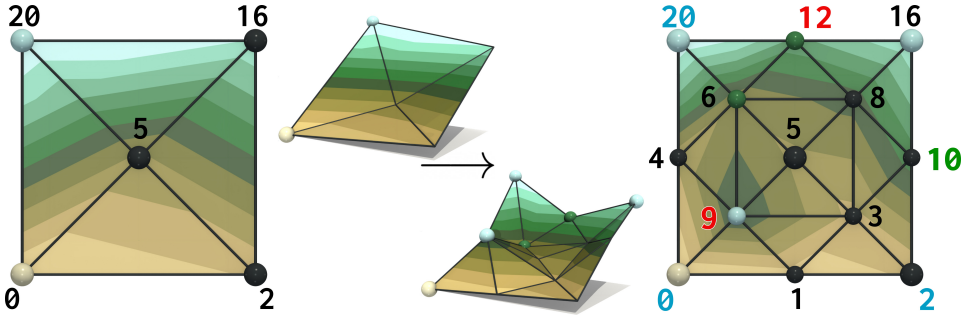


Figure 3.9 – Topologically Invariant *(TI) vertices (numbers denote f values). When progressing down the hierarchy, two* non-monotonic *vertices appear (red labels). This yields new critical points (cyan: maxima, green: saddles, brown: minimum). Old TIs (blue labels), whose link polarity is unchanged, maintain their criticality. New TIs are regular (green label). For* topologically invariant *vertices (blue and green labels), no computation is required. As illustrated in Table 3.1 (page 58), TI vertices represent the majority of the data in real-life datasets.*

Since interpolating vertices, such as v, imply no topological event in the sub-level sets, we call them *topologically invariant new vertices*.

The three key insights of edge-nested triangulations discussed above (summarized in Figure 3.9) form the cornerstone of our progressive approach to topological analysis. As detailed next, checking if vertices are topologically invariant turns out to be less computationally expensive in practice than computing their criticality from scratch. Moreover, the set of topologically invariant vertices tends to represent the majority of the hierarchy (see Section 3.5). This allows for the design of efficient progressive algorithms, presented in the next sections.

3.3 Progressive Critical Points

Our progressive algorithm for critical point extraction starts at the first level of the hierarchy, $\mathcal{M}^0$, and progresses level by level down the hierarchy $\mathcal{H}$ until reaching its final level, $\mathcal{M}^h$, or until interrupted by a user. At each level i, our approach delivers the entire list of critical points of the data for the current resolution ($f^i : \mathcal{M}^i \to \mathbb{R}$). For this, our strategy consists in avoiding recomputation as much as possible and instead efficiently and minimally update the information computed at the previous level $(i-1)$.

3.3.1 Initialization and Updates

This section focuses on the vertices of $\mathcal{H}$ which are not *topologically invariant*. The case of topologically invariant vertices is discussed in Section 3.3.2. In short, our approach computes the criticality of each vertex with the traditional method [Ban70] at the first hierarchy level. However, for the following levels, instead of re-starting this computation from scratch, our algorithm maintains the criticality information computed at the previous levels and only minimally updates this information, where needed, by using dynamic trees [ST83], a specialized data structure for dynamic connectivity tracking.

At the first hierarchy level, $\mathcal{M}^0$ only contains new vertices for which the criticality needs to be initialized. As of the second level, old and new vertices start to co-exist in $\mathcal{M}^1$ and fast update mechanisms can be considered to efficiently update the criticality of the old vertices. For this, we leverage the topological invariance of the link of each old vertex throughout the hierarchy (Section 3.2.3). This allows to store relevant topological information on the link and to quickly update them when progressing down the hierarchy. In particular, we initialize for each *new* vertex v at level i the following information:

- *Link 1-skeleton:* We store the list of *local* edges (and their adjacencies) of $Lk(v)^i$, encoded with pairs of local indices for the neighbors of v. This remains invariant through $\mathcal{H}$ (Section 3.2.3).
- *Link polarity:* We store for each vertex of $Lk(v)^i$ its *polarity* (Section 3.2.3), i.e. its classification as being upper or lower than v. This is encoded with one bit per vertex of $Lk(v)^i$.
- *Link dynamic tree:* An efficient data structure [ST83] for maintaining connected components in dynamic graphs, discussed below.

For each new vertex v which is not topologically invariant, the following data structures are initialized: a list of pairs of local neighbor indices denoting the local edges of $Lk(v)^i$ (up to 24 pairs in a 3D grid), a list of bits denoting the polarity of each neighbor (up to 14 neighbors in a 3D grid), and the dynamic tree, detailed below. The criticality of v is computed with the traditional approach (Section 2.2), by enumerating the connected components of $Lk^+(v)^i$ and $Lk^-(v)^i$. This is usually achieved with breadth-first search traversals or with a Union-Find (UF) data structure [CLRS09]. However, in our setting, we would like to update these connected components as the algorithm progresses down the hierarchy. In particular, if

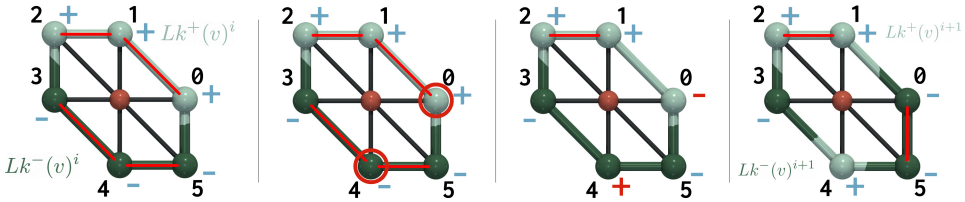


Figure 3.10 – *Updating the criticality of a non-topologically invariant old vertex. From left to right: initial state, identification of non-monotonic vertices (red circles), update of the link polarity (red +/− signs), and update of the connected components of* $Lk^+(v)$ *and* $Lk^-(v)$*. At each step, edges present in the dynamic tree [ST83] are highlighted in red. Only the edges impacted by polarity flips need to be updated in the dynamic tree: edges* $(0,1)$*,* $(3,4)$ *and* $(4,5)$ *are removed, and the edge* $(0,5)$ *is added.*

a *local* edge e belongs to the upper link of v at level i, but not anymore at level $i+1$, the connected components of $Lk^+(v)^{i+1}$ need to be updated accordingly, preferably without recomputing them completely. For this, we use dynamic trees [ST83], which, like the UF data structure, maintain connected components in a graph upon edge insertion, but unlike the UF, also maintain them upon edge removal. In particular, all the vertices of $Lk(v)^i$ are initially inserted in the dynamic tree associated to v. Next, we insert each local edge of $Lk(v)^i$ in the dynamic tree, if both its ends have the same polarity. The criticality of v is then deduced by enumerating the connected components with positive and negative polarity, thanks to the dynamic tree.

For each old vertex v which is not topologically invariant (Figure 3.10), its link polarity is quickly updated based on the non-monotonic new vertices of $Lk(v)^i$. Each local edge e of $Lk(v)^i$ which is *impacted* by a polarity flip of its vertices (Section 3.2.3) is removed from the dynamic tree associated to v if it was present in it (to account for the corresponding disconnection of lower/upper link component), and added to it otherwise, if both its ends have the same polarity (if they belong to the same lower/upper link component). Then, the criticality of v is quickly updated with the fast enumeration of the connected components of positive and negative polarity provided by the dynamic tree. Note that such an efficicent update of the criticality of v would not be feasible with a simple UF data structure, as the connected components of the link of v would need to be recomputed from scratch upon edge removal.

3.3.2 Computation Shortcuts

When moving from the hierarchy level i to $i+1$, topologically invariant old vertices are guaranteed to maintain their criticality (Section 3.2.3). For these, the dynamic trees (Section 3.3.1) do not need to be updated. Moreover, when moving from the hierarchy level i to $i+1$, each topologically invariant new vertex v is guaranteed to be regular. For these, the dynamic trees (Section 3.3.1) are not even initialized (they will only be used when v becomes no longer topologically invariant). Overall, our procedure to update vertex criticality can be summarized as follows:
1) Mononotic vertices: in this step, we loop over all new vertices to check whether or not they are monotonic.
2) Link polarity: in this step, we loop over all vertices to initialize/update their link polarity. For old vertices, updates are only needed for their non-mononotic neighbors. If an old vertex v is topologically invariant, no more computation is required for it at this hierarchy level.
3) Old vertices: each old vertex v which is not topologically invariant efficiently updates its criticality in f^i as described in Section 3.3.1.
4) New vertices: if a new vertex v is topologically invariant, it is classified as regular and no more computation is required for it at this hierarchy level. Otherwise, its criticality is updated (Section 3.3.1).

3.3.3 Parallelism

Critical point computation is an operation which is local to the link of each vertex. Thus, each of the four steps introduced above can be trivially parallelized over the vertices of $\mathcal{M}^i$ with shared-memory parallelism. This implies no synchronization, at the exception of the sequential transition between two consecutive steps.

3.3.4 Extremum Lifetime

As our algorithm progresses down $\mathcal{H}$, the population of critical points evolves. In practice, this means that some features of interest may be captured by the progressive algorithm earlier than others, denoting their importance in the data. To evaluate this, we consider for each extremum e the notion of *Lifetime*, defined as $l(e) = l_d(e) - l_a(e)$, where $l_a(e)$ and $l_d(e)$ stand for the levels where e appeared and disappeared respectively. The evaluation of this measure requires a correspondence between the extrema computed at the levels i and $i+1$, which is in general a challenging

assignment optimization problem [JS06, KE07, BWT+11, RKWH12, SW17, SPCT18c]. For simplicity, we focus here on a simple yet time-efficient heuristic for estimating these correspondences, which can be enabled optionally.

Given a vertex v, identified as maximum at hierarchy level $i-1$, our heuristic consists of computing, for each neighbor n of v, a forward integral line $\mathcal{L}^+(n)^i$ (see definition 2.24, page 22). Each of these lines terminates on local maxima of f^i, which we add to the set of *candidates* for v. At the end of this step, we establish the correspondence between v and its highest candidate in terms of f^i values, noted m^*, and we say that v *maps* to m^* from $i-1$ to i. To focus the integration on a reasonable neighborhood, we restrict the number of edges on each integral line to a user parameter L_{max}, set to 10 in our experiments. If the set of *candidates* of v is empty, the maximum present in v at level $i-1$ is considered to disappear at level i $\left(l_d(v) = i\right)$. It is possible, given a maximum m at level i, that no maximum from the level $i-1$ maps to it. In this case, m is said to appear at the level i $\left(l_a(m) = i\right)$. Finally, if multiple maxima at level $i-1$ map to the same maximum at level i, they are all considered to disappear at the level i, at the exception of the *oldest* maximum (minimizing l_a), as suggested by the *Elder* rule in the case of persistence [EH09]. This optional procedure is run at each hierarchy level and enables the progressive estimation of the lifetime of the maxima. Note that the lifetime of minima is estimated with the symmetric procedure.

3.4 Progressive Persistence Diagrams

Our approach for progressive persistence diagrams leverages and combines the insights and algorithms introduced in the previous sections. It starts at the coarsest hierarchy level, $\mathcal{M}^0$, and then iterates progressively through the hierarchy levels, producing the exact persistence diagram $\mathcal{D}(f^i)$ for each level i, until $i = h$. We first introduce our approach in the non-progressive case (Section 3.4.1, Figure 3.12), and then present our progressive strategy (Section 3.4.2). We focus on minimum-saddle pairs, saddle-maximum pairs being treated symmetrically (in which case, each boundary component is considered as a virtual maximum).

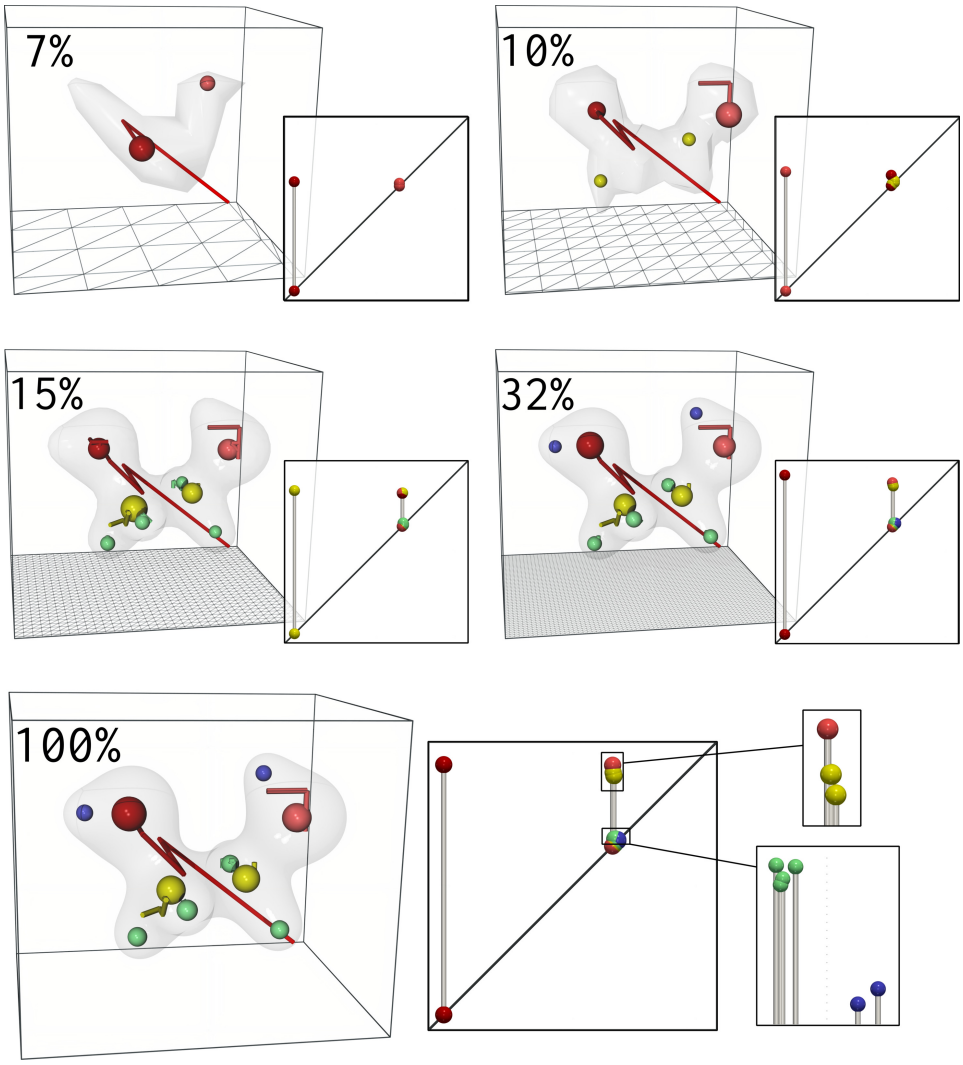


Figure 3.11 – *Progressive persistence diagrams (saddle-maximum pairs, left to right) for the electron density of the ethane-diol molecule (transparent isosurface), at a few steps of the progressive computation. Maxima (denoting the atoms) are shown in the domain with spheres, scaled by persistence and colored by lifetime (red to blue), while their trajectory through the data hierarchy (Section 3.3.4) is shown with a curve (matching color). Our progressive approach captures the heaviest atoms first: two oxygens (at* 7% *of the computation time), then two carbons* (10%) *and finally the six hydrogens* (32%).

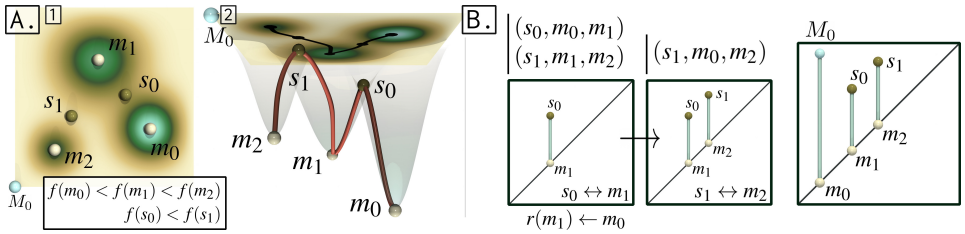


Figure 3.12 – *Computing the minimum-saddle persistence diagram from critical points. Downwards monotonic paths are initiated at saddles to extract a list of critical point triplets (part A, left), which forms a reduced topological representation of the data. This reduced reprensentation is efficiently processed to produce the persistence diagram (part B, right).*

3.4.1 Persistence Diagram from Critical Points

The diagram $\mathcal{D}(f)$ of the extremum-saddle pairs of an input field $f : \mathcal{M} \to \mathbb{R}$ is computed as follows. In short, critical points are used as seeds for the computation of monotonic paths, specifically linking saddles down to minima. This first step identifies merge events occurring at saddle points (*part A*). The merge events are processed in a second step (*part B*) to track the connected components of sub-level sets. Similarly to previous topological techniques based on monotonic paths [CLLR05, MDN12, CWSA16, SM17], our approach emulates the usage of a Union-Find data structure with path compression [CLRS09] (traditionally used for connectivity tracking) by propagating *representants* between merged components. However our strategy is specialized for the production of persistence diagrams, and only visits monotonic paths of minimal length (i.e. integral lines).

Part A:

From data to reduced topological information

1) Critical points. First, critical points are extracted (Section 3.3).

2) Saddle monotonic paths. The second step consists in initiating monotonic paths from each saddle s downwards, to identify at least one minimum for each connected component of sub-level set merging at s (Figure 3.12). For this, we initiate backward integral lines (see definition 2.24, page 22), for each connected component of lower link $Lk^-(s)$ of each saddle s. These integral lines are guaranteed to terminate in local minima of f. Once a backward integral line $\mathcal{L}^-(s)$ terminates in a local minimum m, we back-propagate the vertex identifier of m and store it for each vertex $v \in \mathcal{L}^-(s)$. Then, m is called a *representant* of v, which is noted $r(v) = \{m\}$. This strategy enables the early termination of an integral line $\mathcal{L}^-(s_1)$ when

it merges with another one, $\mathcal{L}^-(s_0)$, computed previously. In that case, we back-propagate the representants reported by the merge vertex on $\mathcal{L}^-(s_1)$ back to s_1. At the end of this step, each saddle s is associated with the list of representants collected by its backward integral lines. These denote local minima which may have initially created the sub-level set components merging at s.

Part B:
From reduced topological information to persistence diagrams

3) Critical triplets. For each saddle s, we create a list of *critical triplets*, in the form (s, m_0, m_1), where m_0 and m_1 are representants of s and thus are local minima. These are obtained by considering pairs among the set of representants of s (computed previously). Note that in practice, for nearly all saddles, this list consists of only one triplet, which describes the fact that s separates two pits, m_0 and m_1. Note that in case of degenerate saddles, multiple triplets emerge. For a degenerate saddle associated with d representants $(m_0, \ldots, m_{d-1})$ in ascending values of f, we create the $d-1$ triplets (s, m_0, m_i) with $0 < i < d$.

4) Critical point pairing. This step iterates over the global list of critical triplets (computed previously) in increasing order of saddle values. The first triplet (s_0, m_0, m_1) represents the earliest merge event between connected components of sub-level sets of f. We introduce its *simplified version*, $\big(s_0, r(m_0), r(m_1)\big)$, which is initially equal to (s_0, m_0, m_1) (initially, a local minimum is itself its own representant). The highest of the two minima, for instance m_1, is then selected to create in $\mathcal{D}(f)$ the critical point pair (s_0, m_1). Indeed, since s_0 is the earliest merge event, m_1 is guaranteed to be the *youngest* minimum, according to the Elder rule [EH09], which created a component of sub-level set merging with another one at s_0. To model the *death* of m_1's component (its merge with the component containing m_0), we update its representant as follows: $r(m_1) \leftarrow r(m_0)$. Thus, all future merging events involving m_1 will re-direct to m_0, as the component born at m_1 died by merging with that containing m_0 (following the Elder rule [EH09]). This simplification process is iterated over the (sorted) global list of critical triplets. At each step, when constructing a simplified triplet $\big(s, r(m_0), r(m_1)\big)$, we recursively retrieve the representants of $r(m_0)$ and $r(m_1)$, until we reach minima only representing themselves. This guarantees that for each merge event of the sub-level set occurring at a saddle s, we can efficiently retrieve the deepest minimum for each of the components merging in s and therefore pair it adequately in $\mathcal{D}(f)$. Note

that the recursive update of representants is equivalent to the so-called *path compression* of UF data structures [CLRS09]. Overall, iterating as described above over the list of triplets results in populating $\mathcal{D}(f)$ with pairs from bottom to top (by increasing death values).

3.4.2 Progressive Strategy

The above algorithm is divided in two parts (A and B, Section 3.4.1). In particular, only part A can leverage our progressive representation of the input data (Section 3.2), as part B processes reduced topological information which has been abstracted from it and which therefore become completely independent. Thus, we focus our progressive strategy on part A. This has a negligible impact on practical performance. In our experience, part B represents less than 5% of the computation on average. Critical points (Step 1) can be extracted progressively as described in Section 3.3. For Step 2, we investigated multiple shortcut mechanisms (similar to Section 3.3.2), to maintain the monotonic paths which remain valid from level i to $i+1$. However, from our experience, the overhead induced by this global maintenance is not compensated by the acceleration it induces at level $i+1$, as monotonic paths are usually highly localized and thus already inexpensive to compute (less than 10% of the non-progressive computation on average). Thus, our overall strategy for progressive persistence diagrams simply consists, at each level i of the triangulation hierarchy $\mathcal{H}$, in updating progressively the critical points (Section 3.3) and then triggering the fast, remaining steps of persistence diagram computation (2, 3, 4) as described in Section 3.4.1.

3.4.3 Parallelism

Our progressive algorithm for persistence diagram computation can be easily parallelized. The initial critical point computation (Step 1, Section 3.4.1) is parallelized as described in Section 3.3.3. Saddle integration (Step 2, Section 3.4.1) can be trivially parallelized over saddles. However, locks need to be used during representant back propagation (to guarantee consistency over concurrent accesses by distinct monotonic paths). Critical triplet generation (Step 3, Section 3.4.1) is also parallelized over saddles. In Step 4 (critical point pairing Section 3.4.1), triplets are sorted in parallel using the efficient GNU implementation [SK08]. The reminder of Step 4 is intrinsically sequential (as representants need to be updated in order

of simplification), but in practice, this step represents less than 1% of the sequential execution, which does not impact parallel efficiency.

3.5 Results

This section presents experimental results obtained on a computer with two Xeon CPUs (3.0 GHz, 2x4 cores, 64GB of RAM), with a C++ implementation of our algorithms (publicly available at: https://github.com/julesvidal/progressive-scalar-topology), written as modules for the Topology ToolKit (TTK) [TFL+17]. The datasets are 3-dimensional (at the exception of *SeaSurfaceHeight*, which is 2-dimensional) and they have been downloaded from public repositories [Kla20, TTK20]. Appendix A provides a detailed list of these datasets, with information about the considered scalar field and the related features of interest.

3.5.1 Progressive Data Representation

In this section, we study the practical relevance of our progressive data representation (Section 3.2). First, we evaluate its qualitative relevance. Our approach for persistence diagram computation (Section 3.4) progressively refines an estimation of the output $\mathcal{D}(f)$, by efficiently updating $\mathcal{D}(f^i)$ at each new hierarchy level i. To evaluate quantitatively the relevance of this estimation $\mathcal{D}(f^i))$, we measure its similarity to the final, exact result $\mathcal{D}(f)$ with the Wasserstein distance (Section 2.4).

For each level i, we measure the L_2-Wasserstein distance $W_2(\mathcal{D}(f), \mathcal{D}(f^i))$. We normalize this distance by dividing it by $W_2(\mathcal{D}(f), \emptyset)$. Then, along the hierarchy $\mathcal{H}$, this normalized distance progresses from 1 to 0 for all datasets. Although this distance may increase in theory from one level to the next, Figure 3.13 shows that it is monotonically decreasing for our datasets.

This shows that in practice, the accuracy of our progressive outputs indeed improves over time. This empirical convergence evaluation gives a global picture of the quality of our progressive data representation. To further evaluate its relevance, we report in Figure 3.14 the ratio of captured *significant pairs* in the diagram $\mathcal{D}(f^i)$ as a function of the computation time. To evaluate this ratio, we select the *significant* pairs of $\mathcal{D}(f)$, i.e. with a relative persistence greater than 0.1. Let n_p be the number of such significant pairs (reported for each dataset in the legend of Figure 3.14, right, along with its percentage over the total number of pairs

in $\mathcal{D}(f)$, in parenthesis). Next, we select the n_p most persistent pairs in $\mathcal{D}(f^i)$ and divide the resulting number of selected pairs, noted $n_p^i \leq n_p$, by n_p. In short, this indicator helps appreciate the number of significant features captured by the hierarchy early in the computation. In particular, Figure 3.14 shows that for most of the datasets, the number of captured significant pairs matches the final estimation as of 10% of the computation time. Figure 3.15 reports the average persistence of the n_p^i significant pairs in $\mathcal{D}(f^i)$ as a function of the computation time, relatively to the average persistence of the n_p significant pairs in $\mathcal{D}(f)$. This indicator helps appreciate how well the significant pairs are captured in the data hierarchy. In particular, this figure shows a clear global trend across datasets: the persistence of the significant pairs tends to be underestimated early in the computation and this estimation improves over time. These quantitative observations (early capture of the significant pairs and underestimation of persistence at the beginning of the computation) can be visually observed in Figure 3.11, which shows that the significant pairs are captured early in the data hierarchy (red and yellow pairs) but that their persistence is indeed underestimated: the corresponding points are initially close to the diagonal in the corresponding diagrams and then, they progressively move away from it.

Figure 3.16 (top) presents additional convergence results on an extensive list of real-life datasets: all the datasets from the *Open Scientific Visualization Dataset* repository [Kla20] which fit in the main memory of our experimental setup. This represents a set of 36 datasets (containing acquired and simulated data). As discussed above, while the monotonic

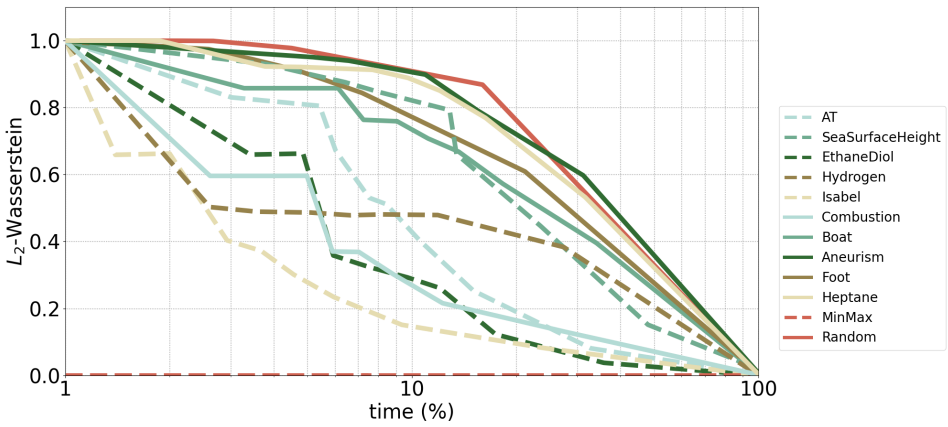


Figure 3.13 – *Empirical convergence of the normalized L_2-Wasserstein distance. Each curve plots the distance between the currently estimated diagram, $\mathcal{D}(f^i)$, and the final, exact diagram, $\mathcal{D}(f)$, as a function of the percentage of computation time (logarithmic scale).*

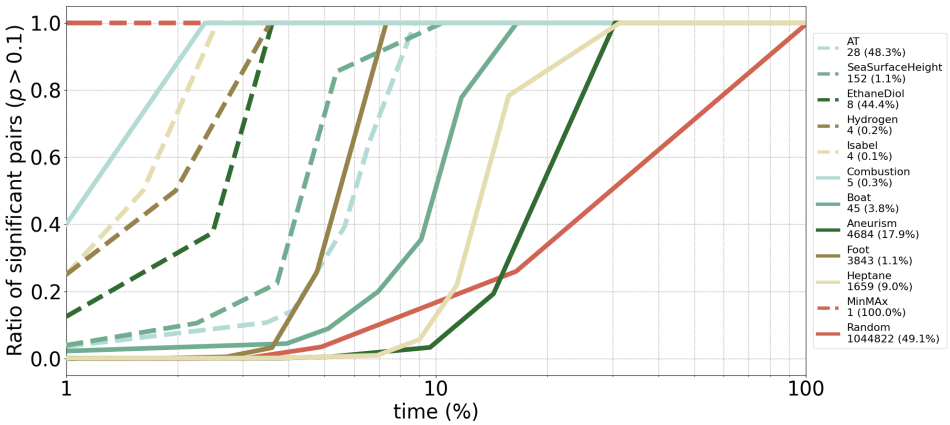


Figure 3.14 – *Ratio of captured* significant pairs *in* $\mathcal{D}(f^i)$ *(cf. Section 3.5.1) as a function of computation time (logarithmic scale).*

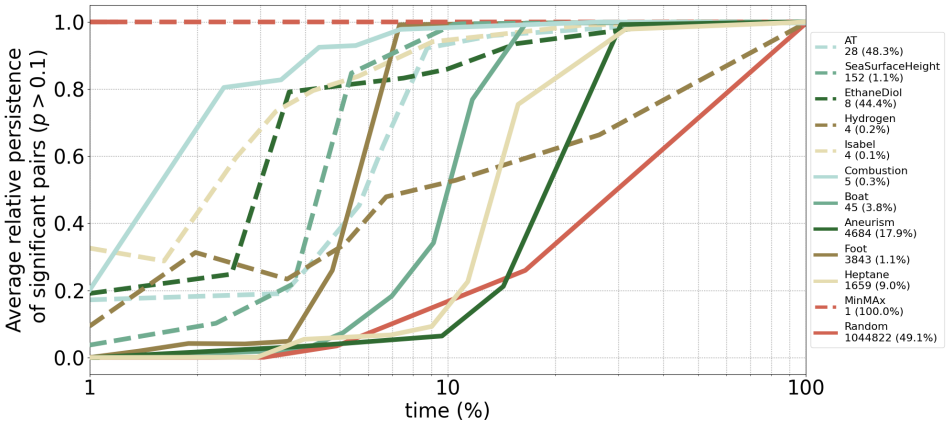


Figure 3.15 – *Average persistence of the* significant pairs *captured in* $\mathcal{D}(f^i)$ *(cf. Section 3.5.1), relatively to the same average in* $\mathcal{D}(f)$*, as a function of computation time (logarithmic scale).*

decrease of the L_2-Wasserstein distance along the computation cannot be guaranteed at a theoretical level, it can still be observed in practice. Note however that for two examples (*Bunny* and *Engine*) a slight oscillation can be observed in the early stages of the computation (between 1% and 2% of the computation time). However, past this point, the distance keeps on decreasing monotonically. Also, note that the convergence curves for all datasets are indeed located between the curves of the two extreme synthetic examples (*MinMax* and *Random*). The bottom part of Figure 3.16, which reports the average distance for all datasets and its standard deviation, further confirms the overall convergence tendency.

Next, we evaluate the computational relevance of our progressive data representation, by reporting the number of Topologically Invariant (TI) vertices (Section 3.2.3), for which no computation is needed. Table 3.1 shows that for real-world datasets, TI vertices represent 72% of the data

Table 3.1 – *Statistics of our progressive data hierarchy. From left to right: number of vertices, number of levels, memory overhead (over TTK), and number of topologically invariant (TI) vertices (Section 3.2.3) in the data hierarchy. For real-world datasets (Random and MinMax excluded), topologically invariant vertices represent 72% of the data on average.*

Dataset	$\sum_{i=0}^{i=h} \lvert\mathcal{M}_0^i\rvert$	h	M. (Mb)	# old TIs	# new TIs	Total TIs	% **TIs**
AT	931,110	9	242	93,145	509,504	602,649	**64.7%**
SeaSurfaceHeight (2D)	1,384,626	11	240	241,264	608,460	849,724	**61.4%**
EthaneDiol	2,057,388	9	527	227,376	1,371,537	1,598,913	**77.7%**
Hydrogen	2,413,532	8	626	245,541	1,528,607	1,774,148	**73.5%**
Isabel	3,605,604	9	970	274,590	1,329,116	1,603,706	**44.5%**
Combustion	4,378,386	9	1,149	421,208	2,445,192	2,866,400	**65.5%**
Boat	4,821,326	9	1,221	575,646	3,690,624	4,266,270	**88.5%**
Random	18,117,518	9	5,389	169,603	54	169,657	**0.9%**
MinMax	18,994,899	9	4,742	2,394,619	16,474,429	18,869,048	**99.3%**
Aneurism	19,240,277	9	4,841	2,367,190	16,027,209	18,394,399	**95.6%**
Foot	19,240,277	9	5,109	1,648,823	10,746,624	12,395,447	**64.4%**
Heptane	31,580,914	10	8,117	3,453,180	22,512,477	25,965,657	**82.2%**

on average, which indicates that efficient update mechanisms can indeed be derived from our progressive data representation. This table also includes the memory overhead induced in progressive mode by the data structures employed by our topological analysis algorithms (Section 2.2 and Section 2.3). In particular, this overhead is estimated by measuring the memory footprint of all the data-structures which are present in our progressive algorithms (Section 3.3 and Section 3.4) but *not* present in the TTK implementation of the state-of-the-art methods. Thus, this column depicts the additional memory needed by our approach in comparison to the standard procedures available in TTK. In particular, this column shows a linear evolution of this memory overhead with the size of the data hierarchy. Note that our implementation is not optimized for memory usage and that important gains can be expected by re-engineering our data structures at a low level.

3.5.2 Time Performance

The time complexity of our progressive algorithm for critical point extraction is linear with the number of input vertices, which results in our hierarchical setup in $\mathcal{O}(\sum_{i=0}^{i=h} \lvert\mathcal{M}_0^i\rvert)$ steps. For persistence diagrams, in the worst possible configuration (degenerate saddles with systematic integral line forking), each saddle would generate monotonic paths which would hit every minimum. This would yield $n_s \times (n_m - 1)$ critical triplets, where n_s and n_m stand for the number of saddles and minima of f. This

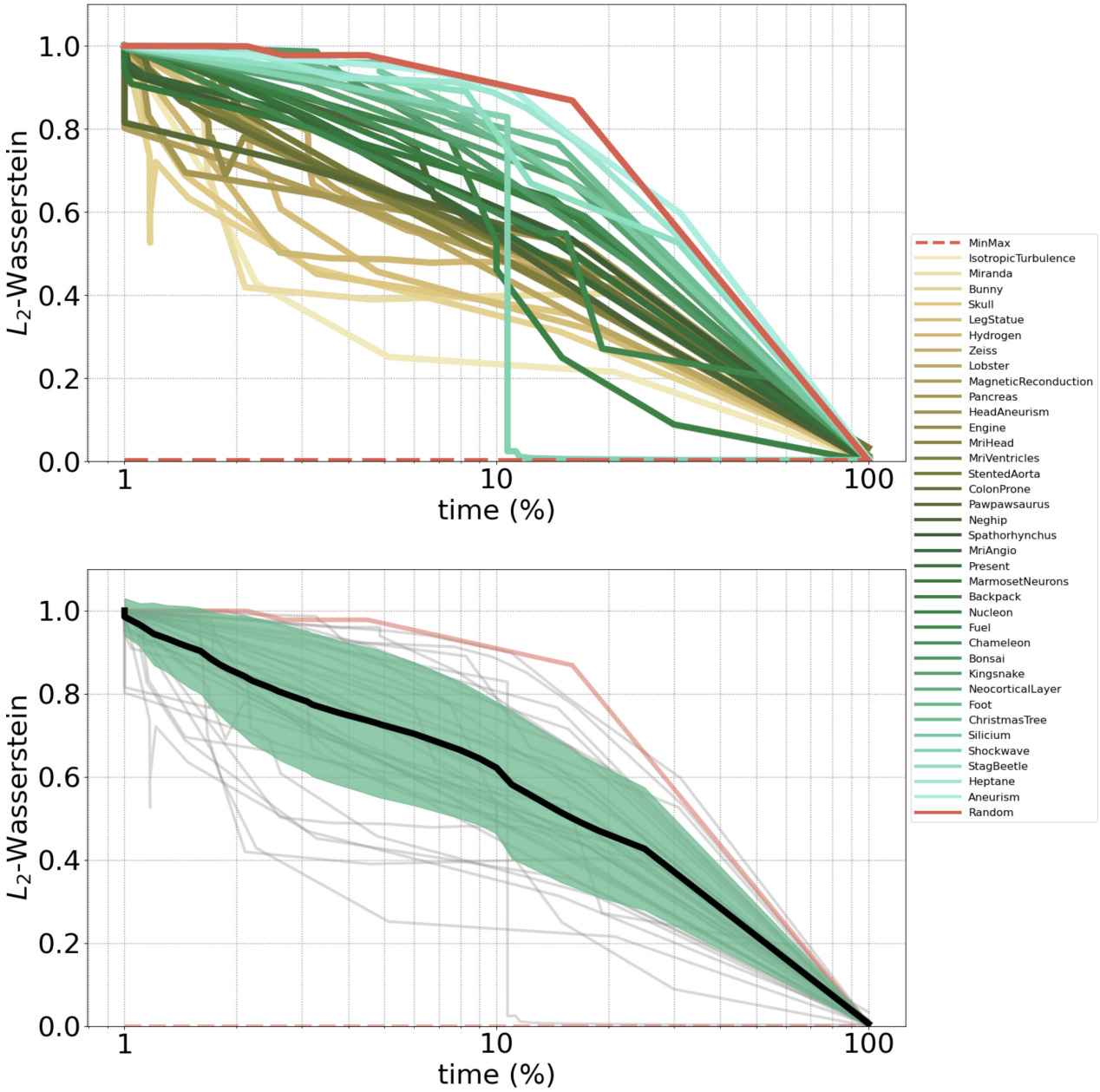


Figure 3.16 – *Empirical convergence of the normalized L_2-Wasserstein distance for an extensive list of datasets.* ***Top****: each curve plots the distance between the currently estimated diagram, $\mathcal{D}(f_i)$, and the final, exact diagram, $\mathcal{D}(f)$, as a function of the percentage of computation times (logarithmic scale). The color map indicates the average distance, from light brown (small distances, fast convergence) to light green (large distances, slow convergence). Extreme synthetic cases are reported in red (dash:* MinMax*, solid:* Random*).* ***Bottom****: Average normalized L_2-Wasserstein distance (black curve) and standard deviation (green hull) for all real-life datasets (i.e. Random and MinMax excluded) as a function of the percentage of computation time. Per-dataset curves are shown in the background (red: synthetic extreme cases, grey: other datasets).*

Table 3.2 – *Sequential computation times (in seconds) of our algorithms for critical point extraction (left) and persistence diagram computation (right). The columns* TTK *report the run times of the default implementations provided by the Topology ToolKit [TFL+17]. The columns* NP *and* Prog *respectively report the timings for the non-progressive (directly initialized at the final hierarchy level) and progressive versions of our algorithms.*

	Critical Points				Persistence Diagram			
Dataset	TTK [Ban70]	NP	Prog	Speedup	TTK [GFJT19]	NP	Prog	Speedup
AT	4.41	0.34	0.25	1.36	0.66	0.31	0.24	1.29
SeaSurfaceHeight (2D)	0.92	0.16	0.26	0.62	0.70	0.24	0.40	0.60
EthaneDiol	9.56	0.73	0.42	1.74	1.45	0.68	0.41	1.66
Hydrogen	11.55	0.92	0.59	1.56	1.90	0.89	0.63	1.41
Isabel	17.98	1.40	1.43	0.98	2.76	1.50	1.62	0.93
Combustion	21.87	1.74	1.33	1.31	4.36	1.82	1.50	1.21
Boat	22.47	1.74	0.73	2.38	3.38	1.81	0.82	2.21
Random	113.12	13.99	21.04	0.66	74.39	25.65	34.40	0.75
MinMax	82.23	6.94	1.56	4.45	14.68	7.00	1.64	4.27
Aneurism	84.03	7.39	2.19	3.37	12.85	8.03	3.43	2.34
Foot	100.58	9.26	8.13	1.14	18.20	12.18	12.06	1.01
Heptane	149.30	12.43	6.46	1.92	19.45	13.22	8.16	1.62

would yield $n_s \times (n_m - 1)$ merge events for the critical pairing step, each with an amortized complexity of $\mathcal{O}\big(\alpha(n_m)\big)$, where α is the inverse of the Ackermann function. However, such configurations are extremely rare in practice and most saddles only yield one triplet, resulting in an overall practical time complexity of $\mathcal{O}\big(\sum_{i=0}^{i=h}(|\mathcal{M}_1^i| + n_s^i \log n_s^i + n_s^i \alpha(n_m^i))\big)$ steps, also accounting for the sorting of triplets.

Table 3.2 reports computation times (sequential run) for the default algorithms (Section 2.2) [Ban70], [GFJT19] available in TTK [TFL+17] and the non-progressive and progressive versions of our algorithms. Non-progressive methods (*TTK* and *NP* columns) compute from scratch only the last hierarchy level h directly. We only report the run times of TTK as an indicative baseline as the differences in triangulation implementations already induce alone important run time variations (TTK emulates implicitly triangulations for regular grids *at query time*, while our implementation stores the explicit list of link edges for each vertex, Section 3.3.1). Interestingly, the *Speedup* columns show that, in addition to their ability to provide continuous visual feedback, our progressive algorithms are also faster than their non-progressive versions (on average, 1.8 times faster for critical points, 1.6 for persistence diagrams). These speedups confirm that the overhead of processing an entire hierarchy ($\sum_{i=0}^{i=h} |\mathcal{M}_0^i|$ vertices in progressive mode, instead of $|\mathcal{M}_0^h|$ in non-progressive mode) and of detecting TI vertices is largely compensated by the gains these vertices provide. Specifically, *old* topologically invariant vertices enable efficient updates between levels (Section 3.3.2), which allows to avoid losing time

Table 3.3 – *Parallel computation times (in seconds, 8 cores) of our algorithms for critical point extraction (left) and persistence diagram computation (right). The columns* TTK *report the run times of the default implementations provided by the Topology ToolKit [TFL+17]. The columns* NP *and* Prog *respectively report the timings for the non-progressive (directly initialized at the final hierarchy level) and progressive versions of our algorithms.*

	Critical Points				Persistence Diagram			
Dataset	TTK [Ban70]	NP	Prog	Speedup	TTK [GFJT19]	NP	Prog	Speedup
AT	0.54	0.06	0.05	1.20	0.29	0.06	0.06	1.00
SeaSurfaceHeight (2D)	0.15	0.04	0.05	0.80	0.28	0.07	0.11	0.64
EthaneDiol	1.18	0.12	0.07	1.71	0.42	0.13	0.11	1.18
Hydrogen	1.41	0.14	0.12	1.17	0.97	0.18	0.19	0.95
Isabel	2.11	0.21	0.21	1.00	0.89	0.25	0.29	0.86
Combustion	2.54	0.25	0.21	1.19	0.82	0.29	0.30	0.97
Boat	2.71	0.26	0.15	1.73	0.81	0.30	0.24	1.25
Random	11.46	1.82	2.67	0.68	19.10	8.96	10.89	0.82
MinMax	9.87	1.25	0.43	2.91	3.24	1.39	0.69	2.01
Aneurism	10.19	1.14	0.50	2.28	4.08	1.89	1.48	1.28
Foot	11.30	1.93	1.75	1.10	5.39	3.37	3.29	1.02
Heptane	17.72	2.38	1.51	1.58	5.74	2.79	2.50	1.12

in comparison to a non-progressive approach. In contrast, *new* topologically invariant vertices enable actual shortcuts in the computation (Section 3.3.2), which allows to *gain* time in comparison to a non-progressive approach. Note that the datasets with the most (resp. least) TI vertices (Table 3.1) are also those for which the largest (resp. smallest) speedups are obtained, confirming the importance of TI vertices in the computation.

Table 3.3 details the performance of the shared-memory parallelization of our progressive algorithms, using OpenMP [DM98], again in comparison to the default algorithms (Section 2.2) [Ban70], [GFJT19] available in TTK [TFL+17] and to the non-progressive version of our algorithms. As mentioned in Section 3.3.3, critical point extraction can be trivially parallelized over vertices, for each of the four steps of our algorithm, resulting in an average parallel efficiency of 66%. The persistence diagram computation results in a more modest efficiency (45%) as monotonic path computations are subject to locks, in addition to be possibly imbalanced.

3.5.3 Stress Cases

Our experiments include two synthetic datasets, whose purpose is to illustrate the most and the least favorable configurations for our approach, to better appreciate the dependence of our algorithms to their inputs. The *MinMax* dataset is an elevation field which only contains one global minimum and one global maximum. It exhibits therefore a lot of regularity. In

contrast, the *Random* dataset assigns a random value to each vertex. Thus, no local coherency can be expected between consecutive levels in the data hierarchy (which is an important hypothesis in our framework).

Table 3.1 confirms the best/worst case aspect of these datasets, as they respectively maximize and minimize the ratio of TI vertices: *MinMax* has nearly only TI vertices (99.3%) while *Random* has nearly none (0.9%).

Table 3.2 confirms, as can be expected, that these two datasets also maximize and minimize the speedup induced by our progressive approach. In particular, our progressive algorithms report a speedup greater than 4 over their non-progressive versions for the *MinMax* dataset. This further confirms the observation made in Section 3.5.2 that processing an entire data hierarchy with the acceleration induced by TI vertices can indeed be faster than computing criticality from scratch at the final hierarchy level only (in particular, up to 4 times). In contrast, this table also shows that in the worst possible case (*Random*, nearly no TI vertices), the processing of the entire hierarchy can be up to 50% slower (for critical points, 30% for persistence diagrams) than computing in non-progressive mode at the final hierarchy level only. All the other datasets exhibit speedups included within these lower (*Random*) and upper (*MinMax*) bounds (on average 1.8 for critical points, 1.6 for persistence diagrams).

In terms of quality, the best/worst case aspect of *MinMax* and *Random* is also illustrated in Figs. 3.13, 3.14 and 3.15, where *MinMax* converges immediately, while *Random* describes the worst case (slow convergence, slow and inaccurate capture of the significant pairs). In these curves, the other datasets cover the span of possible behaviors between these two extreme cases.

3.5.4 Progressive Topological Visualization and Analysis

This section discusses the progressive visualizations and analyses enabled by our approach. Figure 3.1 presents a typical example of progressive persistence diagram computation on the electron density of the adenine-thymine (AT) molecular system. In this figure, the estimated diagrams progressively capture the features in a meaningful way, as the heaviest atoms are captured first and the lighest ones last. In particular, in the diagrams, the introduced points progressively stand out from the diagonal towards their final locations. As of 33% of the computation, the diagram is complete and its accuracy is further improved over time. This illustrates the capacity of our approach to deliver relevant previews of the topolog-

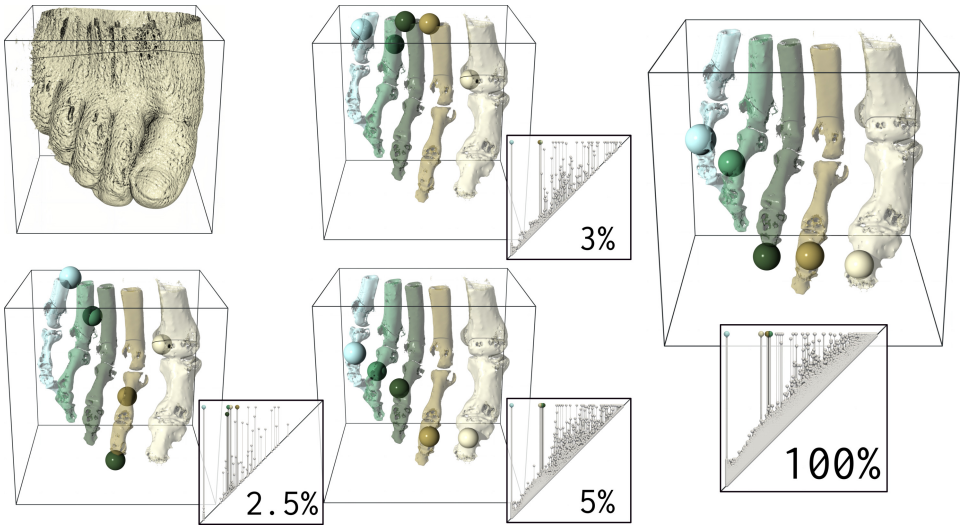


Figure 3.17 – *Progressive persistence diagrams (saddle-maximum pairs, from left to right and top to bottom) of the CT scan of a foot (top, leftmost: isosurface), at a few steps of the computation. A merge tree based segmentation (colored regions, computed with TTK [TFL+17]) reveals the* 5 *most persistence structures in the data. Colored spheres show the* 5 *most persistent maxima reported by the current diagram estimation, illustrating a correct capture of the main structures early in the computation (as of* 3% *of computation).*

ical features of a dataset and to improve them progressively. Figure 3.11 further illustrates our estimation of the lifetime of extrema and their trajectory in the data hierarchy. There, as one progresses down the hierarchy, the prominent maxima are progressively captured and they quickly stabilize in the vicinity of their final location. Figure 3.17 illustrates progressive persistence diagrams for an acquired dataset. There, a merge tree based segmentation (computed with TTK [TFL+17]) is shown in the background. It represents the regions of the five most persistent leaf arcs of the merge tree. The five most persistent maxima reported by the current diagram estimation are reported with spheres. As of 3% of the computation, these maxima are correctly assigned to the final structures (one per toe), while their positional accuracy is further improved with time. Overall, the diagrams (bottom) capture the main features early in the computation, while smaller features and noise are progressively captured as the computation unfolds.

Figure 3.18 presents a gallery of progressive persistence diagrams for several datasets. The diagram estimations capture well the overall shape of the final, exact output (i.e. the number and salience of its main features) and are progressively refined over time. This gallery complements the quantitative analysis reported in Figs. 3.13, 3.14, 3.15 and confirms

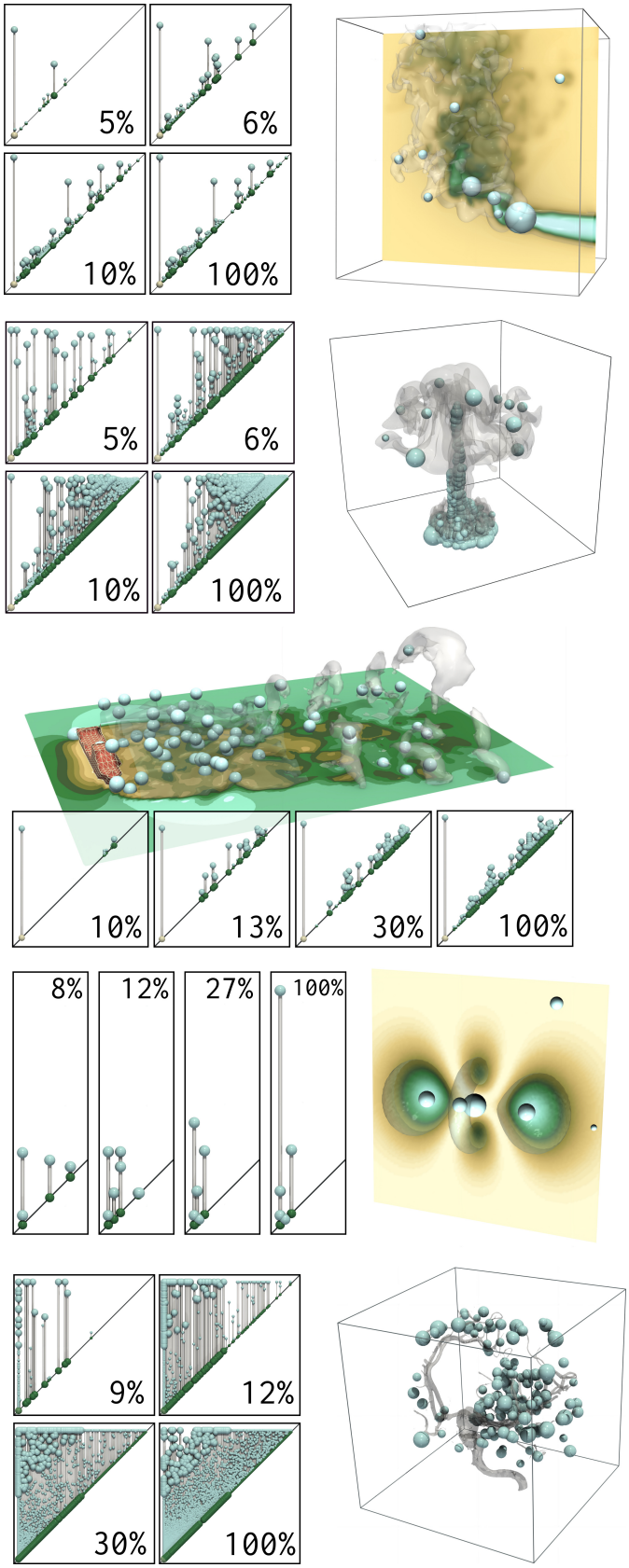


Figure 3.18 – *Progressive persistence diagrams (saddle-maximum pairs) for several data sets (combustion, heptane, boat, hydrogen, aneurism), at a few steps of the computation. Persistent maxima are represented with spheres in the domain (scaled by persistence). The progressive diagrams capture well the overall shape (number and salience of features) of the final, exact output* (100%) *early in the computation and refine it over time.*

visually the interest of our progressive representations, which provide relevant previews of the topological features present in a dataset.

We used the TTK library [TFL+17] to integrate our implementation within the popular visualization system ParaView [AGL05], as shown in the companion video (featured on the Github repository: https://github.com/julesvidal/progressive-scalar-topology). This video illustrates the progressive updates of our topological previews within interactive times, and further demonstrates their interest for interactive visualization environments.

3.6 Limitations and Discussion

Our progressive persistence diagrams tend in practice to capture the main features first. However, this cannot be guaranteed theoretically. For instance, sharp spikes in the data (e.g. high amplitude *and* high frequency noise) can yield persistent maxima only at the last levels of the hierarchy, as illustrated in Figure 3.18 where the global maximum of the *Hydrogen* dataset (fourth row) belongs to a sharp spike in the center of the data (as also reported by the quantitative plots Figs. 3.13 and 3.15). This behavior prevents the definition of theoretical error bounds on our estimations. However, the empirical monotonic decrease of the Wasserstein distance (Figure 3.13) indicates that our progressive representations actually provide reliable estimations, as confirmed by the indicators of Figs. 3.14, 3.15, where the real-world datasets cover the span of possible behaviors between the two stress cases (*MinMax*, *Random*). This can be explained by the fact that, in practice, persistent pairs often coincide with large features in the domain, which get captured early in the data hierarchy.

Although we described our approach generically, we focused in this chapter on an efficient implementation of edge-nested triangulations for regular grids (Section 3.2.2). The generalization of our approach to generic domains requires to investigate triangulation subdivision schemes. Several of them seem compliant with the notion of edge-nested triangulation (Section 3.2.1), such as the Loop subdivision [Loo87] and the *red* triangulation refinement [H. 42, R.E83, J. 95, S. 95]. However, efficiently transforming an arbitrary triangulation into a triangulation which admits an edge-nested hierarchy is an orthogonal question which we leave for future work. Similarly, the reliable tracking of extrema through the hierarchy (for lifetime estimation, Section 3.3.4) relates to another orthogonal problem, for which computationally expensive optimizations may need to

be considered. Our algorithms complete a given hierarchical level before moving on to the next one. This results in increasing update times as the computation converges. In the future, finer update strategies will be considered, by considering adaptive, feature-centric, variable level-of-detail refinement methods. Finally, our algorithm for persistence diagrams does not support saddle-saddle pairs in 3D. However, from our experience, the interpretation of these structures is not obvious in the applications.

Our progressive scheme seems to be particularly efficient for algorithms which visit *all* the vertices of the domain (e.g. critical point extraction), but less beneficial for inexpensive operations which only visit small portions of the data (e.g. integral line computation, Section 3.4.2). This is a lesson learned from our experiments which could serve as guideline for future extensions to other topological analyis algorithms. Also, there is a trade off between the benefits of the progressive scheme and its cost in terms of memory usage. Future work is needed to improve the memory footprint of our approach by optimizing our data structures at a low level. For instance, for triangulations of regular grids and real-life tetrahedral meshes, the maximum number of neighbors around a vertex is typically small, which enables the encoding of local neighbor identifiers with very few bits, instead of full integers (as done in our current implementation). Other variables (such as the polarity, currently stored with a boolean for each neighbor) could also benefit from a more compact bit representation.

3.7 Summary

This chapter introduced an approach for the progressive topological analysis of scalar data. Our work is based on a hierarchical representation of the input data and the fast identification of *topologically invariant vertices*, for which we showed that no computation was required as they were introduced in the hierarchy. This enables the definition of efficient coarse-to-fine topological algorithms, capable of providing interpretable outputs upon interruption requests, and of progressively refining them otherwise until the final, exact output. We instantiated our approach with two examples of topological algorithms (critical point extraction and persistence diagram computation), which leverage efficient update mechanisms for ordinary vertices and avoid computation for the topologically invariant ones. For real-life datasets, our algorithms tend to first capture the most important features of the data and to progressively refine their estimations with time. This is confirmed quantitatively with the empirical convergence

of the Wasserstein distance to the final, exact output, which is monotonically decreasing. More computation time indeed results in more accuracy. Our experiments also reveal that our progressive computations even turn out to be faster overall than non-progressive algorithms and that they can be further accelerated with shared-memory parallelism. We showed the interest of our approach for interactive data exploration, where our algorithms provide progressive previews, continuously refined over time, of the topological features found in a dataset.

4 Approximation of Persistence Diagrams with Guarantees

Contents

Our contribution in one image 71

4.1 Overview 72

4.2 Topology Approximation 72

4.2.1 Hierarchy Processing 73

4.2.2 Vertex Folding 74

4.2.3 Bottleneck Error Control 76

4.2.4 Monotony offsets 77

4.2.5 Parallelism 79

4.2.6 Uncertainty 79

4.3 Results 80

4.3.1 Time Performance 80

4.3.2 Approximation Accuracy 83

4.3.3 Qualitative Analysis 85

4.4 Limitations and Discussion 87

4.5 Summary 88

This chapter presents an algorithm for the efficient approximation of the saddle-extremum persistence diagram of a scalar field. In the previous chapter, we introduced a fast algorithm for such an approximation, by interrupting our progressive computation framework (Section 3.4). However, no theoretical guarantee was provided regarding its approximation quality. In this chapter, we revisit our progressive framework and we introduce in contrast a novel approximation algorithm, with a user controlled approximation error, specifically, on the Bottleneck distance to the exact solution. Our approach is based on a hierarchical representation of the input data, and relies on local simplifications of the scalar field to accelerate the computation, while maintaining a controlled bound on the output error. The locality of our approach enables further speedups thanks to shared memory parallelism. Experiments conducted on real life datasets show that for a mild error tolerance (5% relative Bottleneck distance), our approach improves runtime performance by 18% on average (and up to 48% on large, noisy datasets) in comparison to standard, exact, publicly available implementations. In addition to the strong guarantees on its approximation error, we show that our algorithm also provides in practice outputs which are on average 5 times more accurate (in terms of the L_2-Wasserstein distance) than a naive approximation baseline. We illustrate the utility of our approach for interactive data exploration and we document visualization strategies for conveying the uncertainty related to our approximations.

The work presented in this chapter has been published as part of the proceedings of the IEEE Symposium on Large Data Analysis and Visualization in 2021 [VT21]. It is replicable, using the code and data available at https://github.com/julesvidal/persistence-diagram-approximation. It will soon be integrated into the Topology ToolKit [TFL+17].

Our contribution in one image

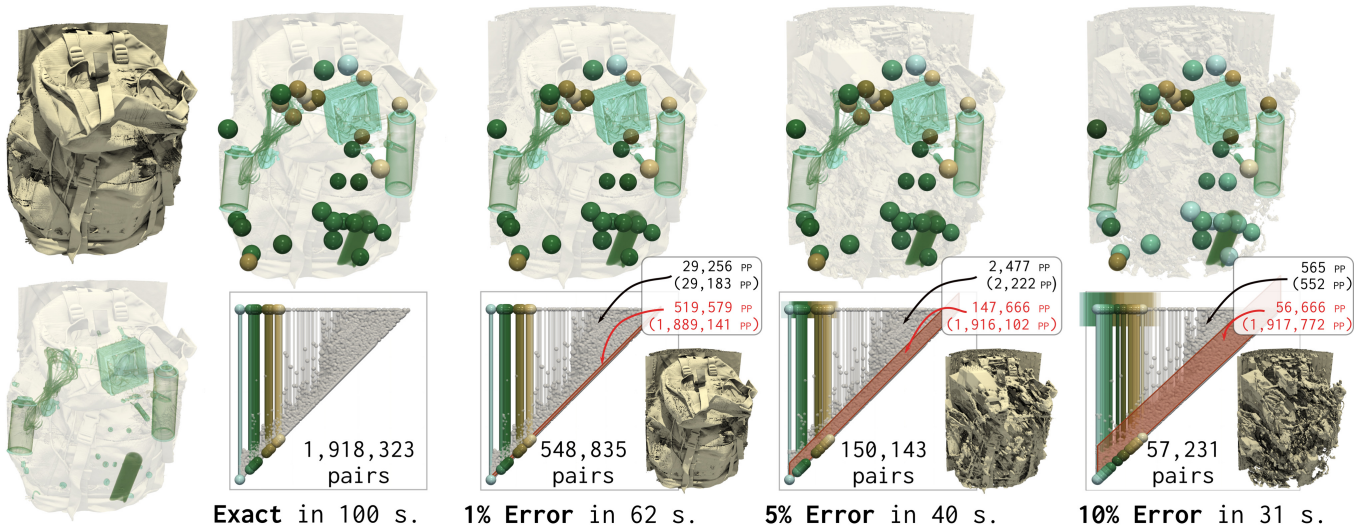


Figure 4.1 – *Approximations of persistence diagrams for a CT scan of a backpack, with different Bottleneck approximation errors. High persistence features correspond to high density objects present in the bag (leftmost, bottom). In this example, our controlled approximation reduces computation time by 70% for an error tolerance of 10%. Our algorithm also provides an approximation of the scalar field that is precise around persistent features (top, volume rendering of the approximated fields) and deteriorated elsewhere (bottom, isocontours capturing the cloth of the bag). For each approximation, the thirty most persistent maxima are represented by spheres (top views) which correctly capture the features of the data. The uncertainty resulting from our approximation is visualized in the diagram with* (i) *colored squares, which bound the correct location of* certain *features (which are guaranteed to be present in the exact result). These are located outside a* (ii) *red band (in the vicinity of the diagonal), which denotes features which* might *not exist in the exact diagram. Our approximations precisely capture the high persistence features of the data (out of the red band, black numbers) and collect significantly less noisy features (red numbers, red band) than the exact result (numbers in parentheses).*

4.1 Overview

Figure 4.2 provides an overview of our approach, which revisits the progressive framework of Chapter 3 to derive a fast approximation algorithm with strong guarantees. First, we exploit the same multiresolution hierarchy of the input data (Section 3.2) to quickly update, down to the finest hierarchy level, the polarity of each vertex (used to identify critical points). This step is described in Section 4.2.1. During the hierarchy traversal, in contrast to the original approach, we artificially increase the number of *topologically invariant vertices* (Section 3.2.3) in order to significantly speedup the computation, through a procedure called *vertex folding*, which artificially degrades the input data. This step is described in Section 4.2.2. The data degradation induced by the vertex folding procedure is precisely controlled in the process, to provide strong guarantees on the approximation error of the output. This is described in Section 4.2.3. Finally, we describe in Section 4.2.4 how to handle degenerate configurations such as flat plateaus. Overall, the approach described in this chapter involves three major differences to the progressive framework presented in Chapter 3, which are detailed in the rest of this section:

1. Our new approximation algorithm is *not* progressive: it does not generate a sequence of progressively refined outputs. Instead, our traversal of the multiresolution hierarchy only updates a minimal amount of information (the vertex polarity). The criticality of each vertex is only evaluated *after* the hierarchy traversal is finished (while the criticality is updated at each hierarchy level in Chapter 3).

2. Our approximation approach is based on a multiresolution degradation of the input data, which accelerates the overall computation, while maintaining a controlled output error.

3. Overall, in contrast to the progressively interrupted results presented in Chapter 3, our approximated output is provided with strong guarantees on its approximation error (in terms of relative Bottleneck distance) and with a significantly improved practical accuracy (in terms of the L_2 Wasserstein distance).

4.2 Topology Approximation

This section presents our novel approach for the controlled approximation of an extremum/saddle persistence diagram. In the following, we focus

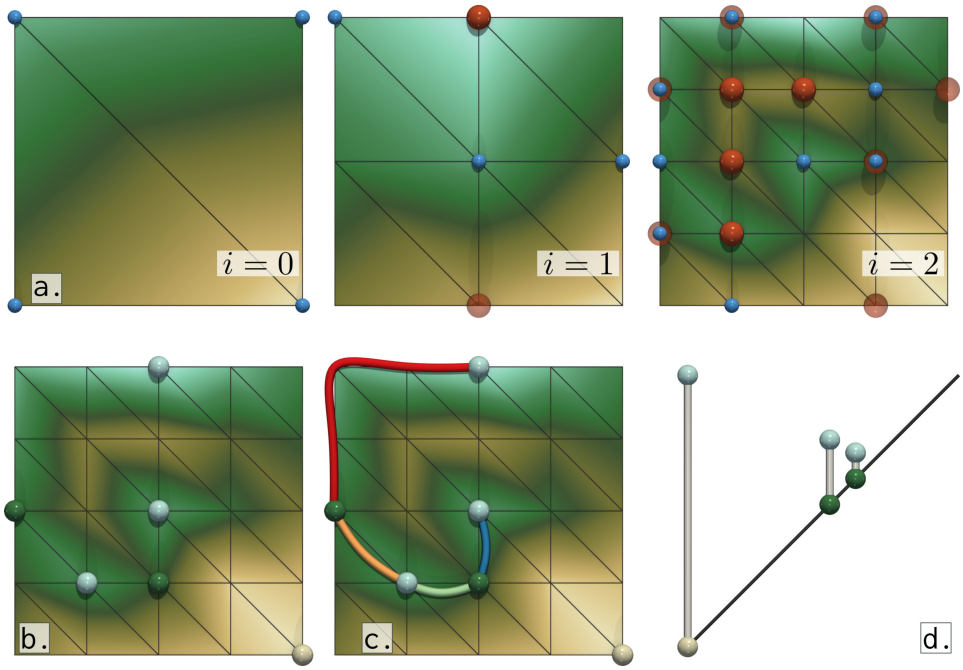


Figure 4.2 – *Overview of our approach. First, the traversal of the hierarchy (a) enables the efficient detection of the vertices (blue spheres) that are* ***not*** *topologically invariant (TI), and for which the criticality must be computed. Non-monotonic vertices (red spheres) can be folded (transparent red spheres) during the traversal, i.e. reinterpolated to artificially increase the number of TI vertices. Second (b), the criticality of all non-TI vertices detected in the first step is computed, to identify the critical points of the approximated field (spheres, cyan: maxima, green: saddles, beige: minimum). Third, the saddle points are used to seed integral lines ending at extrema (c), from which the persistence diagram is deduced (d).*

on the case of minimum/1-saddle pairs ($(d-1)$-saddle/maximum pairs being treated symmetrically). Our method is based on *folding* operations for non-monotonic vertices. As the hierarchy is processed, non-monotonic vertices are inserted at each level. When a non-monotonic vertex is inserted in the hierarchy on a new edge, we can purposely decide to reinterpolate this vertex to enforce its monotony, and hence accelerate the computation of its criticality. The resulting error is the difference between the real scalar value at this vertex and the interpolated value, which bounds the Bottleneck error on the diagram estimation, as detailed next.

4.2.1 Hierarchy Processing

This section presents our computation strategy in the case where the approximation error ε is set to 0 (i.e. exact computations). Given an input edge-nested triangulation hierarchy $\mathcal{H} = \{\mathcal{M}^0, \mathcal{M}^1, \ldots, \mathcal{M}^h\}$, persistence diagrams are evaluated in three steps. First, the hierarchy is completely

traversed, from the coarsest level $\mathcal{M}^0$ to the finest $\mathcal{M}^h$, to efficiently detect topologically invariant vertices, for which the criticality can be efficiently estimated in a second step, at the last level of the hierarchy only ($\mathcal{M}^h$). Third, the persistence diagram is computed from the saddle points identified at the second step.

1) Hierarchy traversal: In order to identify topologically invariant vertices, we compute the link polarity (Section 3.2.3) of the vertices for each level of the hierarchy. The link polarity of a vertex v is encoded in our setting as an array of bits, one per neighbor n of v, denoting whether n is higher or lower than v. The size of the link polarity is the same for each level of $\mathcal{H}$ (Section 3.2.3): at most 6 bits in 2D and 14 bits in 3D. At each level i, the polarity of new vertices is initialized, while the polarity of old vertices is updated to account for the insertion of non-monotonic vertices. The detection of non-monotonic vertices enables the fast identification of regular points: *new* topologically invariant vertices are known to be regular points of f^i, and *old* topologically invariant vertices keep the same criticality at level i than at level $i-1$ (Section 3.2.3). We leverage these informations to avoid the explicit computation of the criticality for some of the vertices. As a new topologically invariant vertex v is identified in $\mathcal{M}^i$, it is flagged as a regular vertex. If its link polarity is not changed in the remaining levels (e.g. v is topologically invariant through the rest of the hierarchy), v is guaranteed to be a regular vertex of $\mathcal{M}^h$ and no further computation will be needed for it.

2) Critical points: At $\mathcal{M}^h$, we compute explicitly the criticality of the vertices which are not yet guaranteed to be regular (as described above). The criticality of a vertex v is computed by enumerating the connected components of $Lk^+(v)$ and $Lk^-(v)$ (Section 2.2).

3) Persistence diagram: The minimum/1-saddle persistence diagram is deduced from the critical points, as presented in steps 2, 3 and 4 of Section 3.4.1.

4.2.2 Vertex Folding

This section describes the variations of the above strategy for the case where the user-controlled approximation error ε is not zero in order to decrease the computational cost. Specifically, we present a strategy to artificially increase the number of topologically invariant vertices during the hierarchy traversal (step 1), which will consequently result in skipping

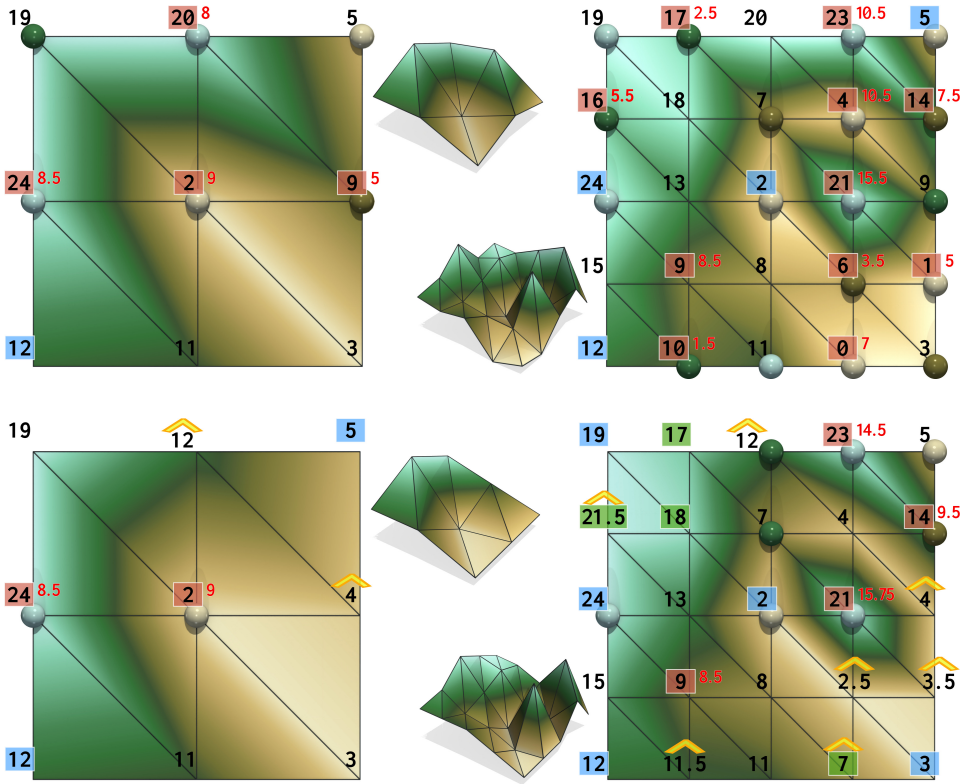


Figure 4.3 – *Traversal of the hierarchy and identification of topological invariants vertices for two different folding threshold (top: $\varepsilon = 0$, bottom $\varepsilon = 0.3$). The numbers denote the values of the approximated scalar field $\widehat{f}$. Red squares indicate the non-monotonic new vertices, whose folding error δ_ε is labelled in red. At the top, no vertex is folded and no approximation is made on the scalar field (i.e. $\widehat{f} = f$). It results in a high number of non-monotonic vertices and a low number of topologically invariant old and new vertices (respectively blue and green squares). In contrast, a folding threshold $\varepsilon = 0.3$ is applied at the bottom. Every non-monotonic vertex n with a folding error $\delta(n) \leq 8$ gets folded (yellow hats). This reduces the number of non-monotonic vertices and more than doubles the number of TI vertices (blue and green squares) of the full precision approach.*

the estimation of vertex criticality (step 2) for a larger number of vertices (hence the overall speedup).

A new vertex n, appearing at a level i of the hierarchy, is inserted at the center of an old edge (o_0, o_1) of $\mathcal{M}^{i-1}$ (Section 3.2.1). Assuming that $f(o_0) < f(o_1) < f(n)$, n is a non-monotonic vertex and impacts the link polarity of o_1. The apparition of such vertices reduces the overall performance as these will trigger explicit criticality computations in step 2. To reduce the number of non-monotonic vertices inserted in $\mathcal{M}^i$, we can choose to reinterpolate a non-monotonic vertex n between its two new neighbors to enforce its monotony. The method that we use to decide which vertex to reinterpolate is detailed in Section 4.2.3. We note the resulting monotonic vertex $\widehat{n}$ and say that this vertex is *folded*. We define

its new approximated value $\widehat{f}(\widehat{n})$ as the interpolation of the approximated values of its two old neighbors: $\widehat{f}(\widehat{n}) = \left(\widehat{f}(o_0) + \widehat{f}(o_1)\right)/2$. The values $\widehat{f}(o_0)$ and $\widehat{f}(o_1)$ are themselves either the result of a linear interpolation if o_0 or o_1 have been previously *folded*. Otherwise, they are equal to $f(o_0)$ and $f(o_1)$.

Formally, we build a sequence $\{\widehat{f}^0, \widehat{f}^1, \ldots, \widehat{f}^h\}$ of PL scalar fields defined on each hierarchy level. The sequence is defined recursively:

1. $\widehat{f}^0 = f^0$
2. For each old vertex o of $\mathcal{M}^i$, $\quad \widehat{f}^i(o) = \widehat{f}^{i-1}(o)$
3. For each new vertex n of $\mathcal{M}^i$ that is not *folded*, $\quad \widehat{f}^i(n) = f^i(n)$
4. For each *folded* new vertex $\widehat{n}$ of $\mathcal{M}^i$ on the edge (o_0, o_1),

$$\widehat{f}^i(\widehat{n}) = \frac{\widehat{f^{i-1}}(o_0) + \widehat{f^{i-1}}(o_1)}{2}$$

We note $\mathcal{F}^i$ the set of folded vertices at level i (with $\mathcal{F}^i \subset \mathcal{M}^i_0$). By construction, folded new vertices are monotonic. Figure 4.3 illustrates how a higher amount of folded vertices at level i implies a higher number of topologically invariant vertices identified on $\mathcal{M}^i$. In particular, if all non-monotonic vertices are folded at level i, all vertices of $\mathcal{M}^i$ are topologically invariant.

4.2.3 Bottleneck Error Control

Computing persistence diagrams with *vertex folding* results in approximations of the exact result $\mathcal{D}(f)$, given by $\mathcal{D}(\widehat{f})$ (diagram of the approximated field $\widehat{f}$). Then the resulting approximation error (in terms of Bottleneck distance) is given by [CEH05]: $W_\infty\big(\mathcal{D}(\widehat{f}), \mathcal{D}(f)\big) \leq \|\widehat{f} - f\|_\infty$, which is rather easy to estimate. For each new vertex n inserted at level i of the hierarchy on the edge (o_0, o_1), we define its *folding error* $\delta(n)$ as the difference between its original scalar value and its reinterpolation value at its level of insertion: $\delta(n) = \left|\left(\widehat{f}^i(o_0) + \widehat{f}^i(o_1)\right)/2 - f(n)\right|$. Then, we have:

$$\|\widehat{f} - f\|_\infty = \max_{\widehat{n} \in \mathcal{F}^h} \delta(\widehat{n}) = \max_{\widehat{n} \in \mathcal{F}^h} |\widehat{f}(\widehat{n}) - f(\widehat{n})| \tag{4.1}$$

In the light of these observations, we use the following folding strategy as we process the hierarchy. Given a target approximation error ε, the hierarchy is processed as described in Section 4.2.1, except that vertex polarity is estimated from the approximated field $\widehat{f}$. For each level i, we choose to fold non-monotonic vertices n with an error $\delta(n) < \varepsilon$. Monotonic vertices or non-monotonic vertices with a higher folding error get added into the

hierarchy without being reinterpolated. Let $\widehat{f}_\varepsilon : \mathcal{M} \to \mathbb{R}$ be the final field approximation (after the hierarchy traversal is completed), given the target approximation error ε. Then it is clear that $\forall v \in \mathcal{M},\ \delta(v) \leq \varepsilon$. Thus:

$$W_\infty\big(\mathcal{D}(\widehat{f}_\varepsilon), \mathcal{D}(f)\big) \leq \|\widehat{f}_\varepsilon - f\|_\infty \leq \varepsilon$$

In the remainder, ε is given as a percentage of the data range ($\varepsilon = 0$ is the exact computation, while $\varepsilon = 1 = 100\%$ folds all non-monotonic vertices). The range of the scalar field also corresponds to the worst L_∞ error we can make on the approximated field, which bounds the worst Bottleneck error that our approach can yield. Consequently, we use the expression of ε as a percentage of the scalar range $r = \max f - \min f$ to indicate the level of our approximation, relatively to this worst Bottleneck error. As such, $\varepsilon = 0.05 = 5\%$ corresponds to a relative maximal error of 5% on the Bottleneck distance to the exact result.

4.2.4 Monotony offsets

We now discuss how to handle degenerate flat plateaus, which become more frequent given our vertex folding strategy. The input scalar field f is injective on the vertices of $\mathcal{M}$ (Section 4.1). This is enforced in practice with an offset field $\mathbb{O} : \mathcal{M}_0 \to \mathbb{N}$, typically corresponding for each vertex to its offset in memory (Figure 4.4a). $\mathbb{O}$ is then used to disambiguate vertices with identical scalar values.

In theory, a *folded* vertex $\widehat{n}$ is guaranteed to be monotonic. However in practice, if $\delta(\widehat{n})$ falls below the precision of the data type used to encode the scalar field, a flat plateau emerges. This occurs frequently for instance when the input data is expressed with integers. Then, given a new folded vertex $\widehat{n}$ inserted on an edge (o_0, o_1), we may have: $\widehat{f}(\widehat{n}) = \widehat{f}(o_0)$, which means $\widehat{n}$ and o_0 will be disambiguated in the algorithm by their offset $\mathbb{O}$. However, this can introduce undesired monotony changes (Figure 4.4b, red squares).

To guarantee the monotony of folded vertices, we introduce a *monotony offset* on each vertex v, noted $\mathbb{M}(v)$, which is modified when the vertex gets folded. The purpose of the monotony offset $\mathbb{M}$ is to take over the regular offset $\mathbb{O}$ if it contradicts the monotony of newly folded vertices. Given a new *folded* vertex $\widehat{n}$ inserted on an edge (o_0, o_1) at level i, that is non-monotonic with respect to o_0 (i.e. $\widehat{f}(\widehat{n}) < \widehat{f}(o_0) < \widehat{f}(o_1)$ or $\widehat{f}(\widehat{n}) > \widehat{f}(o_0) >$

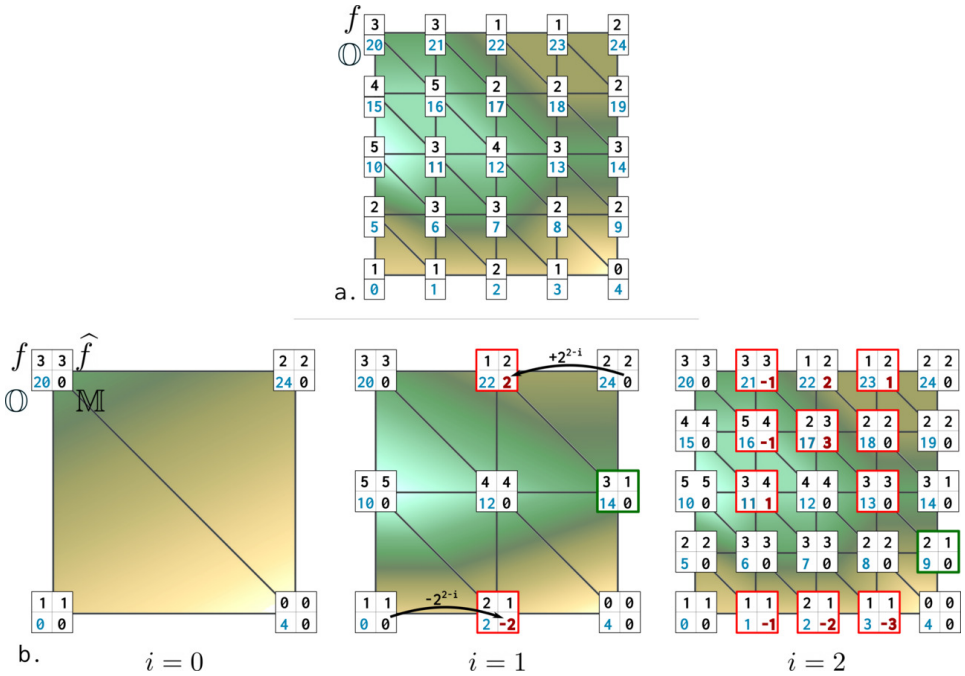


Figure 4.4 – *Monotony offsets on a toy example of an* ***integer*** *field f (a: black numbers) defined on a 2D grid. The injectivity of f is guaranteed by the offset field $\mathbb{O}$ (a: blue numbers). On the bottom row (b), the matrices indicate for each vertex the values of the fields $f, \widehat{f}, \mathbb{O}$ and $\mathbb{M}$. As the hierarchy is processed (b), some vertices get folded (red and green squares) according to a given error threshold. Due to the precision of the field (here integer precision), their interpolated value (numbers on the right) might be identical to that of an old neighbor. In some cases (red squares), the folding actually entails a monotony change as the offset field $\mathbb{O}$ (blue numbers) provides the wrong order relation between neighbor vertices. If such an event occurs, the monotony offset of the folded vertex is updated (red numbers) to enforce monotony. Green squares denote folding cases where $\mathbb{O}$ does not contradict the folding monotony. At the finest resolution (b, rightmost), this results in plateaus (bottom row of the grid) where the injectivity of $\widehat{f}$ is guaranteed and where the monotony is correctly enforced.*

$\widehat{f}(o_1)$), we set:

$$\mathbb{M}(\widehat{n}) = \begin{cases} \mathbb{M}(o_0) - 2^{h-i} & \text{if } \widehat{f}(\widehat{n}) < \widehat{f}(o_0) < \widehat{f}(o_1) \\ & \text{and } \mathbb{O}(\widehat{n}) > \mathbb{O}(o_0) \\ \mathbb{M}(o_0) + 2^{h-i} & \text{if } \widehat{f}(\widehat{n}) < \widehat{f}(o_0) < \widehat{f}(o_1) \\ & \text{and } \mathbb{O}(\widehat{n}) < \mathbb{O}(o_0) \\ \mathbb{M}(o_0) & \text{else} \end{cases}$$

The monotony offset is initially set to zero for all new vertices and modified only in case of vertex folding. Then, the field $\mathbb{M}$ explicitly encodes the monotony of newly folded vertices (Figure 4.4). The monotony offsets are used to disambiguate the comparison of two vertices of identical approximated scalar value in the rest of the approach (criticality estimation, integral lines, etc).

4.2.5 Parallelism

Our approach can be easily parallelized using shared-memory parallelism. The first step of our approach, the traversal of the hierarchy, can be trivially parallelized on the vertices, as all operations are local to a vertex. However the hierarchy must be processed in sequential, which implies synchronization between the different levels. The computation of the criticality for non topologically invariant vertices is also completely parallel. The computation of the persistence diagram from the critical points can be parallelized on the saddle points, however locks are necessary for the parallel computation of integral lines, in order to back-propagate the indices of found extrema. The saddle points are sorted in parallel, using the GNU implementation [SK08]. Finally, processing the merge events represented by the triplets is a sequential task, but represents only a fraction of the total sequential computation time (less that 1% in practice).

4.2.6 Uncertainty

By construction, our approach induces a Bottleneck error of ε, which corresponds to a maximum mismatch of ε between the pairs of $\mathcal{D}(\widehat{f}_\varepsilon)$ and these of $\mathcal{D}(f)$. This means that some approximated persistence pairs (with a persistence below 2ε) may not be present in the exact diagram (as they may be matched to the diagonal at a cost lower than ε). We call these pairs *uncertain*. In contrast, the approximated pairs with a persistence beyond 2ε will certainly be present in the exact diagram, and their exact location is bounded within a square of side 2ε. We call these pairs *certain*. Thus, we can visually convey the level of uncertainty of our approximations directly in the persistence diagrams (Figure 4.5). For this, we use a red band to indicate *uncertain* pairs, and we draw the bounding squares for *certain* pairs (Section 4.3.3).

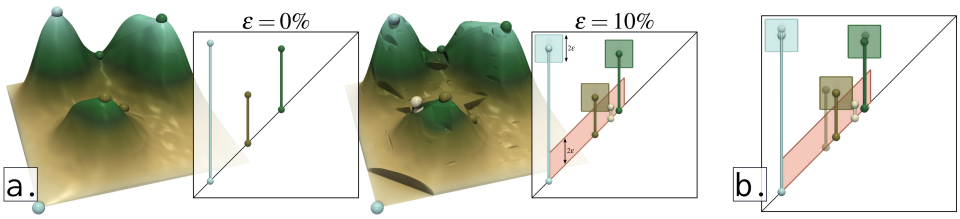


Figure 4.5 – *Approximation uncertainty visualization. The diagram at $\varepsilon = 10\%$ (a., right) exhibits one* uncertain *pair within the red band, which is absent from the exact result (a., left). Squares bound the correct location of* certain *pairs (b., transparent: exact pairs).*

Table 4.1 – *Increase of the number of topologically invariant (TI) vertices for different levels of approximation. For real-world datasets (MinMax and Random excluded), the average proportion of TI vertices rises from 70% for the full precision approach (i.e. $\varepsilon = 0$) to 94% for a relative Bottleneck error of 5% ($\varepsilon = 0.05$).*

Dataset	$\sum_{i=0}^{h} \lvert\mathcal{M}_0^h\rvert$	% TI			
		$\varepsilon = 0$	$\varepsilon = 0.01$	$\varepsilon = 0.05$	$\varepsilon = 0.1$
At	931,102	64.7	95.3	96.1	96.3
SeaSurfaceHeight	1,384,636	61.4	90.8	96.7	98.3
Ethanediol	2,057,380	77.7	97.4	97.6	97.6
Hydrogen	2,413,516	73.5	97.8	97.3	97.4
Isabel	3,605,596	44.5	80.8	91.6	93.5
Combustion	4,378,378	65.5	89.6	96.3	97.3
Boat	4,821,318	88.5	96.9	97.0	97.2
MinMax	18,994,891	99.3	99.5	99.5	99.5
Aneurism	19,240,269	95.6	96.1	97.4	98.2
Foot	19,240,269	64.4	66.6	73.7	86.9
Heptane	31,580,914	82.2	96.4	98.2	98.7
Random	18,117,510	0.9	0.9	1.0	1.1
Backpack	111,929,613	39.1	77.2	94.7	97.7

4.3 Results

This section presents experimental results obtained on a variety of datasets, available on public repositories [Kla20, TTK20]. We refer to Appendix A for an extensive list. We implemented our approach in C++, as modules for the Topology ToolKit (TTK) [TFL+17, BMBF+19]. The experiments were carried out on a desktop computer with two Xeon CPUs (3.0 GHz, 2×4 cores) and 64 GB of RAM.

4.3.1 Time Performance

The time complexity of this approach is similar to the complexity of the non-interruped progressive algorithm presented in Chapter 3. The first two steps of our approach, that amount to the hierarchy traversal and the computation of critical points, have a linear complexity in the number of vertices: $\mathcal{O}(\sum_{i=0}^{i=h} \lvert\mathcal{M}_0^i\rvert)$. The third step, to compute the saddle-extremum persistence from the critical points, is identical to the progressive approach (Section 3.4), except that we compute the persistence diagram exclusively on the last, finest level of the hierarchy. It has a practical time complexity (Section 3.5.2) of $\mathcal{O}\big(\lvert\mathcal{M}_1^h\rvert + n_s \log n_s + n_s \alpha(n_m)\big)$, where α stands for the inverse of the Ackermann function, and n_s and n_m respectively denote the number of saddle points and extrema.

Table 4.1 reports the number of TI vertices for all datasets, for various

Table 4.2 – *Sequential computation times (in seconds) of our approach for the approximation of persistence diagram, for different approximation errors. The column* Default *reports the run time of the default approach in the Topology ToolKit [GFJT19]. The last column indicates the speedup of our approach with an approximation error of* 5%*, against the fastest of the two exact methods (left). Bold numbers indicate the smallest* ε *providing a speedup over reference approaches.*

Dataset	Default[GFJT19]	Progressive (Section 3.4)	Ours			
			$\varepsilon = 1\%$	$\varepsilon = 5\%$	$\varepsilon = 10\%$	5% speedup
At	0.27	0.25	**0.14**	0.15	0.17	38.5%
SeaSurfaceHeight	0.48	0.38	**0.29**	0.26	0.25	30.9%
EthaneDiol	0.48	0.43	**0.28**	0.31	0.33	28.7%
Hydrogen	0.99	0.64	**0.43**	0.48	0.47	24.6%
Isabel	1.29	1.49	**0.95**	0.89	0.92	30.7%
Combustion	2.55	1.37	**0.99**	0.90	0.85	34.1%
Boat	1.22	0.82	0.97	1.06	1.02	-28.1%
MinMax	4.01	1.92	2.14	2.28	2.13	-18.4%
Aneurism	4.66	3.43	3.62	3.52	**3.00**	-2.7%
Foot	9.86	10.42	10.38	**8.14**	6.14	17.4%
Heptane	8.09	7.41	**5.41**	5.18	5.16	30.1%
Random	37.29	30.77	**28.95**	29.04	30.46	5.6%
Backpack	77.28	107.31	**62.06**	40.11	31.76	48.1%

approximation errors. The column $\varepsilon = 0$ corresponds to the numbers of TI vertices previsously reported in Section 3.5.1 in the exact case. For the majority of datasets, we observe a large increase in the number of TI vertices, from a proportion of 70% on average on the real-life datasets to a proportion of 90% for a small error of 1%, and 94% for a mild tolerance of 5% on the approximation. The largest increase in the number of TI vertices is reported for *Backpack* (being a large and noisy dataset). This table confirms that our strategy of *vertex folding* (Section 4.2.2) indeed implies a sensible increase in the number of TI vertices, even for mild approximation errors. Regarding the criticality estimation (step 2, Section 4.2.1), as no computation is needed for the vertices which remained topologically invariant throughout the hierarchy (Section 4.2.1), a higher proportion of TI vertices in the data is thus likely to significantly decrease the computational workload, resulting in lower computation times.

Table 4.2 details the sequential computation times of our approach for different approximation errors. They are compared with public implementations of exact algorithms, both available in TTK [TFL+17]: the *progressive* approach of Chapter 3 (run up to the finest resolution, hence producing an exact result), and the *default* algorithm used in TTK [GFJT19] (run at the finest hierarchy level). The last column of Table 4.2 present the speedups obtained with a Bottleneck error tolerance of 5%, compared with the fastest of either reference approaches. We observe an average reduc-

Table 4.3 – *Parallel computation times (in seconds, on 8 physical cores) of our algorithm with different approximation errors. The presented speedups relate to the sequential run times.*

Dataset	Default[GFJT19]	Progressive (Section 3.4)	$\varepsilon = 0.01$	speedup	$\varepsilon = 0.05$	speedup	$\varepsilon = 0.1$	speedup
At	0.20	0.07	0.05	2.61	0.07	2.24	0.07	2.25
SeaSurfaceHeight	0.21	0.13	0.08	3.65	0.07	3.91	0.07	3.77
EthaneDiol	0.21	0.11	0.08	3.69	0.09	3.60	0.10	3.35
Hydrogen	0.72	0.20	0.13	3.18	0.16	2.95	0.17	2.80
Isabel	0.47	0.46	0.54	1.77	0.28	3.15	0.30	3.05
Combustion	0.34	0.51	0.54	1.83	0.31	2.92	0.25	3.41
Boat	0.29	0.29	0.33	2.94	0.37	2.85	0.34	2.98
MinMax	0.80	0.61	0.69	3.11	0.54	4.23	0.54	3.97
Aneurism	2.03	1.51	1.62	2.23	1.36	2.59	1.33	2.25
Foot	3.12	2.34	2.65	3.91	1.91	4.25	2.04	3.01
Heptane	2.44	2.35	2.08	2.60	1.33	3.88	1.51	3.41
Random	27.04	8.47	7.15	4.05	7.77	3.74	8.89	3.43
Backpack	30.36	24.88	12.63	4.92	8.50	4.72	7.02	4.53

tion of the run times of 18% on real-world datasets, which confirms that our strategy of maximizing the number of TI vertices effectively reduces the computation times. The observed speedups are consistent with the increases in the proportion of TI vertices reported in Table 4.1: a large increase in the number of TI vertices implies an important reduction of the computation time. Interestingly, we find our method most beneficial on datasets that initially present a low amount of TI vertices. This usually corresponds to a high level of noise, which impedes both reference methods. In particular, the highest speedup is achieved on *Backpack*, our largest, noisiest real-world dataset, with a reduction of computation time of nearly 50%. In contrast, our method fails to reduce the computation times on smooth datasets such as *MinMax*, *Boat* or *Aneurism*, for which the *progressive* approach really shines. These datasets exhibit high initial proportions of TI vertices, which limits the increase in TI vertices enabled by our approach (Table 4.1). For *Random*, the interpolation cost of folded vertices seems to counterbalance the speedup induced by the increase in TI vertices.

Table 4.3 lists the computation times obtained with the parallel version of our algorithm. We find an overall average parallel efficiency of 43% with an error level of 5%, which is on par with the progressive approach (Section 3.4). Although the traversal of the hierarchy can be trivially parallelize over vertices, it is subject to synchronization steps between hierarchy levels. The last step of the approach, deducing the persistence diagram from the critical points, is less balanced. Indeed, the parallel computation of integral lines between saddle points and extrema (Section 4.2.1) necessitates locks.

4.3.2 Approximation Accuracy

By design, our approach produces approximated persistence diagrams with a guaranteed bound on the Bottleneck distance to the exact result (Section 4.2.3). In the following, we additionally evaluate the accuracy of our approximation by considering the L_2 Wasserstein distance (Section 2.4, computed with the Auction algorithm [Ber81, KMN16]) between our approximations and the exact result. We also evaluate $\|\widehat{f} - f\|_2$ to quantify the pointwise error of the approximated field $\widehat{f}$. Results are given in Table 4.4.

For comparison, we perform the same evaluation for a naive baseline approximation which provides identical guarantees. This baseline consists in computing, with an exact algorithm, the diagram of a *staircase* function $f^{\square}$, i.e. a quantized version of the input data f, with a quantization step of 2ε. By construction, the staircase function verifies $\|f^{\square} - f\|_\infty < \varepsilon$ (Figure 4.6) and its persistence diagram is then guaranteed not to exceed a Bottleneck error of ε [CEH05]. Table 4.4 compares the accuracy of this baseline approximation to our algorithm. In terms of L_2 distance, our approximations of the input scalar fields are around 2 times more accurate on average (on real-world datasets, *Random* and *MinMax* excluded). This difference could be expected, as our method performs local and adaptive linear interpolations, while the staircase approach systematically flattens

Table 4.4 – *Accuracy comparison between our approach and a naive baseline approximation based on the computation of a staircase function, for the same relative Bottleneck error of 5%. Our approximations are more accurate on average, both in terms of the L_2 distance of the approximated field (2 times more accurate) and in terms of the L_2-Wasserstein distance to the exact persistence diagram (5 times).*

Dataset	Staircase L_2	Ours L_2	Ratio	Staircase W_2	Ours W_2	Ratio
At	276.66	75.31	3.67	3.31	1.01	3.29
SeaSurfaceHeight	92.3	58.05	1.59	8.75	1.69	5.17
EthaneDiol	337.55	73.2	4.61	1.78	0.61	2.9
Hydrogen	11.0	13.0	0.85	25.69	12.73	2.02
Isabel	3,591.99	1,569.98	2.29	42.26	8.08	5.23
Combustion	38.04	17.59	2.16	0.87	0.21	4.1
Boat	24.18	9.11	2.66	2.14	0.49	4.37
MinMax	117.55	0.37	314.77	0.0	0.0	-
Aneurism	6.0	9.0	0.67	1,198.14	128.61	9.32
Foot	16,006.11	10,784.33	1.48	5,859.67	1,419.51	4.13
Heptane	5.0	12.0	0.42	716.1	61.12	11.72
Random	29,270.73	3,468.14	8.44	23,863.56	1,038.1	22.99
Backpack	142.0	142.0	1.0	17,941.13	1,933.83	9.28

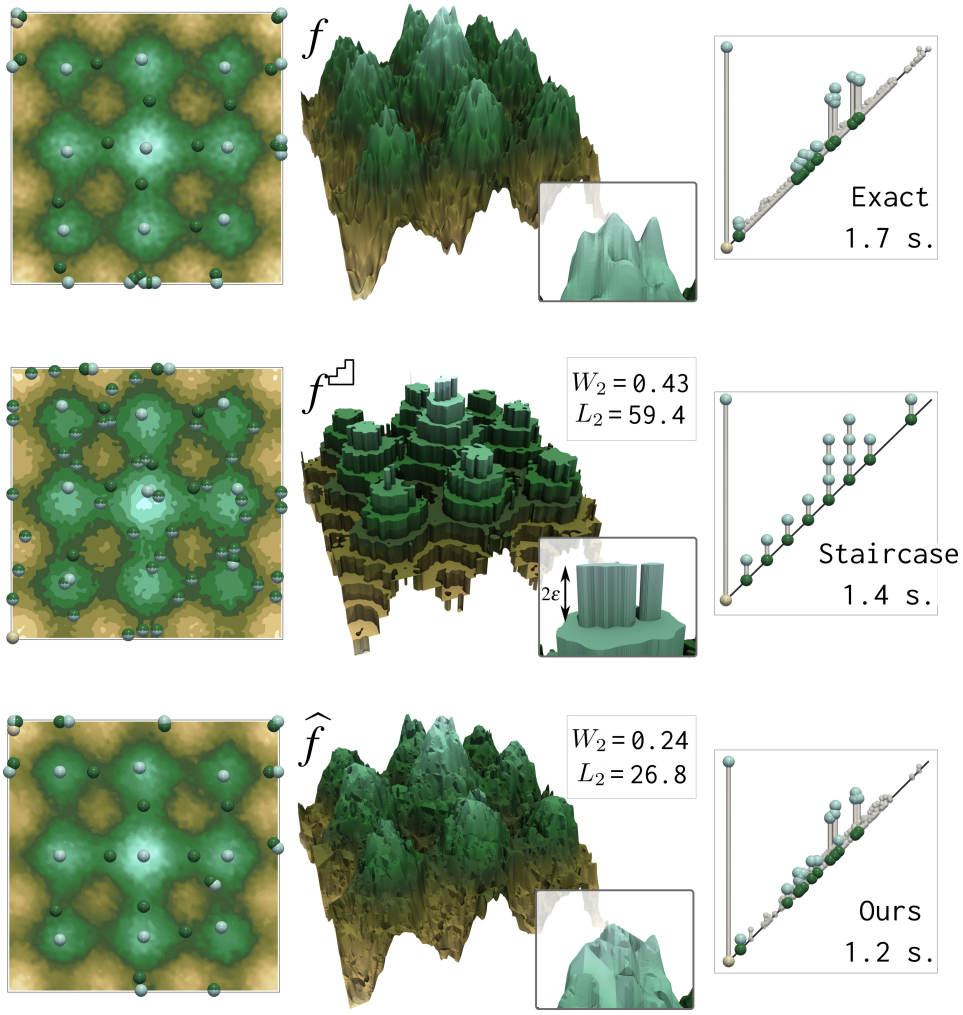


Figure 4.6 – *Comparison of our approach to a naive baseline approximation (staircase field, middle row), for the same tolerance of 5% on the Bottleneck error. Our approach (bottom row) provides an approximated persistence diagram that resembles the exact result (top row), and is around 2 times closer in terms of the Wasserstein distance (W_2). The high persistence critical points (spheres, 2D domain) capture more precisely the features of the data in our case. Our approximated field is also two times closer (L_2 norm) to the exact field than the naive approximation.*

the data. However, our approach is also significantly more accurate with regards to the L_2-Wasserstein distance to the exact persistence diagrams: 4 times on average on our real-world datasets.

Figure 4.6 further illustrates the limitations of the *staircase* baseline. For a given approximation error (5%), our method gives an approximated diagram that is visually more similar to the exact result, and which better depicts the number of salient features, as well as the noise in the data. In contrast, the diagram produced by the *staircase* approximation is more difficult to interpret, as the positions of persistence pairs are quantized on a grid in the 2D birth/death space, resulting in several co-located pairs,

which cannot be distinguished visually. Our approximation of the scalar field is also closer to the exact field, both visually and in terms of the L_2 norm, enabling more accurate critical point approximations in the domain (top).

4.3.3 Qualitative Analysis

This section discusses the utility of our approximations from a qualitative point of view, for data analysis and visualization purposes. Figure 4.1 shows the result of our approach on the *Backpack* dataset. In this example, our approximated diagrams correctly capture the high persistence maxima of the scalar field, which correspond to high density objects inside the bag (bottles, wires, and metallic parts of the bag). The approximate field $\widehat{f}_\varepsilon$ resulting from the vertex folding (Section 4.2.2) is more precise in the vicinity of features of high persistence. Indeed, the volume rendering in the top views of Figure 4.1 shows clearly the objects inside the bag (high persistence maxima), while isocontours capturing the cloth of the bag (Figure 4.1, bottom) illustrate the deterioration of the field in this region. The same phenomenon can be noted for the *Foot* dataset (Figure 4.7). More vertices are folded in the vicinity of low persistence features, typically the skin of the foot, and the level of deterioration of the field increases with the approximation error. Conversely, high persistence features (bones of the toes) are well captured.

The above observation suggests that our method provides a better approximation for persistent features, and a more degraded evaluation for

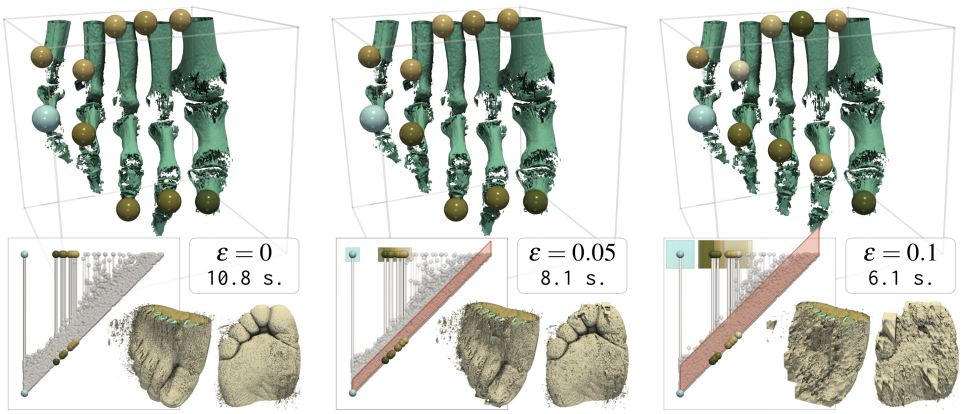


Figure 4.7 – *Approximated persistence diagrams on the Foot dataset. The 3D top views show the ten most persistent maxima (spheres), corresponding to the bones of the foot (isocontour, computed on the approximated field $\widehat{f}_\varepsilon$). Our approximated diagrams correctly capture the most persistent features at a reduced computational cost.*

less persistent structures, which was an original motivation for our approach (to focus the computational efforts on relevant structures). This is confirmed in Figure 4.1 by the number of *noisy* pairs (of low persistence, within the red band) in the approximated diagrams, which is significantly lower than the amount of noisy pairs (of identical persistence) in the exact diagram (Figure 4.1, in parenthesis).

Figure 4.8 compares our approximations to the progressive approach. To generate an approximated result with the progressive approach, we interrupt its computation at the penultimate hierarchy level (the computation would become exact at the final level). As documented in Section 3.5.4 of Chapter 3, such intermediate results can provide useful previews, but however, with no guarantee on the approximation error. This is illustrated in Figure 4.8, where the progressive approximation fails at correctly capturing the maximum of largest persistence. In contrast, our method produces persistence diagrams that correctly convey the salience of the features, and are five times more accurate, in terms of the Wasserstein distance.

Indications about the approximation uncertainty (Section 4.2.6) can be displayed in the output diagrams. Figure 4.9 shows our approximations for the *Isabel* dataset. For each approximation error, the red band indicates uncertain pairs, which may not be part of the exact result. *Certain* pairs are represented with a square bounding their correct location. These glyphs give a good sense of the approximation uncertainty, and are useful to assess the reliability of the diagram. For instance, a large pair in the uncertain zone may indicate the presence of a medium persistence feature

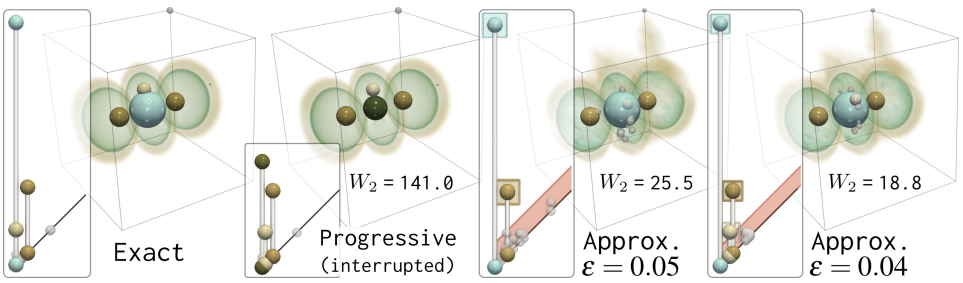


Figure 4.8 – *Approximation of persistence diagrams on the hydrogen dataset. The best interrupted result of the progressive approach (Chapter 3) fails at correctly capturing the global maximum (accurately detected only at the last level), resulting in a diagram that is 5 times less accurate than our 5% approximation (L_2-Wasserstein distance to the exact result). In contrast, our approximation correctly captures the high persistence features of the data. On the far right, our 4% approximated diagram detects as certain the fourth most persistent maxima, which was marked as uncertain with $\varepsilon = 0.05$.*

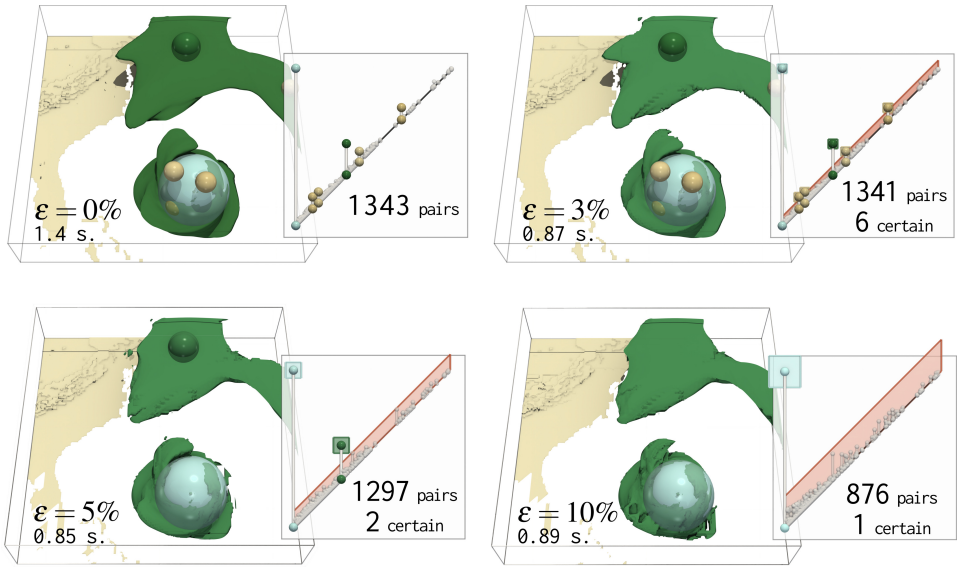


Figure 4.9 – *Approximated diagrams for the Isabel hurricane dataset. The most persistent maxima of the field (the magnitude of the wind velocity) are represented in the 3D view as spheres (scaled by persistence). The approximation uncertainty is visualized in the diagram: in the red band indicates uncertain persistence pairs that may not exist in the exact persistence diagram, and colored squares indicate a bound on the location of the persistence pairs that are certain to exist in the exact diagram. Our approximated persistence diagrams correctly capture these certain pairs in the order of their persistence, with higher persistence features being detected at higher tolerance on the Bottleneck error.*

in the data. This can be confirmed with the computation of a slighlty better estimation, as illustrated in Figure 4.8 for the fourth most persistent feature.

4.4 Limitations and Discussion

Our approximations tend to generate much less low-persistence features than exact algorithms (Figure 4.1), which can be an issue if features of interested are hidden among noisy features near the diagonal. On the upside, this characteristic of our approximations make them well suited for subsequent analysis and processing (e.g. distances and clustering), as diagrams are often thresholded in practice prior to further computations, to remove low persistence pairs anyway.

An important limitation of our approximation approach, compared to the work presented in Chapter 3, is its lack of progressivity. Indeed, to provide strong approximation guarantees, the hierarchy has to be completely traversed and no intermediate result can be provided.

Another limitation is that our approach only supports saddle-

extremum persistence pairs at the moment. However, from our experience, these correspond in practice to the key features users tend to be interested in.

Finally, our approach provides strong guarantees on the Bottleneck distance. Future work is needed for the theoretical study of the impact of our approximations on the L_2 Wasserstein metric.

4.5 Summary

This chapter introduced a method for the approximation of the persistence diagram of a scalar field. This work revisits the approach previously introduced (Chapter 3), that generated preview diagrams upon interruption of a progressive framework. We addressed the main drawback of this approach, namely the lack of guaranteed error bounds on the diagram estimations. In contrast, we presented a novel algorithm that efficiently computes the approximation of a persistence diagram within a user controlled approximation error on the Bottleneck distance to the exact result. We showed that the approximated persistence diagrams are relevant for visualization and data analysis tasks, as they correctly describe the high persistence features in the data (*i.e.* the number and salience of important features), and they are more concise in practice than the exact diagrams. The uncertainty related to our approximations can be effectively depicted visually inside the diagrams.

5 Progressive Wasserstein Barycenters of Persistence Diagrams

Contents

Our contribution in one image 93

5.1 Context 94

5.1.1 Related Work 94

5.1.2 Contributions 97

5.2 Background 98

5.2.1 Efficient Wasserstein distance computation by Auction . . 98

5.2.2 Wasserstein barycenters of Persistence diagrams 100

5.3 Overview 101

5.4 Progressive Barycenters 101

5.4.1 Auctions with Price Memorization 101

5.4.2 Accuracy-driven progressivity 102

5.4.3 Persistence-driven progressivity 103

5.4.4 Parallelism 104

5.4.5 Computation time constraints 104

5.5 Application to Ensemble Topological Clustering 107

5.6 Results 108

5.6.1 Time performance 109

5.6.2 Barycenter quality 111

5.6.3 Ensemble visual analysis with Topological Clustering . . . 114

5.7 Limitations 116

5.8 Overall time-constrained pipeline 118

5.9 Summary 119

Modern numerical simulations are subject to a variety of input parameters, related to the initial conditions of the system under study, as well as the configuration of its environment. Given recent advances in hardware computational power, engineers and scientists can now densely sample the space of these input parameters, in order to better quantify the sensitivity of the system. For scalar variables, this means that the data which is considered for visualization and analysis is no longer a single field, but a collection, called *"ensemble"*, of scalar fields representing the same phenomenon, under distinct input conditions and parameters. In this context, extracting the global trends in terms of features of interest in the ensemble is a major challenge. Although it is possible to compute a persistence diagram for each member of an ensemble, in particular in-situ [BAA+16, ABG+15], this process only shifts the problem from the analysis of an ensemble of scalar fields to an ensemble of persistence diagrams. Then, given such an ensemble of diagrams, the question of estimating a diagram which is *representative* of the set naturally arises, as such a representative diagram could visually convey to the users the global trends in the ensemble in terms of features of interest. For this, naive strategies could be considered, such as estimating the persistence diagram of the mean of the ensemble of scalar fields. However, given the additive nature of the pointwise mean, this yields a persistence diagram with an incorrect number of features (Figure 5.1), which is thus not representative of any of the diagrams of the input scalar fields.

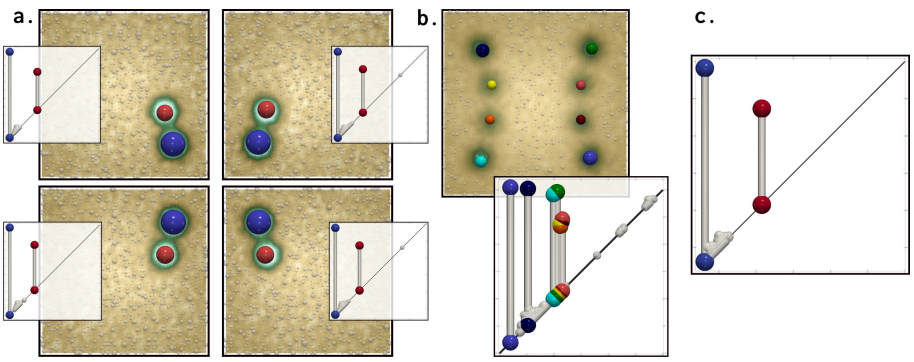


Figure 5.1 – *Synthetic ensemble of a pattern with 2 gaussians and additive noise (a). The persistence diagram of the pointwise mean (b) contains 8 highly persistent features although each of the input ensemble members contain only 2 features. The Wasserstein barycenter (c) provides a diagram which is representative of the set, with a feature number, range and salience which better describes the input ensemble (2 large features).*

A promising alternative consists in considering the *barycenter* of a set of diagrams, given a distance metric between them, such as the so-called *Wasserstein* metric Section 2.4, hence the term *Wasserstein barycenter*. For this, an algorithm has been proposed by Turner et al. [TMMH14]. However, it is based on an iterative procedure, for which each iteration relies itself on a demanding optimization problem (optimal assignment in a weighted bipartite graph [Mun57]), which makes it impractical for real-life datasets.

To address this issue, this chapter presents an efficient algorithm for the progressive approximation of Wasserstein barycenters of persistence diagrams, with applications to the visual analysis of ensemble data. Given a set of scalar fields, our approach enables the computation of a persistence diagram which is representative of the set, and which visually conveys the number, data ranges and saliences of the main features of interest found in the set. Such representative diagrams are obtained by computing explicitly the discrete Wasserstein barycenter of the set of persistence diagrams, a notoriously computationally intensive task. In particular, we revisit efficient algorithms for Wasserstein distance approximation [Ber81, KMN16] to extend previous work on barycenter estimation [TMMH14]. We present a new fast algorithm, which progressively approximates the barycenter by iteratively increasing the computation accuracy as well as the number of persistent features in the output diagram. Such a progressivity drastically improves convergence in practice and allows to design an interruptible algorithm, capable of respecting computation time constraints. This enables the approximation of Wasserstein barycenters within interactive times. We present an application to ensemble clustering where we revisit the k-means algorithm to exploit our barycenters and compute, within execution time constraints, meaningful clusters of ensemble data along with their barycenter diagram. Extensive experiments on synthetic and real-life data sets report that our algorithm converges to barycenters that are qualitatively meaningful with regard to the applications, and quantitatively comparable to previous techniques, while offering an order of magnitude speedup when run until convergence (without time constraint). Our algorithm can be trivially parallelized to provide additional speedups in practice on standard workstations.

The work presented in this chapter have been published in the journal IEEE Transactions on Visualization and Computer Graphics as part of the proceedings of the IEEE VIS 2019 conference [VBT19], where it won a Best Paper Honorable Mention award. It was certified replicable by the Graph-

ics Replicability Stamp Initiative (http://www.replicabilitystamp.org/). Our implementation and the data needed to reproduce this work are available at https://github.com/julesvidal/wasserstein-pd-barycenter. It is also integrated in the Topology ToolKit [TFL+17].

Our contribution in one image

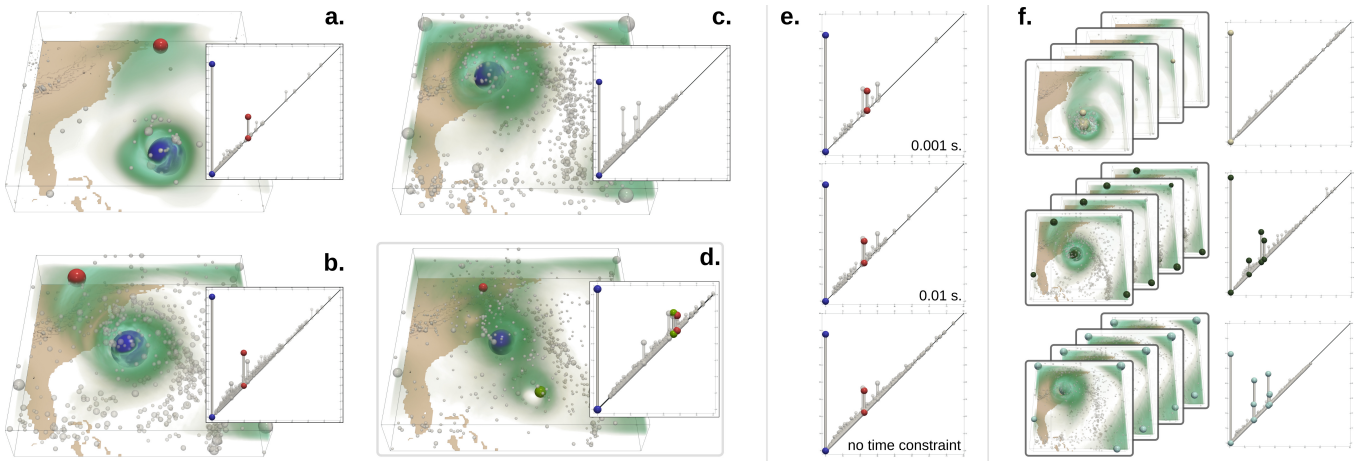


Figure 5.2 – *The Persistence diagrams of three members (a-c) of the Isabel ensemble (wind velocity) concisely and visually encode the number, data range and salience of the features of interest found in the data (eyewall and region of high speed wind, blue and red in (a)). In these diagrams, features with a* persistence *smaller than 10% of the function range or on the boundary are shown in transparent white. The pointwise mean for these three members (d) exhibits three salient interior features (due to distinct eyewall locations, blue, green and red), although the diagrams of the input members only report two salient interior features at most, located at drastically different data ranges (the red feature is further down the diagonal in (a) and (b)). The Wasserstein barycenter of these three diagrams (e) provides a more representative view of the features found in this ensemble, as it reports a feature number, range and salience that better matches the input diagrams (a-c). Our work introduces a progressive approximation algorithm for such barycenters, with fast practical convergence. Our framework supports computation time constraints (e) which enables the approximation of Wasserstein barycenters within interactive times. We present an application to the clustering of ensemble members based on their persistence diagrams ((f), lifting: $\alpha = 0.2$), which enables the visual exploration of the main trends of features of interest found in the ensemble.*

5.1 Context

5.1.1 Related Work

The literature related to this work can be classified into three main categories, reviewed in the following: *(i)* uncertainty visualization, *(ii)* ensemble visualization, and *(iii)* persistence diagram processing.

Uncertainty visualization

The analysis and visualization of uncertainty in data is a notoriously challenging problem in the visualization community [PK12, BHJ+14, unco8, JS03, MAH+05, PWL97]. In this context, the data variability is explicitly modeled by an estimator of the probability density function (PDF) of a pointwise random variable. Several representations have been proposed to visualize the related data uncertainty, either focusing on the entropy of the random variables [PGA13], on their correlation [PW12], or gradient variation [PMW13]. When geometrical constructions are extracted from uncertain data, their positional uncertainty has to be assessed. For instance, several approaches have been presented for level sets, under various interpolation schemes and PDF models [AE13, ASE16, PRW11, PWH11, PH11, PH13, PPH13, SKS12, AJ19]. Other approaches addressed critical point positional uncertainty, either for Gaussian [LS16, MOT10, MOT11, PPH12] or uniform distributions [GST14, BJB+12, Szy13]. In general, visualization methods for uncertain data are specifically designed for a given distribution model of the pointwise random variables (Gaussian, uniform, etc.). This challenges their usage with ensemble data, where PDF estimated from empirical observations can follow an arbitrary, unknown model. Moreover, most of the above techniques do not consider multi-modal PDF models, which is a necessity when several distinct trends occur in the ensemble.

Ensemble visualization

Another category of approaches has been studied to specifically visualize the main trends in ensemble data. In this context, the data variability is directly encoded by a series of global empirical observations (*i.e.* the members of the ensemble). Existing visualization techniques typically construct geometrical objects, such as level sets or streamlines, for each member of the ensemble. Then, given this ensemble of geometrical objects, the question of estimating an object which is *representative* of the ensemble

naturally arises. For this, several methods have been proposed, such as spaghetti plots [DHLZ02] in the case of level-set variability in weather data ensemble [PWB+09, SZD+10], or box-plots for the variability of contours [WMK13] and curves in general [MWK14]. For the specific purpose of trend variability analysis, Hummel et al. [HOGJ13] developed a Lagrangian framework for classification in flow ensembles. Related to our work, clustering techniques have been used to analyze the main trends in ensembles of streamlines[FBW16] and isocontours [FKRW16]. However, only few techniques have focused on applying this strategy to topological objects. Overlap-based heuristics have been investigated to estimate a representative contour tree from an ensemble [WZ13, Kra10]. Favelier et al. [FFST18] introduced an approach to analyze critical point variability in ensembles. It relies on the spectral clustering of the ensemble members according to their *persistence map*, a scalar field defined on the input geometry which characterizes the spatial layout of persistent critical points. However, the clustering stage of this approach takes as an input a distance matrix between the persistence maps of all the members of the ensemble, which requires to mantain them all in memory, which is not conceivable for large-scale ensembles counting a high number of members. Moreover, the clustering itself is performed on the spectral embedding of the persistence maps, where distances are loose approximations of the intrinsic metric between these objects. In contrast, the clustering application described in this chapter directly operates on the persistence diagrams of the ensemble members, whose memory footprint is orders of magnitude smaller than the actual ensemble data. This makes our approach more practical, in particular in the perspective of an in-situ computation [BAA+16] of the persistence diagrams. Also, it is built on top of our progressive algorithm for Wasserstein barycenters, which focuses on the Wasserstein metric between diagrams. Optionally, our work can also integrate the spatial layout of critical points by considering a geometrically lifted version of this metric [SPCT18b].

Persistence diagram processing

To define the barycenter of a set of persistence diagrams, a metric *(i)* first needs to be introduced to measure distances between them. Then, barycenters *(ii)* can be formally defined as minimizers of the sum of distances to the set of diagrams.

(i) The estimation of distances between topological abstractions has

been a long-studied problem, in particular for similarity estimation tasks. Several heuristical approaches have been documented for the fast estimation of structural similarity [HSKK01a, TN14, SSW14, TN13]. More formal approaches have studied various metrics between topological abstractions, such as Reeb graphs [BGW14, BMW15] or merge trees [BYM+14]. For persistence diagrams, the *Bottleneck* [CSEH05] and *Wasserstein* distances [Mon81, Kan42, EH09], have been widely studied, for instance in machine learning [Cut13] and adapted for kernel based methods [RHBK15, CCO17, RSL17]. The numerical computation of the Wasserstein distance between two persistence diagrams requires to solve an optimal assignment problem between them. The typical methods for this are the exact Munkres algorithm [Mun57], or an auction-based approach [Ber81] that provides an approximate result with improved time performance. Kerber et al. [KMN16] specialized the auction algorithm to the case of persistence diagrams and showed how to significantly improve performances by leveraging adequate data structures for proximity queries. Soler et al. [SPCT18b] introduced a fast extension of the Munkres approach, also taking advantage of the structure of persistence diagrams, in order to solve the assignment problem in an exact and efficient way, using a reduced, sparse and unbalanced cost matrix.

(ii) Recent advances in optimal transport [Cut13] enabled the practical resolution of transportation problems between continuous quantities [CD14, SDGP+15]. These methods have been successfully applied to persistence diagrams [LCO18]. However, this application requires to represent the input diagrams as heat maps of fixed resolution [ACE+17]. Such a rasterization can be interpreted as a pre-normalization of the diagrams, which can be problematic for applications where the considered diagrams have different persistence scales, as typically found with time-varying phenomena for instance. Moreover, this approach does not explicitly produce a persistence diagram as an output, but a heat map of the population of persistence pairs in the barycenter. This challenges its usage for visualization applications, as the features of interest in the barycenter cannot be directly inferred from the barycenter heat map. In contrast, our approach produces explicitly a persistence diagram as an output, from which the geometry of the features (their number, data ranges and salience) can be directly visually inspected. Moreover, such an explicit representation also enables the efficient geometrical lifting of the Wasserstein metric [SPCT18b], which is relevant for scientific visualization applications but which would require to regularly sample a five dimen-

sional space with heat map based approaches. Turner et al. [TMMH14] introduced an algorithm for the computation of a Fréchet mean of a set of persistence diagrams with regard to the Wasserstein metric. This approach provides explicit barycenters, which makes it appealing for the applications. However, its very high computational cost makes it impractical for real-life data sets. In particular, it is based on an iterative procedure, for which each iteration relies itself on N optimal assignment problems [Mun57] between persistence diagrams (for the Wasserstein distances), where N is the number of members in the ensemble. A naive approach to address this computational bottleneck would be to combine this method with the efficient algorithms for Wasserstein distances mentioned previously [KMN16, SPCT18b]. However, as shown in Table 5.6, such an approach is still computationally expensive and it can require up to hours of computation on certain data sets. In contrast, our algorithm converges in multi-threaded mode in a couple of minutes at most, and its progressive nature additionally allows for its interruption within interactive times, while still providing qualitatively meaningful results.

5.1.2 Contributions

This chapter presents the following new contributions:

1. *A progressive algorithm for Wasserstein barycenters of persistence diagrams:* We revisit efficient algorithms for Wasserstein distance approximation [Ber81, KMN16] in order to extend previous work on barycenter estimation [TMMH14]. In particular, we introduce a new approach based on a progressive approximation strategy, which iteratively refines both computation accuracy and output details. The persistence pairs of the input diagrams are progressively considered in decreasing order of persistence. This focuses the computation towards the most salient features of the ensemble, while considering noisy persistent pairs last. The returned barycenters are explicit and provide insightful visual hints about the features present in the ensemble. Our progressive strategy drastically accelerates convergence in practice, resulting in an order of magnitude speedup over the fastest combinations of existing techniques. The algorithm is trivially parallelizable, which provides additional speedups in practice on standard workstations. We present an *interruptible* extension of our algorithm to support computation time constraints. This enables to produce barycenters accounting for the main features of the data within interactive times.

2. *An interruptible algorithm for the clustering of persistence diagrams:* We extend the above methods to revisit the *k-means* algorithm and introduce an interruptible clustering of persistence diagrams, which is used for the visual analysis of the global feature trends in ensembles.

5.2 Background

5.2.1 Efficient Wasserstein distance computation by Auction

This section briefly describes a fast approximation of Wasserstein distances by the auction algorithm [Ber81, KMN16].

The assignment problem involved in the definition of the Wasserstein distance (Equation 2.2, page 28):

$$W_q\big(\mathcal{D}(f), \mathcal{D}(g)\big) = \min_{\phi \in \Phi} \left(\sum_{a \in \mathcal{D}(f)} d_q\big(a, \phi(a)\big)^q \right)^{1/q} \tag{2.2}$$

can be modeled in the form of a weighted bipartite graph, where the points $a \in \mathcal{D}'(f)$ are represented as nodes, connected by edges to nodes representing the points of $b \in \mathcal{D}'(g)$, with an edge weight given by $d_2(a,b)^2$ (Equation 2.1, page 28). To efficiently estimate the optimal assignment, Bertsekas introduced the *auction* algorithm [Ber81] (Figure 5.3), which replicates the behavior of a real-life auction: the points of $\mathcal{D}'(f)$ are acting as *bidders* that iteratively make offers for the purchase of the points of $\mathcal{D}'(g)$, known as the *objects*. Each bidder $a \in \mathcal{D}'(f)$ makes a benefit $\beta_{a \to b} = -d_2(a,b)^2$ for the purchase of an object $b \in \mathcal{D}'(g)$, which is itself labeled with a price $p_b \geq 0$, initially set to 0. During the iterations of the auction, each bidder a tries to purchase the object b of highest *value* $v_{a \to b} = \beta_{a \to b} - p_b$. The bidder a is then said to be assigned to the object b. If b was previously assigned, its previous owner becomes unassigned. At this stage, the price of b is increased by $\delta_a + \varepsilon$, where δ_a is the absolute difference between the two highest values $v_{a \to b}$ that the bidder a found among the objects b, and where $\varepsilon > 0$ is a constant. This bidding procedure is repeated iteratively among the bidders, until all bidders are assigned (which is guaranteed to occur by construction, thanks to the ε constant). At this point, it is said that an *auction round* has completed: a bijective, possibly sub-optimal, assignement ϕ exists between $\mathcal{D}'(f)$ and $\mathcal{D}'(g)$. The overall algorithm will repeat auction rounds, which progressively increases prices under the effect of competing bidders.

The constant ε plays a central role in the auction algorithm. Let

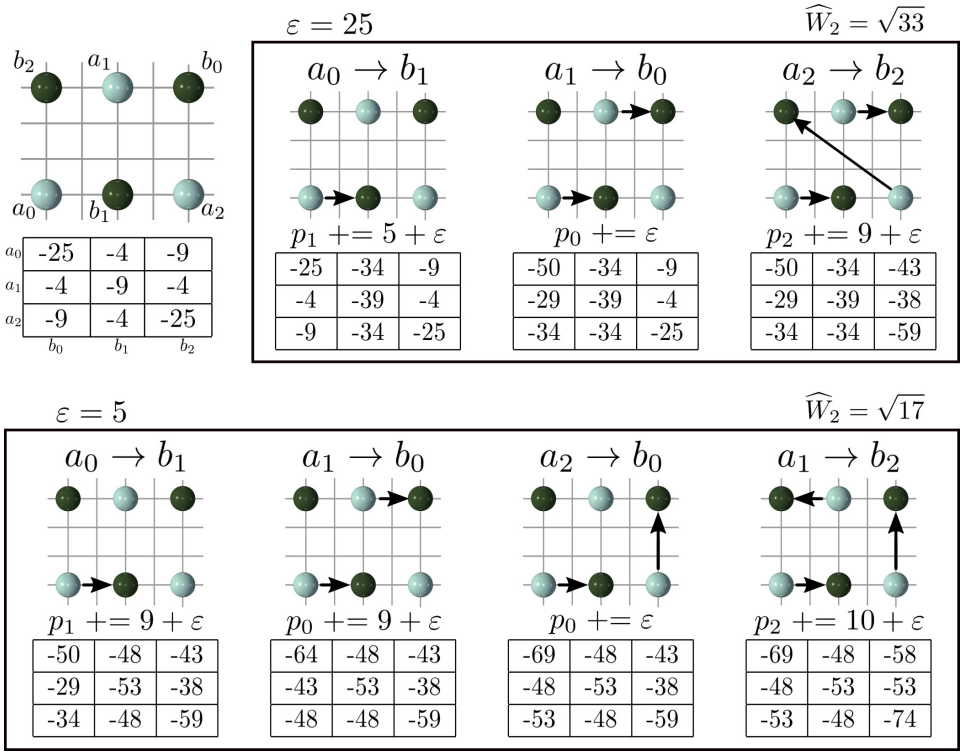


Figure 5.3 – *Illustration of the* Auction *algorithm [Ber81] on a 2D example for the optimal assignment of 2 point sets (light and dark green). Boxes and columns represent* auction rounds *and* auction iterations *respectively. Each matrix reports the value $v_{a\to b}$ currently estimated by the bidder a for the purchase of the object b. Initially (top left), $v_{a\to b} = \beta_{a\to b} = -d_2(a,b)^2$ as p_b is set to 0. Assignments are shown with black arrows. After the first round (top box), a sub-optimal assignment is achieved, which becomes optimum at the second round (bottom box). Note that bidders can steal objects from each other within one auction round (iteration 3, second round).*

$\widehat{W_2}(\mathcal{D}'(f), \mathcal{D}'(g)) = \sqrt{\sum_{a\in\mathcal{D}'(f)} d_2(a, \phi(a))^2}$ be the approximation of the Wasserstein distance $W_2(\mathcal{D}(f), \mathcal{D}(g))$, obtained with the assignment ϕ returned by the algorithm. Large values of ε will drastically accelerate convergence (as they imply fewer iterations for the construction of a bijective assignment ϕ within one auction round, Figure 5.3), while low values will improve the accuracy of $\widehat{W_2}$. This observation is a key insight at the basis of our approach. Bertsekas suggests a strategy called *ε-scaling*, which decreases ε after each auction round. In particular, if:

$$\widehat{W_2}(\mathcal{D}'(f), \mathcal{D}'(g))^2 \leq (1+\gamma)^2 \Big(\widehat{W_2}(\mathcal{D}'(f), \mathcal{D}'(g))^2 - \varepsilon |\mathcal{D}'(f)| \Big) \tag{5.1}$$

then it can be shown that [BC91, KMN16]:

$$W_2(\mathcal{D}(f), \mathcal{D}(g)) \leq \widehat{W_2}(\mathcal{D}'(f), \mathcal{D}'(g)) \leq (1+\gamma) W_2(\mathcal{D}(f), \mathcal{D}(g)) \tag{5.2}$$

This result is particularly important, as it enables to estimate the optimal assignment, and thus the Wasserstein distance, with an on-demand

accuracy (controlled by the parameter γ) by using Equation 5.1 as a stopping condition for the overall auction algorithm. For persistence diagrams, Kerber et al. showed how the computation could be accelerated by using space partitioning data structures such as kd-trees [KMN16]. In our applications, ε is initially set to be equal to $1/4$ of the largest edge weight $d_2(a,b)^2$, and is divided by 5 after each auction round, as recommended by Bertsekas [Ber81]. We use $\gamma = 0.01$, as suggested by Kerber et al. [KMN16].

5.2.2 Wasserstein barycenters of Persistence diagrams

Let $\mathbb{D}$ be the space of persistence diagrams. The discrete *Wasserstein barycenter* of a set $\mathcal{F} = \{\mathcal{D}(f_1), \mathcal{D}(f_2), \ldots, \mathcal{D}(f_N)\}$ of persistence diagrams can be introduced as the Fréchet mean of the set, under the metric W_2. It is the diagram $\mathcal{D}^*$ that minimizes its distance to all the diagrams of the set (*i.e.* minimizer of the so-called Fréchet energy):

$$\mathcal{D}^* = \underset{\mathcal{D}\in\mathbb{D}}{\arg\min} \sum_{\mathcal{D}(f_i)\in\mathcal{F}} W_2\big(\mathcal{D}, \mathcal{D}(f_i)\big)^2 \tag{5.3}$$

The computation of Wasserstein barycenters involves a computationally demanding optimization problem, for which the existence of at least one locally optimum solution has been shown by Turner et al. [TMMH14], who also introduced the first algorithm for its computation. This algorithm (Algorithm 1) consists in iterating a procedure that we call *Relaxation* (line 3 to 8), which resembles a Lloyd relaxation [Llo82], and which is composed itself of two sub-routines: *(i) Assignment* (line 5) and *(ii) Update* (line 7). Given an initial barycenter candidate $\mathcal{D}$ randomly chosen among the set $\mathcal{F}$, the first step (*(i) Assignment*) consists in computing an optimal assignment $\phi_i : \mathcal{D} \to \mathcal{D}(f_i)$ between $\mathcal{D}$ and each diagram $\mathcal{D}(f_i)$ of the set $\mathcal{F}$, with regard to Equation 2.2. The second step (*(ii) Update*) consists in updating the candidate $\mathcal{D}$ to a position in $\mathbb{D}$ which minimizes the sum of its squared distances to the diagrams of $\mathcal{F}$ under the current set of assignments $\{\phi_1, \phi_2, \ldots, \phi_N\}$. In practice, this last step is achieved by replacing each point $a \in \mathcal{D}$ by the arithmetic mean (in the birth/death space) of all its assignments $\phi_i(a)$. The overall algorithm continues to iterate the *Relaxation* procedure until the set of optimal assignments ϕ_i remains identical for two consecutive iterations.

Algorithm 1: Reference algorithm for Wasserstein Barycenters [TMMH14].

Input : Set of diagrams $\mathcal{F} = \{\mathcal{D}(f_1), \mathcal{D}(f_2), \ldots, \mathcal{D}(f_N)\}$
Output : Wasserstein barycenter $\mathcal{D}^*$

1: $\mathcal{D}^* \leftarrow \mathcal{D}(f_i)$ // with i randomly chosen in $[1, N]$
2: **while** $\{\phi_1, \phi_2, \ldots, \phi_N\}$ change **do**
3: // Relaxation start
4: **for** $i \in [1, N]$ **do**
5: $\phi_i \leftarrow Assignment(\mathcal{D}(f_i), \mathcal{D}^*)$ // optimizing Equation 2.2
6: **end for**
7: $\mathcal{D}^* \leftarrow Update(\phi_1, \ldots, \phi_n)$ // arithmetic means in birth/death space
8: // Relaxation end
9: **end while**
10: **return** $\mathcal{D}^*$

5.3 Overview

The key insights of our approach are twofolds. First, in the reference algorithm (Algorithm 1), from one *Relaxation* iteration to the next (lines 3 to 8), the estimated barycenter is likely to vary only slightly. Thus, the assignments involved in the Wasserstein distance estimations can be re-used as initial conditions along the iterations of the barycenter *Relaxation* (Section 5.4.1). Second, in the initial *Relaxation* iterations, the estimated barycenter can be arbitrarily far from the final, optimized barycenter. Thus, for these early iterations, it can be beneficial to relax the level of accuracy of the *Assignment* step, which is the main bottleneck and to progressively increase it as the barycenter converges to a solution. Progressivity can be injected at two levels: by controlling the accuracy of the distance estimation itself (Section 5.4.2) and the resolution of the input diagrams (Section 5.4.3). Our framework is easily parallelizable (Section 5.4.4) and the progressivity allows to design an interruptible algorithm, capable of respecting running time constraints (Section 5.4.5). The same ideas presented in these sections can also be applied to the design of a progressive and interruptible clustering algorithm (Section 5.5).

5.4 Progressive Barycenters

5.4.1 Auctions with Price Memorization

The *Assignment* step of the Wasserstein barycenter computation (line 5, 1) can be resolved in principle with any of the existing techniques for Wasser-

stein distance estimation [Mun57, Mor10, KMN16, SPCT18b]. Among them, the Auction based approach [Ber81, KMN16] (Section 5.2.1) is particularly relevant as it can compute very efficiently approximations with on-demand accuracy.

In the following, we consider that each distance computation involves *augmented* diagrams (Section 2.4, page 27) . Each input diagram $\mathcal{D}(f_i) \in \mathcal{F}$ is then considered as a set of *bidders* while the output barycenter $\mathcal{D}^*$ contains the *objects* to purchase. Each input diagram $\mathcal{D}(f_i)$ maintains its own list of prices p_b^i for the purchase of the objects $b \in \mathcal{D}^*$ by the bidders $a \in \mathcal{D}(f_i)$. The search by a bidder for the two most valuable objects to purchase is accelerated with space partitioning data structures, by using a kd-tree and a lazy heap respectively for the off- and on-diagonal points [KMN16] (these structures are re-computed for each *Relaxation*). Thus, the output barycenter $\mathcal{D}^*$ maintains only one kd-tree and one lazy heap for this purpose. Since Wasserstein distances are only approximated in this strategy, we suggest to relax the overall stopping condition (Algorithm 1) and stop the iterations after two successive increases in Fréchet energy (Equation 5.3), as commonly done in gradient descent optimizations. In the rest of the paper, we call the above strategy the *Auction barycenter algorithm [TMMH14]+[KMN16]*, as it just combines the algorithms by Turner et al. [TMMH14] and Kerber et al. [KMN16].

However, this usage of the auction algorithm results in a complete reboot of the entire sequence of auction rounds upon each *Relaxation*, while in practice, for the barycenter problem, the output assignments ϕ_i may be very similar from one *Relaxation* iteration to the next and thus could be re-used as initial solutions. For this, we introduce a mechanism that we call *Price Memorization*, which consists in initializing the prices p_b^i for each bidder $a \in \mathcal{D}(f_i)$ to the prices obtained at the previous *Relaxation* iteration (instead of 0). This has the positive effect of encouraging bidders to bid in priority on objects which were previously assigned to them, hence effectively re-using the previous assignments as an initial solution. This memorization makes most of the early auction rounds become unnecessary in practice, which enables to drastically reduce their number, as detailed in the following.

5.4.2 Accuracy-driven progressivity

The reference algorithm for Wasserstein barycenter computation (Algorithm 1) can also be interpreted as a variant of gradient descent

[TMMH14]. For such methods, it is often observed that approximations of the gradient, instead of exact computations, can be sufficient in practice to reach convergence. This observation is at the basis of our progressive strategy. Indeed, in the early *Relaxation* iterations, the barycenter can be arbitrarily far from the converged result and achieving a high accuracy in the *Assignment* step (line 5) for these iterations is often a waste of computation time. Therefore we introduce a mechanism that progressively increases the accuracy of the *Assignment* step along the *Relaxation* iterations, in order to obtain more accuracy near convergence.

To achieve this, inspired by the internals of the auction algorithm, we apply a *global* ε-scaling , where we progressively decrease the value of ε, but only at the end of each *Relaxation*. Combined with *Price Memorization* , this strategy enables us to perform *only one* auction round per *Assignment* step. As large ε values accelerate auction convergence at the price of accuracy, this strategy effectively speeds up the early *Relaxation* iterations and leads to more and more accurate auctions, and thus assignments, along the *Relaxation* iterations.

In practice, we divide ε by 5 after each *Relaxation*, as suggested by Bertsekas [Ber81] in the case of the regular auction algorithm (Section 5.2.1). Moreover, to guarantee precise final barycenters (obtained for small ε values), we modify the overall stopping condition to prevent the algorithm from stopping if ε is larger than 10^{-5} of its initial value.

5.4.3 Persistence-driven progressivity

In practice, the persistence diagrams of real-life data sets often contain a very large number of critical point pairs of low persistence. These numerous small pairs correspond to noise and are often meaningless for the applications. However, although they individually have only little impact on Wasserstein distances (Equation 2.2), their overall contributions may be non-negligible. To account for this, we introduce in this section a persistence-driven progressive mechanism, which progressively inserts in the input diagrams critical point pairs of decreasing persistence. This focuses the early *Relaxation* iterations on the most salient features of the data, while considering the noisy ones last. In practice, this encourages the optimization to explore more relevant local minima of the Fréchet energy (Equation 5.3) that favor persistent features.

Given an input diagram $\mathcal{D}(f_i)$, let $\mathcal{D}_\rho(f_i)$ be the subset of its points with persistence higher than ρ: $\mathcal{D}_\rho(f_i) = \{a \in \mathcal{D}(f_i) \mid y_a - x_a > \rho\}$.

To account for persistence-driven progressivity, we run our barycenter algorithm (with *Price Memorization*, Section 5.4.1, and accuracy-driven progressivity, Section 5.4.2) by initially considering as an input the diagrams $\mathcal{D}_\rho(f_i)$. After each *Relaxation* iteration (2, line 10), we decrease ρ such that $|\mathcal{D}_\rho(f_i)|$ does not increase by more than 10% (to progress at uniform speed) and such that ρ does not get smaller than $\sqrt{\tau\varepsilon}$ (we set $\tau = 4$ to replicate locally Bertsekas's suggestion for ε setting, Section 5.2.1). Initially, ρ is set to half of the maximum persistence found in the input diagrams. Along the *Relaxation* iterations, the input diagrams $\mathcal{D}_\rho(f_i)$ are progressively populated, which yields the introduction of new points in the barycenter $\mathcal{D}^*$, which we initialize at locations selected uniformly among the newly introduced points of the N inputs. This strategy enables to *distribute* among the inputs the initialization of the new barycenter points. The corresponding prices are initialized with the minimum price p_b^i found for the objects $b \in \mathcal{D}^*$ at the previous iteration.

5.4.4 Parallelism

Our progressive framework can be trivially parallelized as the most computationally demanding task, the *Assignment* step (Algorithm 2), is independent for each input diagram $\mathcal{D}(f_i)$. The space partitioning data structures used for proximity queries to $\mathcal{D}^*$ are accessed independently by each bidder diagram. Thus, we parallelize our approach by running the *Assignment* step in n_t independent threads.

5.4.5 Computation time constraints

Our persistence-driven progressivity (Section 5.4.3) focuses the early iterations of the optimization on the most salient features, while considering the noisy ones last. However, as discussed before, low persistence pairs in the input diagrams are often considered as meaningless in the applications. This means that our progressive framework can in principle be *interrupted* before convergence and still provide a meaningful result.

Let t_{max} be a user defined time constraint. We first progressively introduce points in the input diagrams $\mathcal{D}_\rho(f_i)$ and perform the *Relaxation* iterations for the first 10% of the time constraint t_{max}, as described in Section 5.4.3. At this point, the optimized barycenter $\mathcal{D}^*$ contains only a fraction of the points it would have contained if computed until convergence. To guarantee a precise output barycenter, we found that continuing the *Relaxation* iterations for the remaining 90% of the time, without

introducing new persistence pairs, provided the best results. In practice, in most of our experiments, we observed that this second optimization part fully converged even before reaching 90% of the computation time constraint. Algorithm 2 summarizes our overall approach for Wasserstein barycenters of persistence diagrams, with price memorization, progressivity, parallelism and time constraints. Several iterations of our algorithm are illustrated on a toy example in Figure 5.4.

Algorithm 2: Our overall algorithm for Progressive Wasserstein Barycenters.

Input : Set of diagrams $\mathcal{F} = \{\mathcal{D}(f_1), \mathcal{D}(f_2), \ldots, \mathcal{D}(f_N)\}$
Time constraint t_{max}
Output : Wasserstein barycenter $\mathcal{D}^*_\rho$

1: $\mathcal{D}^*_\rho \leftarrow \mathcal{D}_\rho(f_i)$ // with i randomly chosen in $[1, N]$
2: **while** the Fréchet energy decreases **do**
3: // Relaxation start
4: **for** $i \in [1, N]$ **do**
5: // In parallel **// Section 5.4.4**
6: $\phi_i \leftarrow Assignment(\mathcal{D}_\rho(f_i), \mathcal{D}^*_\rho)$ **// Section 5.4.1**
7: **end for**
8: $\mathcal{D}^*_\rho \leftarrow Update(\phi_1, \ldots, \phi_n)$ // arithmetic means in birth/death space
9: $EpsilonScaling()$ **// Section 5.4.2**
10: **if** $t < 0.1 \times t_{max}$ **then** $PersistenceScaling()$ **// Section 5.4.3**
11: **else if** $t >= t_{max}$ **then return** $\mathcal{D}^*_\rho$ **// Section 5.4.5**
12: // Relaxation end
13: **end while**
14: **return** $\mathcal{D}^*_\rho$

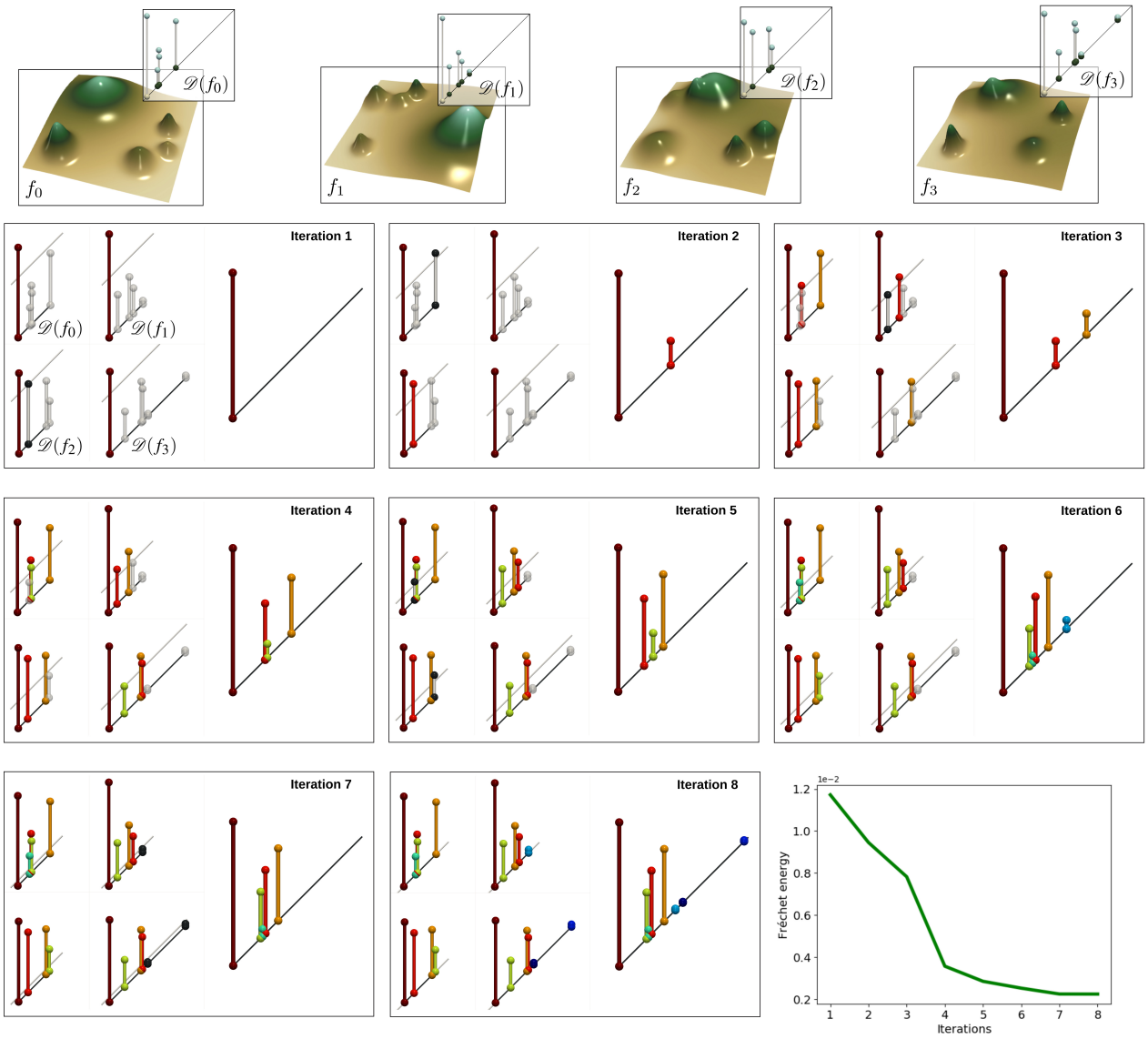


Figure 5.4 – *Iterations of our progressive algorithm on a randomly generated toy example. The top row presents the input toy ensemble and the corresponding persistence diagrams. Then, for each iteration, the four input diagrams are showed on the left and the corresponding barycenter on the right. In the input diagrams, under the persistence threshold (white line), the persistence pairs in transparent white have not been added yet to the assignment problem. The colors of the remaining pairs indicate their matched pairs in the barycenter. Initially, only the largest pair is present in each diagram, and the barycenter is identical to the first thresholded input diagram. The persistence threshold is progressively decreased along the relaxation process, allowing more persistence pairs to be added in the diagrams. At each iteration, one Update step occurs, followed by a decreasing of the persistence threshold, and one Assignment step, whose result is displayed. As such, persistence pairs in the input diagrams can be matched to the diagonal, and are in this case represented with a white bar with black spheres. This event takes place at iterations 1, 2, 3, 5, and 7, and always precedes the addition of a pair in the barycenter. On the contrary, pairs can be deleted from the barycenter if they are not matched to any pair anymore in the input diagrams, such as the blue pair at iteration 6. The evolution of the Fréchet energy (bottom right) illustrates the fast convergence of the barycenter towards a local minimizer.*

5.5 Application to Ensemble Topological Clustering

This section presents an application of our progressive framework for Wasserstein barycenters of persistence diagrams to the clustering of the members of ensemble data sets. Since it focuses on persistence diagrams, this strategy enables to group together ensemble members that have the same topological profile, hence effectively highlighting the main trends found in the ensemble in terms of features of interest.

The k-means is a popular algorithm for the clustering of the elements of a set, if distances and barycenters can be estimated on this set. The latter is efficiently computable for persistence diagrams thanks to our novel progressive framework and we detail in the following how to revisit the k-means algorithm to make it use our progressive barycenters as estimates of the centroids of the clusters.

The k-means is an iterative algorithm, which highly resembles barycenter computation algorithms (Section 5.2.2), where each *Clustering* iteration is composed itself of two sub-routines: *(i) Assignment* and *(ii) Update*. Initially, k cluster centroids $\mathcal{D}_j^*$ ($j \in [1, k]$) are initialized on k diagrams $\mathcal{D}(f_i)$ of the input set $\mathcal{F}$. For this, in practice, we use the *k-means++* heuristic described by Celebi et al. [CKV13], which aims at maximizing the distance between centroids. Then, the *Assignment* step consists in assigning each diagram $\mathcal{D}(f_i)$ to its closest centroid $\mathcal{D}_j^*$. This implies the computation, for each diagram $\mathcal{D}(f_i)$ of its Wasserstein distance W_2 to all the centroids $\mathcal{D}_j^*, j \in [1, k]$. For this step, we estimate these pairwise distances with the Auction algorithm run until convergence ($\gamma = 0.01$, Section 5.2.1). In practice, we use the *accelerated k-means* strategy described by Elkan [Elk03], which exploits the triangle inequality between centroids to skip, given a diagram $\mathcal{D}(f_i)$, the computation of its distance to centroids $\mathcal{D}_j^*$ which cannot be the closest. Next, the *Update* step consists in updating each centroid's location by placing it at the barycenter (with Algorithm 2) of its assigned diagrams $\mathcal{D}(f_i)$. The algorithm continues these *Clustering* iterations until convergence, *i.e.* until the assignments between the diagrams $\mathcal{D}(f_i)$ and the k centroids $\mathcal{D}_j^*$ do not evolve anymore, hence yielding the final clustering.

From our experience, the *Update* step of a *Clustering* iteration is by far the most computationally expensive. To speed up this stage in practice, we derive a strategy that is similar to our approach for barycenter approximation: we reduce the computation load of each *Clustering* iteration and progressively increase their accuracy along the optimization. This strat-

Table 5.1 – *Comparison of running times (in seconds, 1 thread) for the estimation of Wasserstein barycenters of Persistence diagrams. N and $\#_{\mathcal{D}(f_i)}$ respectively stand for the number of members in the ensemble and the average size of the input persistence diagrams.*

Data set	N	$\#_{\mathcal{D}(f_i)}$	Sinkhorn [LCO18]	Munkres [TMMH14]+[SPCT18b]	Auction [TMMH14]+[KMN16]	**Ours**	Speedup
Starting Vortex (Figure 5.10)	12	36	40.98	0.06	0.67	0.28	0.2
Vortex Street (Figure 5.9)	45	14	54.21	0.14	0.47	0.23	0.6
Isabel (3D) (Figure 5.2)	12	1,337	1,070.57	> 24H	377.42	82.95	4.5
Sea Surface Height (Figure 5.11)	48	1,379	4,565.37	> 24H	949.08	75.90	12.5
Gaussians (Figure 5.5)	100	2,078	7,499.33	> 24H	8,975.60	785.53	11.4

egy is motivated by a similar observation: early centroids are quite different from the converged ones, which motivates an accuracy reduction in the early *Clustering* iterations of the algorithm. Thus, for each *Clustering* iteration, we use a single round of auction with price memorization (Section 5.4.1), and a *single* barycenter update (*i.e.* a single *Relaxation* iteration, 2). Overall, only one global ε-scaling (Section 5.4.2) is applied at the end of each *Clustering* iteration. This enhances the k-means algorithm with accuracy progressivity. If a diagram $\mathcal{D}(f_i)$ migrates from a cluster j to a cluster l, the prices of the objects of $\mathcal{D}_l^*$ for the bidders of $\mathcal{D}(f_i)$ are initialized to 0 and we run the auction algorithm between $\mathcal{D}(f_i)$ and $\mathcal{D}_l^*$ until the pairwise ε value matches the global ε value, in order to obtain prices for $\mathcal{D}(f_i)$ which are comparable to the other diagrams. Also, we apply persistence-driven progressivity (Section 5.4.3) by adding persistence pairs of decreasing persistence in each diagram $\mathcal{D}(f_i)$ along the *Clustering* iterations. Finally, a computation time constraint can also be provided, as described in Section 5.4.5. Results of our clustering scheme are presented in Section 5.6.3.

5.6 Results

This section presents experimental results obtained on a computer with two Xeon CPUs (3.0 GHz, 2x4 cores), with 64GB of RAM. The input persistence diagrams were computed with the FTM algorithm [GFJT17, TFL+17]. We implemented our approach in C++, as TTK modules.

Our experiments were performed on a variety of simulated and acquired 2D and 3D ensembles, taken from Favelier et al. [FFST18]. The data is described in Appendix A. The *Gaussians* ensemble contains 100 2D synthetic noisy members, with 3 patterns of Gaussians (Figure 5.5). The considered features of interest in this example are the maxima. The *Vortex Street* ensemble (Figure 5.9) includes 45 runs of a 2D simulation of flow

Table 5.2 – *Running times (in seconds) of our approach (run until convergence) with 1 and 8 threads. N and* $\#_{\mathcal{D}(f_i)}$ *stand for the number of members in the ensemble and the average size of the diagrams.*

Data set	N	$\#_{\mathcal{D}(f_i)}$	1 thread	8 threads	Speedup
Starting Vortex (Figure 5.10)	12	36	0.28	0.19	1.5
Vortex Street (Figure 5.9)	45	14	0.23	0.10	2.3
Isabel (3D) (Figure 5.2)	12	1,337	82.95	31.75	2.6
Sea Surface Height (Figure 5.11)	48	1,379	75.90	19.40	3.9
Gaussians (Figure 5.5)	100	2,078	785.53	117.91	6.6

turbulence behind an obstacle. The considered scalar field is the curl orthogonal component, for 5 fluids of different viscosity. In this application, salient extrema are typically considered as reliable estimations of the center of vortices. Thus, each run is represented by two diagrams, processed independently by our algorithms: one for the $(0, 1)$ pairs (involving minima) and one for the $\big((d-1), d\big)$ pairs (involving maxima). The *Starting Vortex* ensemble (Figure 5.10) includes 12 runs of a 2D simulation of the formation of a vortex behind a wing, for 2 distinct wing configurations. The considered data is also the curl orthogonal component and diagrams involving minima and maxima are also considered. The *Isabel* data set (Figure 5.2) is a volume ensemble of 12 members, showing key time steps (formation, drift and landfall) in the simulation of the Isabel hurricane [Sci04]. In this example, the eyewall of the hurricane is typically characterized by high wind velocities, well captured by velocity maxima. Thus we only consider diagrams involving maxima. Finally, the *Sea Surface Height* ensemble (Figure 5.11) is composed of 48 observations taken in January, April, July and October 2012 (https://ecco.jpl.nasa.gov/products/all/). Here, the features of interest are the center of eddies, which can be reliably estimated with height extrema. Thus, both the diagrams involving the minima and maxima are considered and independently processed by our algorithms. Unless stated otherwise, all results were obtained by considering the Wasserstein metric W_2 based on the original pointwise metric (Equation 2.1) without geometrical lifting (*i.e.* $\alpha = 0$, Section 2.4).

5.6.1 Time performance

Table 5.1 evaluates the time performance of our progressive framework when run until convergence (*i.e.* no computation time constraint). This table also provides running times for 3 alternatives. The column, *Sinkhorn*, provides the timings obtained with a Python CPU implementation kindly

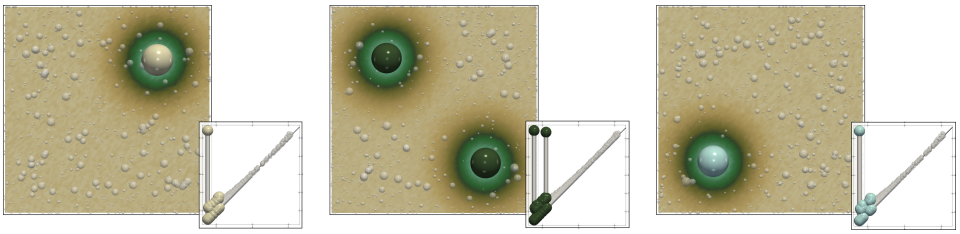


Figure 5.5 – *Clustering the* Gaussians *ensemble. From left to right, pointwise mean and Wasserstein barycenter for each of the identified clusters (t_{max} : 10s.) with geometrical lifting ($\alpha = 0.65$).*

provided by Lacombe et al. [LCO18], for which we used the recommended parameter values (entropic term: $10^{-1}/\#_{\mathcal{D}(f_i)}$ heat map resolution: 100^2). Note that this approach casts the problem as an Eulerian transport optimization under an entropic regularization term. Thus, it optimizes for a convex functional which is considerably different from the Fréchet energy considered in our approach (Equation 5.3). Overall, these aspects, in addition to the difference in programming language, challenge direct comparisons and we only report running times for completeness. The columns *Munkres*, noted [TMMH14]+[SPCT18b], and *Auction*, noted [TMMH14]+[KMN16], report the running times of our own C++ implementation of Turner's algorithm [TMMH14] where distances are respectively estimated with the exact method by Soler et al. [SPCT18b] and our own C++ implementation of the auction-based approximation by Kerber et al. [KMN16] (with kd-tree and lazy heap, run until convergence, $\gamma = 0.01$).

As discussed in Section 3.1.1, the *Sinkhorn* method [LCO18] considers an intermediate heat map representation of persistence diagrams and does not explicitly produce a diagram on its output, which limits its applicability. Moreover, the geometrical lifting described in Section 2.4, which is particularly useful in our applications, is difficult to express in this Eulerian setting, where 5-dimensional histograms would be needed, which is impractical. In contrast, the *Munkres* and *Auction* approaches produce explicit barycenters and optimize the same functional as our approach (Equation 5.3), which allows direct comparison.

As predicted, the cubic time complexity of the *Munkres* algorithm makes it impractical for barycenter estimation, as the computation completed within 24 hours for only two ensembles. The *Auction* approach is more practical but still requires up to hours to converge for the largest data sets. In contrast, our approach converges in sequential in less than 15 minutes at most. The column *Speedup* reports the gain obtained with our method against the fastest of the two explicit alternatives, *Munkres* or

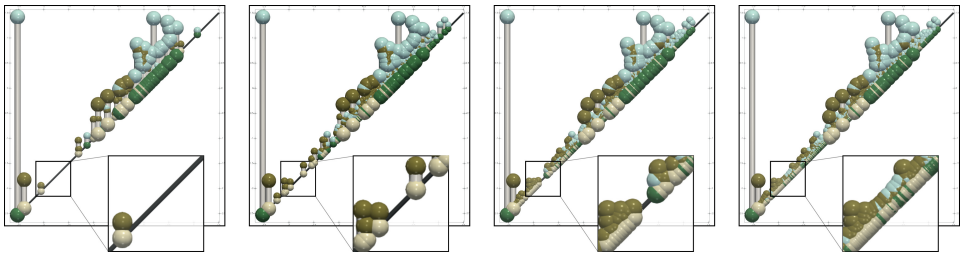


Figure 5.6 – *Interrupted Wasserstein barycenters for one cluster of the Sea Surface Height ensemble with different computation time constraints. From left to right : 0.1 s., 1 s., 10 s., and full convergence (21 s.).*

Auction. For ensembles of realistic size, this speedup is about an order of magnitude. As reported in Table 5.2, our approach can be trivially parallelized with OpenMP by running the *Assignment* step (Algorithm 2) in independent threads (Section 5.4.4). As the size of the input diagrams $\mathcal{D}(f_i)$ may strongly vary within an ensemble, this trivial parallelization may result in load imbalance among the threads, impairing parallel efficiency. In practice, this strategy still provides reasonable speedups, bringing the computation down to a couple of minutes at most.

5.6.2 Barycenter quality

Table 5.3 compares the Fréchet energy (Equation 5.3) of the converged barycenters for our method and the *Auction barycenter* alternative [TMMH14]+[KMN16]. The Wasserstein distances between the results of the two approaches are provided in Table 5.4. Note that the two algorithms (our progressive method and the *Auction* approach) may converge to different local minimas of the Fréchet energy, which can be arbitrarily distant from each other while being of similar quality (visually and quantitatively in terms of Fréchet energy). Overall, the distances between the two outputs are comparable to the minimum distances observed within each ensemble. This indicates that the barycenters resulting from the two approaches are rather close, in particular relatively to the maximum observed distances.

The Fréchet energy has been precisely evaluated with an estimation of Wasserstein distances based on the Auction algorithm run until convergence ($\gamma = 0.01$). While the actual values for this energy are not specifically relevant (because of various data ranges), the ratio between the two methods indicates that the local minima approximated by both approaches are of comparable numerical quality, with a variation of 2% in energy at most. Figure 5.7 provides a visual comparison of the converged

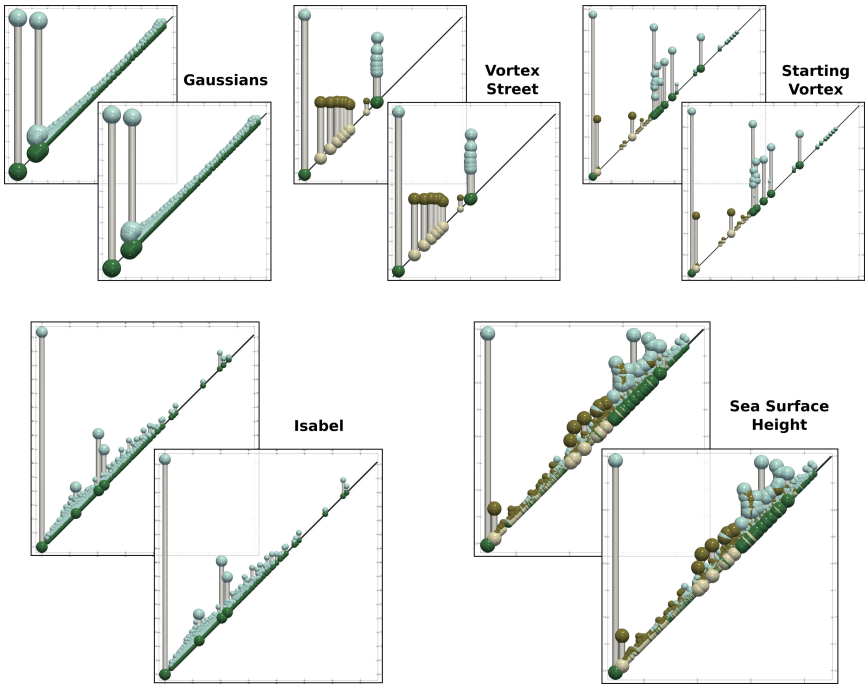


Figure 5.7 – *Visual comparison between the converged Wasserstein barycenters obtained with the* Auction barycenter *algorithm [TMMH14]+[KMN16](on the back) and our approach (front) for one cluster (Section 5.6.3) of each ensemble data set. Differences are barely noticeable, and only for small persistence pairs.*

Wasserstein barycenters obtained with the *Auction barycenter* alternative [TMMH14]+[KMN16] and our method, for one cluster of each of our data sets (Section 5.6.3). This figure shows that differences are barely noticeable and only involve pairs with low persistence, which are often of small interest in the applications.

Figure 5.8 compares the convergence rates of the *Auction barycenter* [TMMH14]+[KMN16](red) to three variants of our framework: without (blue) and with (orange) persistence progressivity and with time computation constraints (green, complete computations for increasing time constraints). It indicates that our approach based on *Price Memorization* and single auction round (Sections 5.4.1 and 5.4.2, blue curve) already sub-

Table 5.3 – *Comparison of Fréchet energy (Equation 5.3) at convergence between the* Auction barycenter *method [TMMH14]+[KMN16] and our approach.*

Data set	N	$\#_{\mathcal{D}(f_i)}$	Auction [TMMH14]+[KMN16]	Ours	**Ratio**
Starting Vortex (Figure 5.10)	45	14	112,787.0	112,642.0	1.00
Vortex Street (Figure 5.9)	12	36	415.1	412.5	0.99
Isabel (3D) (Figure 5.2)	12	1,337	2,395.6	2,337.1	0.98
Sea Surface Height (Figure 5.11)	48	1,379	7.2	7.1	0.99
Gaussians (Figure 5.5)	100	2,078	39.4	39.0	0.99

Table 5.4 – *Wasserstein distances (column W_2) between the barycenters computed with our approach and the Auction approach [TMMH14]+[KMN16], for all ensembles. For comparison, the maximal and minimal distances observed between diagrams for each ensemble are also reported.*

Data set	N	$\#_{\mathcal{D}(f_i)}$	Maximal distance	Minimal distance	W_2
Starting Vortex	45	14	471.52	9.27	25.29
Vortex Street	12	36	25.07	0.02	1.38
Isabel (3D)	12	1,337	72.96	19.84	15.79
Sea Surface Height	48	1,379	2.18	0.76	0.69
Gaussians	100	2,078	5.30	1.52	0.75

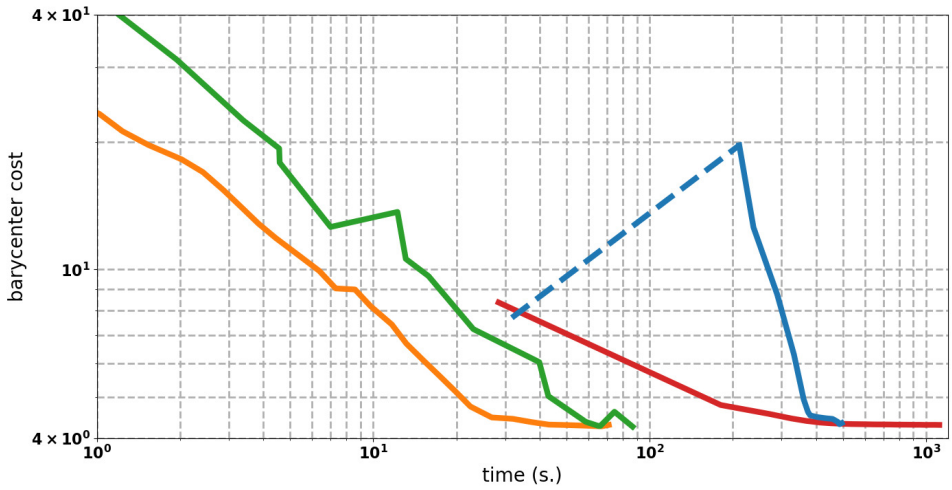


Figure 5.8 – *Comparison of the evolution of the Fréchet energy (log scale,* Sea Surface Height*, maximum diagrams), for the* Auction barycenter *method [TMMH14]+[KMN16](red) and 3 variants of our approach: without (blue) and with (orange) persistence progressivity, and with time constraints (green). Persistence-driven progressivity drastically accelerates convergence.*

stantially accelerates convergence (the first iteration, dashed, is performed with a large ε and thus induces a high energy). Interestingly, persistence-driven progressivity (orange) provides the most important gains in convergence speed. The number of *Relaxation* iterations is larger for our approach (43, orange) than for the *Auction barycenter* method (23, red), which emphasizes the low computational effort of each of our iterations. Finally, when the *Auction barycenter* method completed its first *Relaxation* iteration (leftmost red point), our persistence-driven progressive algorithm already achieved 80% of its iterations, resulting in a Fréchet energy almost twice smaller. The quality of the barycenters obtained with the interruptible version of our approach (Section 5.4.5) is illustrated in Figures 5.2 and 5.6 for varying time constraints. As predicted, features of decreasing persistence progressively appear in the diagrams, while the most salient features

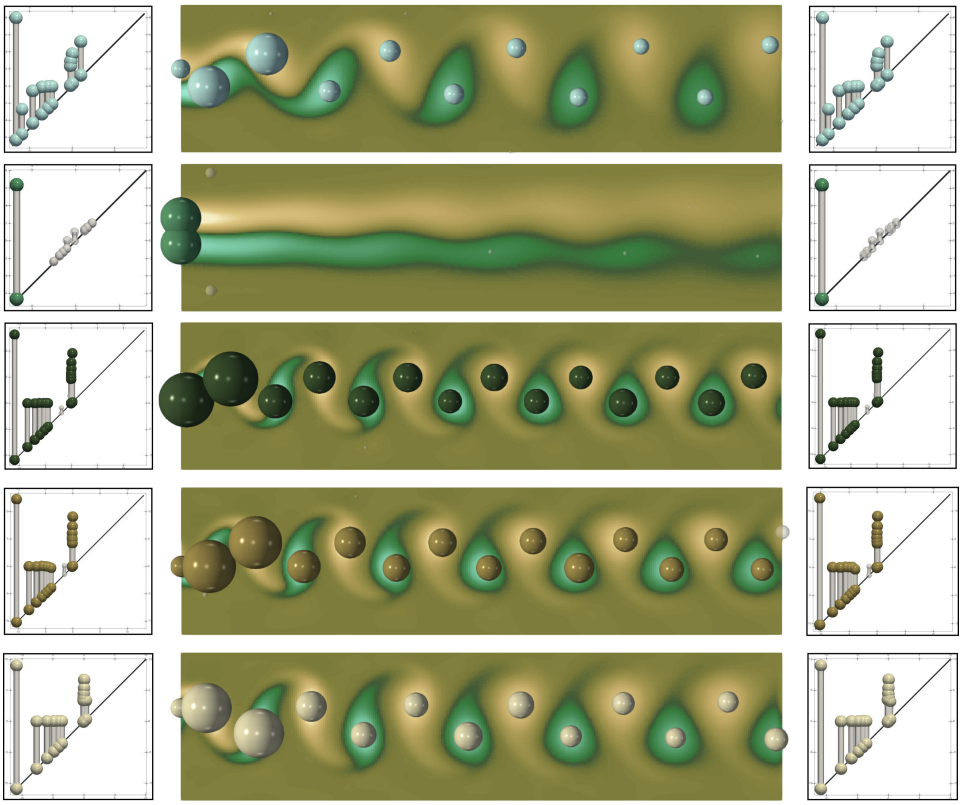


Figure 5.9 – *Clusters automatically identified by our topological clustering (t_{max}: 10 seconds) on the* Vortex Street *dataset. From top to bottom: pointwise mean of each cluster. Left: Centroids computed by our interruptible clustering algorithm. Right: Wasserstein barycenters of the clusters, computed by our progressive algorithm run until convergence. Differences are visually indistinguishable. Barycenter extrema are scaled in the domain by persistence (spheres).*

are accurately represented for very small constraints, allowing for reliable estimations within interactive times (below a second).

5.6.3 Ensemble visual analysis with Topological Clustering

In the following, we systematically set a time constraint t_{max} of 10 seconds. To facilitate the reading of the diagrams, each pair with a persistence smaller than 10% of the function range is shown in transparent white, to help visually discriminate salient features from noise. Figure 5.5 shows the clustering of the *Gaussians* ensemble by our approach. This synthetic ensemble exemplifies the motivation for the geometrical lifting (Section 2.4). The first and third clusters both contain a single Gaussian, resulting in diagrams with a single persistent feature, but located in drastically different areas of the domain $\mathcal{M}$. Thus, the diagrams of these two clusters would be indistinguishable for the clustering algorithm if geometrical lifting was not considered. If feature location is important for

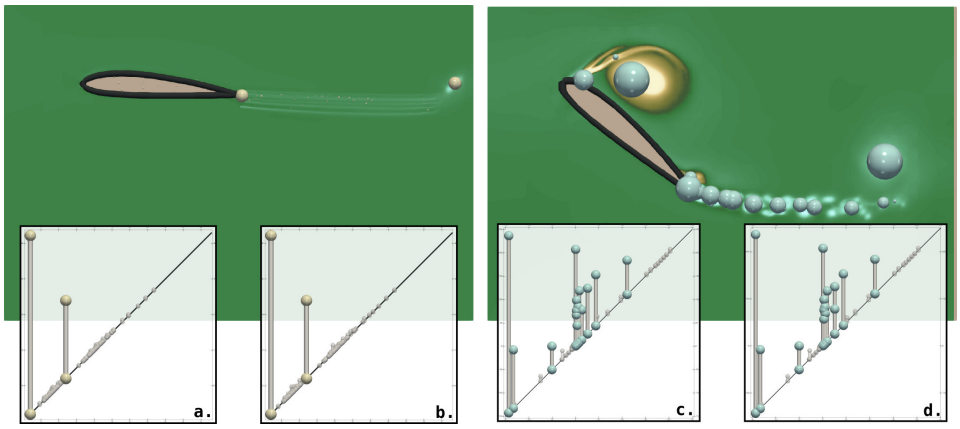


Figure 5.10 – *Clusters automatically identified by our topological clustering (t_{max} : 10s.) for the* Starting Vortex *dataset. Left insets (a, c): Centroids computed by our interruptible clustering algorithm. Right insets (b, d): Wasserstein barycenters of the clusters, computed by our progressive algorithm run until convergence. Differences are visually indistinguishable. Top: pointwise mean of each cluster, with barycenter extrema scaled by persistence (spheres).*

the application, our approach can be adjusted thanks to geometrical lifting (Section 2.4). For the *Gaussians* ensemble, this makes our clustering approach compute the correct clustering. Moreover, taking the geometry of the critical points into account allows us to represent in $\mathcal{M}$ the extrema involved in the Wasserstein barycenters (spheres, scaled by persistence, Figure 5.5) which allows user to have a visual feedback in the domain of the features representative of the set of scalar fields. Geometrical lifting is particularly important in applications where feature location bears a meaning, such as the Isabel ensemble (Figure 5.2 (f)). For this example, our clustering algorithm with geometrical lifting automatically identifies the right clusters, corresponding to the three states of the hurricane (formation, drift and landfall). For the remaining examples, geometrical lifting was not necessary ($\alpha = 0$). For the *Vortex Street* ensemble (Figure 5.9), our approach manages to automatically identify the correct clusters, precisely corresponding to the distinct viscosity regimes found in the ensemble. Note that the centroids computed by our topological clustering algorithm with a time constraint of 10 seconds (left) are visually indistinguishable from the Wasserstein barycenters of each cluster, computed after the fact with our progressive algorithm run until convergence (right). This indicates that the centroids provided by our topological clustering are reliable and can be used to visually represent the features of interest in the ensemble. In particular, for the *Vortex Street* example, these centroids enable the clear identification of the number and salience of the vortices: pairs

which align horizontally and vertically respectively denote minima and maxima of flow vorticity, which respectively correspond to clockwise and counterclockwise vortices. Figure 5.10 presents our results on the *Starting Vortex*, where our approach also automatically identifies the correct clustering, corresponding to two wing configurations. In this example, the difference in turbulence (number and strength of vortices) can be directly and visually read from the centroids returned by our algorithm (insets). Finally, Figure 5.11 shows our results for the *Sea Surface Height*, where our topological clustering automatically identifies four clusters, corresponding to the four seasons: winter (top left), spring (top right), summer (bottom left), fall (bottom right). As shown in the insets, each season leads to a visually distinct centroid diagram. In this example, as diagrams are larger, differences between the interrupted centroids (left) and the converged barycenters (right) become noticeable. However, these differences only involve pairs of small persistence, whose contribution to the final clustering reveal negligible in practice.

Overall, our approach provides the same clustering results than Favelier et al. [FFST18]: the returned clusterings are correct for both approaches, for all of the above data sets. However, once the input persistence diagrams are available, our algorithm computes within a time constraint of ten seconds only, while the approach by Favelier et al. requires up to hundreds of seconds (on the same hardware) to compute intermediate representations (*Persistence Maps*) which are not needed in our work.

5.7 Limitations

In our experiments, we focused on persistence diagrams which only involve extrema, as these often directly translate into features of interest in the applications. Although our approach can consider other types of persistence pairs (*e.g.* saddle-saddle pairs in 3D), from our experience, the interpretation of these structures is not obvious in practice and future work is needed to improve the understanding of these pairs in the applications. Thanks to the assignments computed by our algorithm, the extrema of the output barycenter can be embedded in the original domain (Figure 5.5 to 5.11). However, in practice a given barycenter extremum can be potentially assigned with extrema which are distant from each other in the ensemble members, resulting in its placement at an in-between location which may not be relevant for the application. Regarding the Fréchet energy, our

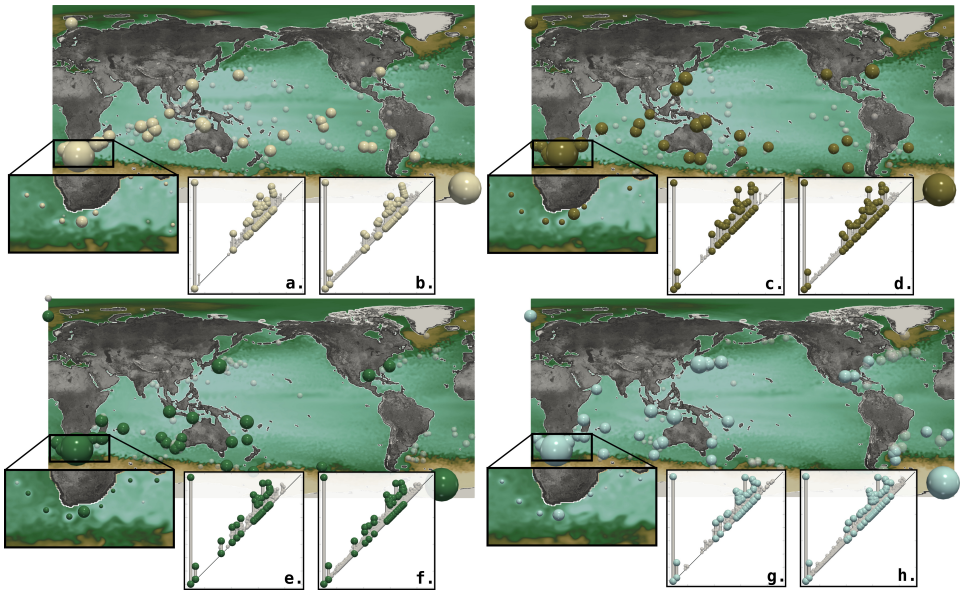


Figure 5.11 – *Clusters automatically identified by our topological clustering (t_{max} : 10s.) on the* Sea Surface Height *dataset (the greyscale represents a satellite view of continents). From left to right, top to bottom: pointwise mean of each cluster with barycenter extrema scaled by persistence (spheres). Left insets (a, c, e, g): Centroids computed by our interruptible clustering algorithm. Right insets (b, d, f, h): Wasserstein barycenters of the clusters, computed by our progressive algorithm run until convergence.*

experiments confirm the proximity of our approximated barycenters to actual local minima (Figure 5.7, Table 5.3). However, theoretical proximity bounds to these minima are difficult to formulate and we leave this for future work. Also, as it is the case for the original algorithm by Turner et al. [TMMH14], there is no guarantee that our solutions are global minimizers. For the clustering, we observed that the initialization of the *k*-means algorithm had a major impact on its outcome but we found that the *k-means++* heuristic [CKV13] provided excellent results in practice. Recent work on the vectorization of persistence diagrams with theoretical guarantees on cluster separability [RCL+21] could be instrumental to obtain theoretical guarantees on our clustering approach. Finally, when the geometrical location of features in the domain has a meaning for the applications, the geometrical lifting coefficient (Section 2.4) must be manually adjusted by the user on a per application basis, which involves a trial and error process. However, our interruptible approach greatly helps in this process, as users can perform such adjustments at interactive rates.

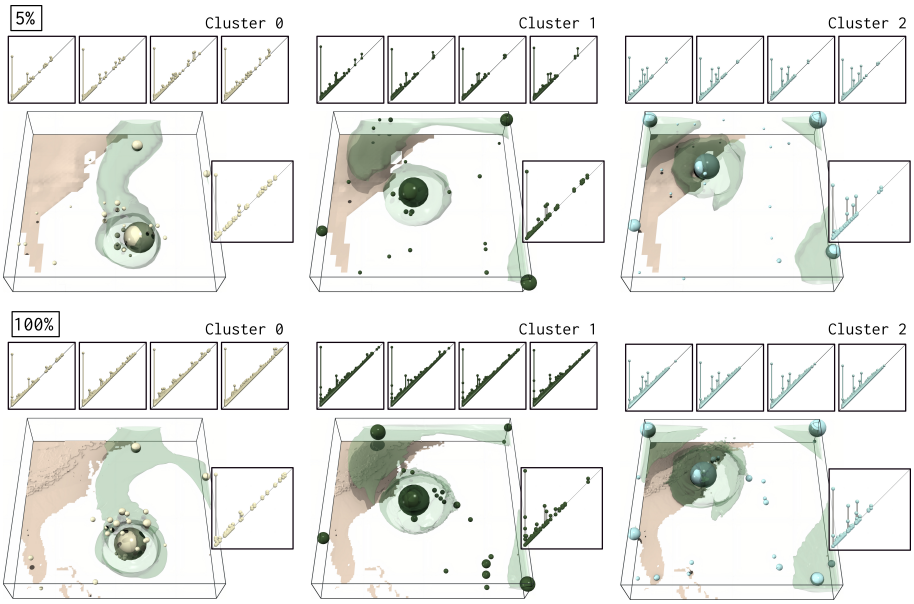


Figure 5.12 – *Topological clustering of the Isabel ensemble dataset. Progressive persistence diagrams (top, interrupted at* 5% *of computation) are used as an input to our clustering approach (constrained to* 1 *second of computation). This time-constrained ensemble clustering yields the same, correct classification (one color per cluster, from left to right) as the one returned with the exact diagrams (bottom).*

5.8 Overall time-constrained pipeline

Figure 5.12 illustrates the interest of our progressive barycenter computation for the control of the run time of batch-mode analysis pipelines. We consider the Isabel ensemble [Sci04, TTK20] (12 members illustrating 3 hurricane behaviors: formation, drift and landfall, Figure 5.12, left to right). The progressive approach detailed in Chapter 3 is used to generate a persistence diagram for each member, and is interrupted at a predefined threshold of computation time (Figure 5.12, top). Then, our progressive clustering algorithm (Section 5.5) is used to cluster these diagrams (with a time constraint of one second). Overall, this results in a topological clustering pipeline whose time execution is fully controlled, from the feature extraction to their clustering. For reasonable computation thresholds (5% in Figure 5.12), this pipeline returns the same, correct classification (one color per cluster) as the one returned with the exact diagrams (bottom). Additionally, the diagram centroids (large diagrams in the figure) computed in the two cases are visually similar. This demonstrates that the main trends of an ensemble in terms of features (the main clusters) can still be estimated reliably, while additionally controlling the execution time of the clustering pipeline.

5.9 Summary

In this chapter, we presented an algorithm for the progressive approximation of Wasserstein barycenters of Persistence diagrams, with applications to the visual analysis of ensemble data. Our approach revisits efficient algorithms for Wasserstein distance approximation [Ber81, KMN16] in order to specifically extend previous work on barycenter estimation [TMMH14]. Our experiments showed that our strategy drastically accelerates convergence and reported an order of magnitude speedup against previous work, while providing barycenters which are quantitatively and visually comparable. The progressivity of our approach allows for the definition of an *interruptible* algorithm, enabling the estimation of reliable barycenters within interactive times. We presented an application to ensemble data clustering, where the obtained centroid diagrams provided key visual insights about the global feature trends of the ensemble. We combined this approach with the progressive computation of persistence diagrams presented in Chapter 3, to build an overall time-constrained data analysis pipeline.

An application use case

6

Contents

6.1 The VESTEC project 123

6.1.1 Purpose 123

6.1.2 Numerical simulations for urgent decision-making 124

6.1.3 Use cases 125

6.1.4 Challenges 128

6.2 Topological Data Analysis in VESTEC 129

6.2.1 Persistence diagrams for Data Reduction 129

6.2.2 In-Situ Computation 130

6.2.3 Statistical Analysis 131

6.3 Results: The Space Weather Use Case 131

6.3.1 *In-Situ* Computation of Persistence Diagrams 131

6.3.2 Wasserstein distances between diagrams 132

6.3.3 Topological Clustering 133

6.4 Conclusion 135

This chapter presents VESTEC, the European project in which this thesis took place and was funded. VESTEC stands for Visual Exploration and Sampling for Extreme Computing. The goal of the project is to build a platform dedicated to urgent supercomputing and interactive data analysis in order to quickly support the decision-making process during urgent catastrophic events, such as floods, fires, earthquakes, etc. On the practical side, it gave us the opportunity to work in collaboration with experts in high-performance computing (HPC), data visualization, and especially experts from applications fields. Part of the work presented in the previous chapters was integrated into the VESTEC data analysis workflow, in interaction with HPC environments and in the context of *in-situ* data analysis applied to real urgent applications.

6.1 The VESTEC project

6.1.1 Purpose

The purpose of the VESTEC project is to build a chain of simulation and analysis tools to support the decision-making process in cases of catastrophic events. Examples of such disaters are floods, fires, epidemics, earthquakes, . . . , *e.g.* events that require urgent organized responses. The project focuses on three specific application cases, that are detailed in Section 6.1.3. The VESTEC consortium is composed of academics and companies, regrouping experts in high-performance computing, data analysis and visualization, as well as experts in specific application fields (Section 6.1.3). The different partners are:

- The *DLR*, the German Aerospace Center, coordinator of the project and involved in the management of sensor data from satellites, as well as the development of immersive visualization software.
- The Edinburgh Parallel Computing Center (EPCC), a supercomputing center based at the University of Edinburgh.
- The Royal Institute of Technology (KTH), in Stockholm.
- Sorbonne Université, in Paris.
- Kitware, a company specialized in the development of Open-Source visualization software, notably ParaView.
- Intel, developer of the Open-Source ray tracing engine OSPRay.
- The Fondazione Bruno Kessler (FBK), based in Trento, Italy.
- The Université Paul Sabatier in Toulouse, developer of the weather forecast simulation code MESO-NH.
- Technosylva, a company based in California and Spain, and specialized in the development of software for wildfire analytics.

The complete system being built in the scope of the project combines a high-performance computing (HPC) environment to perform highly parallelized numerical simulations, together with data analysis tools and visualization solutions to explore the results. An illustration of the system architecture is showed in Figure 6.1.

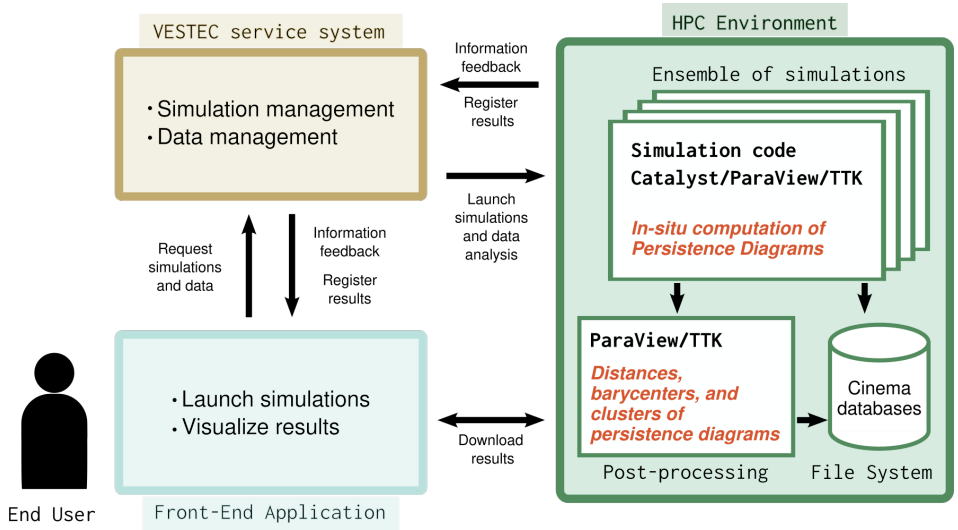


Figure 6.1 – *Simplified architecture of the VESTEC system*

6.1.2 Numerical simulations for urgent decision-making

Concretely, given a catastrophic situation, this system would give the opportunity to the decision makers (first responders, public health analysts, ...) to launch ensembles of numerical simulations on an available HPC cluster. Simulating a disaster enables to apprehend the possible outcomes of the phenomenon, as well as the efficiency of potential responses. In order to get the most insight, the space of input parameters of the simulations is typically densely explored, as some parameters of the simulation might be subject to uncertainty. For multi-physics, multi-parameters simulations, this generates a substantial ensemble of outputs. Trend analysis then must be performed to try and predict the most likely scenarios. The analysis of the results would give hints about the best strategic response to a disaster, in order to minimize damage and casualties.

In the case of wildfires, which is one of the application use cases of the VESTEC project, the decision makers are typically the firefighters in charge of containing the fire, ordering evacuations and assisting the victims. As firefighters get informed that a fire has begun to spread somewhere, they can launch en ensemble of simulations on an available HPC cluster somewhere in order to try and predict its evolution. The simulations take localization data into account, including information about the topography and vegetation of the region. They are also backed up with real time sensor data, including satellite imagery to follow the evolution of the fire, and by weather forecast simulators. Different set of parameters are used for the simulation runs, in order to predict the evolution of the simulation under different circumstances, for instance different responses

from the firefighters, changes in the wind direction, or the potential start of neighboring fires lighted by flying embers. The combination of these different parameters generates a substantial quantity of outputs. Data analysis is then performed to compute the likelihood of different scenarios, and identify main trends in the data. Then, visualization techniques are used to adequately convey the results to the end user, a fire analyst or a chief firefighter, with the aim of identifying the optimal response to the fire, and the highest risk regions that would need to be evacuated. The urgency of the situation requires that the whole pipeline be executed quickly, or at least be able to provide quick exploitable results in a first approximation.

6.1.3 Use cases

The VESTEC project is developed around three use cases: wildfire, mosquito-borne diseases , and space weather analysis.

Wildfire Analysis

Wildfire analysis has proven itself to be a key study field nowadays, as more extreme fires are observed each year all over the globe. These events, fueled by severe droughts induced by climate change, are becoming more and more dreadful. For instance, since the 1970s, California experienced more than a fivefold increase in annual wildfire extent [WAG+19]. Such large blazes cause tremendous material damage and numerous casualties, but they also constitute true ecological and environmental disasters. The Australian bushfires of 2019-2020 were deemed « *among the worst wildlife disasters in modern history* » in a report commissioned by the World Wide Fun for Nature, with over 3 billions animals killed or displaced (not counting invertebrates). In 2019, 9.060 km^2 of the Amazon rain forest were destroyed by wildfire, with drastic environmental impact on climate change due to gas emissions and destruction of vegetation.

In the fight against wildfires, prevention is key. However in the event where the disaster already started, numerical simulations can help a great deal to predict the evolution of the fire, in order to help organize evacuations or plan the response of firefighters. Numerical simulations can notably indicate whether the fire is likely to reach a crucial location, such as a natural reserve or a hospital, or whether a localized response from the firefighters is likely to stop the fire spreading. Wildfires can evolve rapidly, and their evolution is subject to changes of winds, to the topography and

Figure 6.2 – *Example of a dataset from the wildfire use case. From left to right: localization of the fire (California), the corresponding scalar field (time of arrival of the fire), and the associated persistence diagram.*

vegetation type, and to the response of firefighters. This motivates the use of highly parallelized simulation codes and high-performance data analysis tools to quickly explore different outcomes, in a context of urgency.

Technosylva is a company specialized in wildfire simulation and analysis. They commercialize a wildfire analysis software and service and work with firefighting departments, notably the California Department of Forestry and Fire Protection (CAL FIRE). They are responsible for the wildfire use case in VESTEC. Figure 6.2 shows an example of a typical dataset from this use case, generated with their software.

Mosquito-borne diseases

The repercussions of the COVID-19 pandemic has shown the importance of monitoring outbreaks of diseases and particularly of planning ahead for emergency responses. One of the most striking impressions of the pandemic was the feeling that responses were always a step behind the disease's evolution. Indeed, the long incubation period of the coronavirus disease made it hard to apprehend and counter its outbreak.

In the scope of the VESTEC project, the second use case is focused on the analysis of mosquito-borne diseases. In the recent years, Europe has seen an increasing number of cases of potent viral diseases carried by mosquitoes, such as Chikungunya, Dengue, or Zika. Two major Chikungunya outbreaks took place in Italy in 2007 and 2017 [CRM+20]. In 2015, a large Zika virus outbreak also started in Brazil and propagated through both Americas [ZSC+17], raising notable concerns about the 2016 Olympics Games in Rio de Janeiro. In Europe, the main culprit for the propagation of such diseases is the Tiger mosquito (*Aedes albopictus*), an

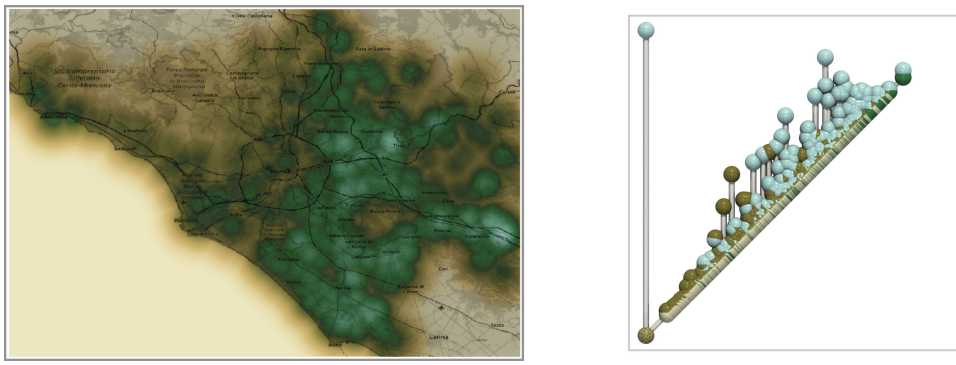


Figure 6.3 – *Example of a dataset from the mosquito-borne disease use case. The numerical simulation provides the Basic Reproduction Number of the epidemy, noted R_0, an epidemiological indicator which represents the expected number of secondary infections generated by one infectious individual. From left to right: the scalar field R_0 computed for the Zika virus disease in the region of Rome, and the associated persistence diagram.*

invasive mosquito species whose presence in southern Europe is gradually increasing. The Tiger mosquito can carry several diseases that it transmits from human to human. Because of its adaptation to urban areas and to colder climates, the population of *Ae. albopictus* is expected to drastically increase in the coming decades. Climate-change projections also suggest that this species, along with other species of disease-transmitting mosquito, will colonize new territories in the coming years [ADB+16], which makes the analysis of the propagation of mosquito-borne diseases an important topic. Numerical simulations are a natural and viable option to try and predict the propagation of such diseases and avoid an outbreak.

The Fondazione Bruno Kessler (FBK) is one of the top research institutes in Italy. The VESTEC partners from FBK are experts in the modeling and simulation of the spreading of epidemics. Since the start of the coronavirus crisis, they are notably in charge of running simulations of the disease evolution to assist the Italian government in taking public health decisions. They are responsible for the mosquito-borne diseases use case. Figure 6.3 shows an example of a typical dataset from this use case.

Space Weather

The term *space weather* describes the physical phenomena taking place in the Solar system, and particularly around the Earth, with emphasis on magnetic and radiative phenomena. Besides their theoretical interest, such events can have disastrous consequences. The electronic components presents in communication satellites are indeed particularly sensitive to geomagnetic storms, which are disturbances of the Earth's magnetic field

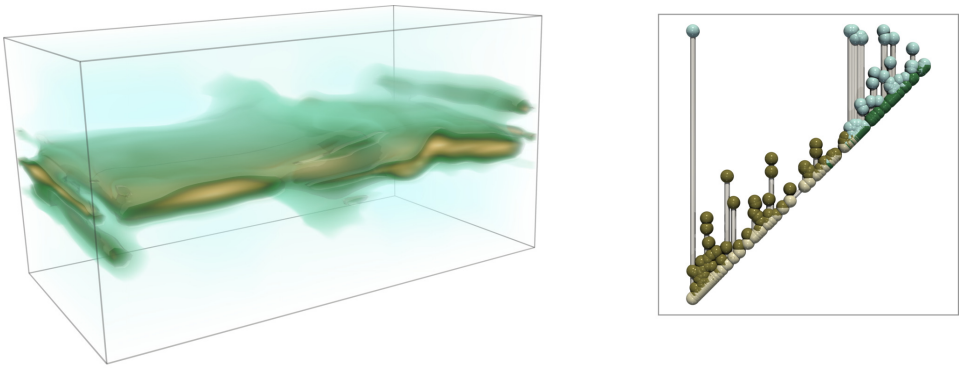


Figure 6.4 – *Example of a dataset from the space weather use case, a numerical simulation of magnetic reconnection. From left to right: the scalar field (magnitude of the magnetic field) and the associated persistence diagram.*

caused by solar winds interacting with the magnetosphere. Understanding space weather is thus of great importance to predict and circumvent satellite failures. In VESTEC, space weather is studied through the analysis of magnetic reconnection events occurring in the magnetosphere. Magnetic reconnection is a small-scale phenomenon that takes place in magnetized plasmas, where huge quantities of energy are converted from magnetic energy to radiative energy, effectively creating solar flares and magnetic storms, and thus potentially endangering satellites, astronauts, but also even power grids and pipelines on Earth [HC20]. Due to the complexity of the phenomenon, only standard theoric simulations are performed for the moment. Ensemble of 3D simulations of magnetic reconnection are performed under several different conditions.

In VESTEC, the referent partners on the space weather use case are physical simulation experts from the Royal Institute of Stockholm (KTH). Figure 6.4 shows an example of a typical dataset from this use case.

6.1.4 Challenges

A system such as VESTEC, dedicated to high performance computing for urgent decision-making, faces various challenges. First, on the simulation side, the system must be flexible enough for the user to be able to precisely and rapidly describe the ensemble of simulations to launch, and to run them on an available HPC cluster. On the other hand, the system would need to be general enough to support more than one application case, with different needs. As the end user is unlikely to be a simulation and analysis expert, specific workflows must be predefined for each application.

Data management in general is one of the main challenges to overcome. Ensembles of multi-parameter, time-varying simulations produce a

substantial quantity of outputs. Writing such data to disk, or transferring data for analysis or visualization can easily become a bottleneck of the urgent computing pipeline. Simulation codes are also interfaced with live sensor data that have to be managed accordingly.

Another point of importance is the interactivity of the approach. In this context of urgency, it is important that users should get feedback from the simulation and the data analysis progress, in order to start their interpretation process at the earliest. Progressive data analysis methods are thus of interest for such applications, where execution times and computing resources have to be optimized.

6.2 Topological Data Analysis in VESTEC

Our work in the project mainly dealt with the data analysis part of the VESTEC pipeline, showed in red in Figure 6.1. Specifically, in order to perform a trend analysis of the simulations results, we need to be able to compare different results, and perform statistical tasks such as clustering operations. Topological methods provide a great asset for this application case, as the main structures of the data are encoded in light and robust representations, which facilitates data management.

6.2.1 Persistence diagrams for Data Reduction

Topological Data Analysis was used in the project as a mean to perform data reduction on the large quantities of outputs generated by ensemble of numerical simulations. The simulations from the three use cases are all time-varying, and they supply several scalar fields at each time steps. As mentioned before, this produces numerous scalar fields to analyze and possibly store for visualization.

The persistence diagram was chosen in the project as the main topological representation to represent each scalar field. This choice was motivated by the fact that persistence diagrams are lightweight representations of the features of interest in a scalar field and are stable to additive noise, which makes them great candidate representations to perform data reduction. Additionally, stable and well-understood metrics exist for these representations, which is needed to develop statistical analysis methods. A persistence diagram was therefore computed on each output scalar field of the simulation step. This step effectively reduces the data size by a factor 100 for scalar fields from the wildfire and mosquito-borne disease

use cases, and a factor of almost 1000 for 3D scalar fields from the space weather use case. Discussions were needed with the experts from each use case to identify which scalar field was relevant for topological analysis, and how to preprocess the raw data for each use case.

6.2.2 In-Situ Computation

The computation of persistence diagrams is done using the Topology ToolKit (TTK) [TFL+17], on the same HPC environment where simulations are run (Figure 6.1). In order to limit data management, the computation takes place *in-situ*. This means that persistence diagrams are computed and stored on-the-fly, without having to store to disk the actual simulation data, which can be discarded afterwards.

TTK leverages the VTK/ParaView software ecosystem on which it is based to support in-situ computation with Catalyst [ABG+15]. Catalyst is an API designed for the developers of simulation codes, which enables them to make the internal data of the simulation accessible, without disk storage, in the form of VTK objects, in place, at run-time. Catalyst can be configured when the simulation is run to apply specific data analysis and visualization pipelines on each of the time steps generated by the simulation. Typically, Catalyst can be configured to execute a python script at user-defined time steps of the simulation, to run an analysis pipeline directly from the simulation code. Note that the simulation code must implement an interface with Catalyst, to make its internal generated data available in the form of VTK objects. The output of the analysis pipeline can be, itself, stored to disk if desired.

Thus, persistence diagrams are computed in-situ using Catalyst for different time steps of the simulation and stored in CINEMA databases, which are SQL-type databases of VTK files. This enables to express in a post-hoc post-processing SQL type queries for the inspection of specific diagrams (for a user defined selection of time steps, variables, etc.). Additionally, it may be desirable for some applications to store selected scalar fields in a non-reduced way, at key time steps of the simulation, to support further interactive manipulation in a post-process, or for visualization purposes. To cope with data size, a lossy topology-preserving compressor [SPCT18a] can be called through Catalyst before storing the scalar field to disk.

6.2.3 Statistical Analysis

Once all simulations have terminated, the persistence diagrams corresponding to each time step have been computed and stored in CINEMA databases. They are then used to perform statistical analysis tasks upon user request, such as comparing the evolution of different simulation runs based on the evolution of the topology of the scalar field. This analysis is run in a post-processing step, also on the HPC environment.

The tools used to perform such tasks are the computation of distances, barycenters and clusters of persistence diagrams. Wasserstein distances between diagrams are efficiently approximated using the Auction algorithm (Section 5.2.1). Barycenters and clusters of persistence diagrams are computed using our progressive approach (Chapter 5).

6.3 Results: The Space Weather Use Case

In this section, we demonstrate the data analysis pipeline for the space weather use case , which is the most advanced use case in the VESTEC project at the time of the writing of this chapter.

6.3.1 *In-Situ* Computation of Persistence Diagrams

The magnetic reconnection events in space weather are simulated using iPIC3D [MLRu10], a widely-used, massively parallel Particle-In-Cell (PIC) simulation code developed at KTH. In the context of VESTEC, this code was interfaced with Catalyst for the in-situ computation of persistence diagrams. To experiment with this use case, an ensemble of *four* simulation runs were launched on the same 3-dimensional domain: a box of dimensions $128 \times 64 \times 64$, and during 2500 time steps. Different sets of input parameters were used, with variations in the initial magnetic field and in the type of particles present in the domain. The persistence diagram of the magnitude of the magnetic field is computed for each run and each time step, resulting in an output ensemble of $10,000$ persistence diagrams, stored in a cinema database. They can then be accessed in a post-processing step through SQL type queries. 600 of these diagrams corresponding to the late time steps are represented in Figure 6.5.

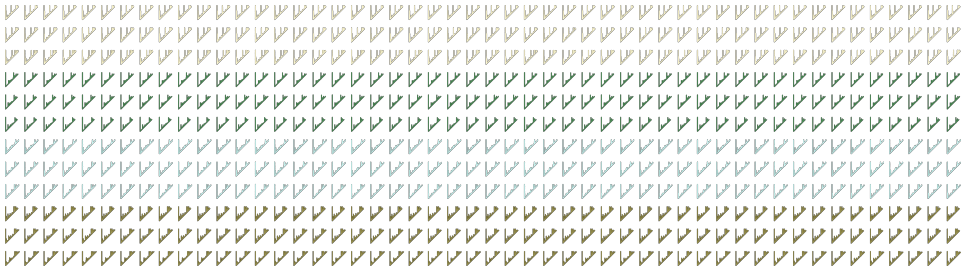


Figure 6.5 – *Persistence diagrams computed in-situ during numerical simulation. Diagrams from the last 300 time steps are represented, one every two time steps. Colors denote the corresponding simulation runs.*

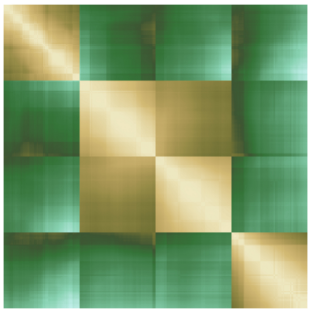


Figure 6.6 – *Wasserstein distance matrix between persistence diagrams corresponding to the late time steps of the four simulation runs (depicted in Figure 6.5).*

6.3.2 Wasserstein distances between diagrams

As mentioned before, statistical analysis is performed in a post-processing step (not *in-situ*) on the ensemble of persistence diagrams. To begin with, persistence diagrams can be compared with the computation of the Wasserstein distance (Section 2.4). The matrix of Wasserstein distances between diagrams (Figure 6.6) gives visual hints about the difference between runs. The matrix correctly captures the different simulation runs, with low values of the distance (yellow) between diagrams within runs. On the contrary, diagrams from different runs are more distant from one another (high values, in green to light blue). The visualization of the matrix also hints that the second and third simulations runs are similar, at least on the late time steps, which suggests the presence of three main trends in the data.

Additionally, the Wasserstein distance matrix can be used to embed the ensemble of persistence diagrams as a 3D point cloud. This gives a good representation of the similarity between different simulation runs. This step of dimension reduction is done using Multi-Dimensional Scaling (MDS), which is available in TTK through the scikit learn python package. Figure 6.7 gives an example of such a visualization for the four runs of

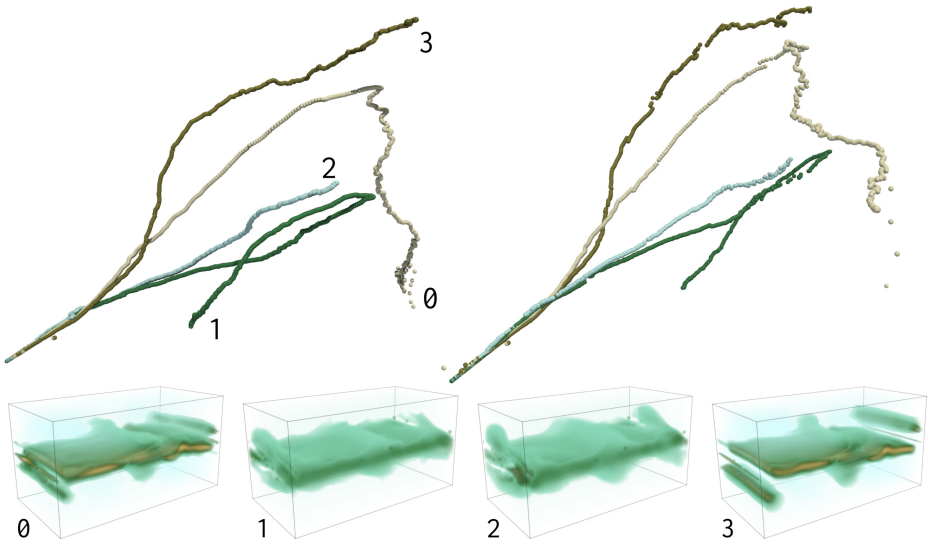


Figure 6.7 – *Visualization of the evolution of the simulation runs using MDS on the Wasserstein distance matrix between persistence diagrams. Each sphere corresponds to a persistence diagram, represented for the four simulation runs (colors), one every two time steps. Top left: visualization of the exact persistence diagrams. Top right: approximated diagrams with a* 5% *Bottleneck error. Bottom: example of a compressed scalar field in the late time steps for each one of the simulation runs.*

the space weather simulations. This representation shows hows the runs exhibit different behaviors past a certain point in time. Although the persistence diagrams are similar at the beginning of the simulation (bottom left of each curve), the different runs promptly start to diverge. On the right, the same visualization is showed using our approximated persistence diagrams (Chapter 4) with a Bottleneck error of 5% instead of the exact diagrams. The approximation of persistence diagrams enables us to get an exploitable approximated representation of the ensemble of simulation runs, while gaining time during their computation.

6.3.3 Topological Clustering

The persistence diagrams computed during the simulation step are also used to perform clustering operations. Our progressive clustering algorithm (Section 5.5) is used in VESTEC, as it supports time constraints and it provides centroids diagrams, representative of the repartition of topological features within each cluster. Figure 6.8 shows the result of our topological clustering algorithm applied to the space weather use case. In this illustration, the 3D representation of the ensemble of simulation runs is augmented with the result of the clustering, performed on the late diagrams. With $k = 4$ (top row), our method correctly retrieves the clas-

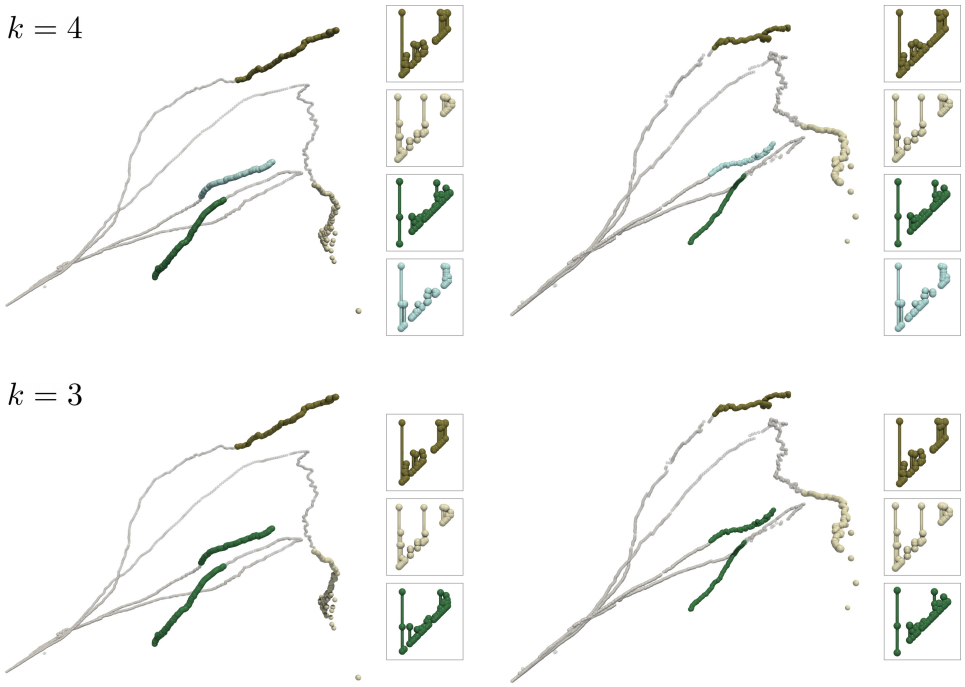


Figure 6.8 – *Topological clustering performed on the persistence diagrams on the late time steps with $k = 4$ (top row) and $k = 3$ (bottom row), together with the resulting centroids. On the left: exact persistence diagrams. On the right, the diagrams were approximated with a 5% maximal Bottleneck error.*

sification corresponding to the four different simulation runs. With $k = 3$ (bottom row), the second and third simulation runs are clustered together, which was hinted by the visualization of the Wasserstein distance matrix (Figure 6.6). In this case, the approximation of persistence diagrams with a 5% tolerance on the Bottleneck error performs as well as the exact diagrams. In particular, the centroids found in the two cases are very similar. They provide visual summaries of the topological features inside each cluster, and can be used to compute distances between clusters, or estimate the variance within each cluster to support the trend analysis process.

In this example, a simple hierarchical clustering run on the L_2 distance matrix between members of the ensemble also yields a correct classification. However it requires to write and store to disc the whole dataset for each timestep, which becomes impracticable for the large-scale data sets considered in VESTEC. In contrast, our approach only requires to store and transfer the persistence diagrans, which are over 1000 times lighter in this case than the input data.

6.4 Conclusion

The work developed in this thesis is rather adapted to the expectations of a system such as the one built in VESTEC. Topological Data Analysis (TDA) is a great asset to reduce large ensembles of data and only consider the prominent features, encoded in lightweight data structures. Moreover, our focus on the development of progressive and approximate methods for TDA satisfies the need for interactivity in a system dedicated to urgent computing.

Our algorithms were integrated in the VESTEC prototype, and show promising results on data from the related use cases. Future work is still needed to provide more advanced statistical indicators on ensembles of persistence diagrams, in order to perform trend analysis on the result of large ensembles of simulations. For instance, computing deviations and variances between clusters of persistence diagrams could help to quantify the uncertainties of different simulation runs, although the relevance of such indicators is likely to depend on the application use case.

In 2022, the VESTEC project is expected to enter its last phase, including the final execution of the pre-identified use cases in collaboration with domain experts. This final step of the project will help gather user feedback and evaluate the impact of the progressive computations on the overall data exploration [MSA+19]. A journal paper is currently being prepared in collaboration with all the partners of the project, in order to present in depths the context and the outcomes of VESTEC.

Conclusion 7

In this thesis we worked towards the development of progressive methods for the topological analysis of scalar data. The aim of the progressivity is to offer an interactive visual exploration of the topology of the data, and to control the time and computing resource consumption of the data analysis and visualization process. Our work was integrated in a toolchain developed in the context of the VESTEC project, with the purpose of supporting the decision-making process in urgent catastrophic situations, thanks to interactive high-performance computing and interactive data analysis and visualization.

7.1 Summary of contributions

The contributions of our work span different aspects of the data analysis process, from the computation of reduced topological representation to the analysis and visualization of these representations through statistical analysis tasks. All the techniques presented in this thesis were implemented as open-source software and integrated into the Topology ToolKit [TFL+17].

Progressive computation of topological representations

In a first effort towards the computation of progressive topological representation of scalar data, we introduced in Chapter 3 an approach based on a multiresolution hierarchy of the data. Our edge-nested hierarchies enable the fast identification of topologically invariant vertices, for which topological information can be deduced without additional computation at each level of the hierarchy. This enables the definition of efficient *coarse-to-fine* topological algorithms for the extraction of the critical points, and

the computation of the extremum-saddle persistence diagram of a scalar field. Experiments performed on diverse samples of data sets show that our progressive methods are on average even faster that non-progressive reference methods, with an average speed up of 1.8 for the critical points, and 1.6 for the persistence diagram. Yet, they produce a collection of intermediate outputs in addition to the exact result, which enables a continuous visual feedback along the computation. We showed that upon interruption, our approach provides meaningful partial results that are useful for the visual analysis of the topology of the data, in a fraction of the total computation cost. Although no theoretical guarantees are provided on the quality of the interrupted results, we found that they are in practice good approximations of the exact result. We demonstrated this point with quantitative studies on an extensive collection of data sets. In particular, we showed that the Wasserstein distance of the intermediate persistence diagram to the exact result is monotonically decreasing in the large majority of cases.

Approximation of persistence Diagrams with Guarantees

In order to address a limitation of our progressive computation of persistence diagrams, the lack of guarantees on the quality of the interrupted results, we developed an approximation scheme for persistence diagrams that provide strong guarantees on the result (presented in Chapter 4). Specifically, the Bottleneck distance between our estimation and the exact result is controlled. Our approach is based on the same edge-nested multiresolution hierarchy used in the progressive computation of the persistence diagram. We artificially increase the number of topologically invariant vertices with local simplifications of the scalar field, thus reducing the total computation time by 18% with a mild relative Bottleneck distance of 5%. This method is particularly relevant to approximate the persistence diagram on noisy data sets: we obtained a reduction of 48% of the computation time for our largest, noisiest data set. In addition to the persistence diagram estimation, we provide visual hints of the uncertainty of our approximation, and a scalar field that correspond to the approximated diagram. Our approximation method is finer than a naive baseline alternative with the same guarantees, being a lot more precise in terms of the L_2-Wasserstein metrics on the diagrams and the L_2 metrics on the approximated fields.

Progressive barycenters of persistence diagrams

On the analysis of reduced topological representations, we focused on the problem of computing an *average* persistence diagram in an ensemble of input persistence diagrams. We introduced in Chapter 5 a new progressive approach for the efficient computation of the Wasserstein barycenter of a set of persistence diagrams, a notoriously computationally intensive task. We revisited the Auction algorithm for the efficient estimation of the Wasserstein distance between diagrams, and an existing algorithm for the estimation of a Wasserstein barycenter of diagrams. We introduced progressivity in the approach at two levels: a progressive increase in the accuracy of our computation, and a progressive increase in the number of persistent features in the output diagram. This results in a method that is an order of magnitude faster on average that the combination of the fastest existing techniques while offering estimations of similar quality. It is also interruptible and time-constrained, providing meaningful estimations of the barycenters in interactive times. Finally, we revisited the *k-means* algorithm to introduce an application of our method to ensemble clustering. Our algorithm progressively computes, within user-defined execution time contraints, meaningful clusters of ensemble data along with their barycenter diagram.

7.2 Discussion

Limitations and discussions about our different contributions were already presented in the dedicated chapters. However we would like to highlight hereafter some additional interesting points about the design of progressive approaches in topological data analysis.

A key aspect to the progressivity is the production of intermediate, *interpretable* partial results. The quality of these intermediate results are of crucial importance. In other words, the quality of the progressive approach highly depends on the path that it takes to solve the computation. In order to evaluate this quality, it is best to provide theoretical guarantees on the intermediate results, but that is no easy task from our experience. The work detailed in Chapter 3 lacks theoretical guarantees on the interrupted results of the progressive computation of the critical points and persistence diagrams, which is the main limitation of our approach. In that case, we provided empirical results to demonstrate their utility for the visual analysis of the topology of data. In the same way, no theoret-

ical guarantee is provided on our interrupted Wasserstein barycenters of persistence diagrams (Chapter 5), and we also provide empirical evidence that they correctly converge towards a meaningful result. In contrast, we focused in the development of our approximation approach for the persistence diagram (Chapter 4) on delivering strong guarantees on the result. Unfortunately, although we reuse our progressive hierarchical data representation, our method is not progressive as the hierarchy has to be processed until the very last level to yield guarantees. Future work is needed to be able to efficiently resume the computation, in order to progressively decrease the error threshold for instance.

We already greatly detailed the advantages of progressive methods for the purposes of interactivity and computing resources optimization. However, an important lesson from our work is the idea that progressive approaches can sometimes be designed to be faster that their non-progressive counterpart. Indeed, for the progressive computation of critical points detailed in Chapter 3, the use of a progressive hierarchical representation of the data enables the identification of the topologically invariant vertices, which allows shortcuts in the computation. In the same manner, the progressive computation of the Wasserstein barycenter enables to compute matchings between diagrams that are drastically less costly, and gears the convergence of the barycenter. The common idea to these two examples is that having computed a first approximation of our result can enable the identification of shortcuts in the later computation. Thus, refining an approximation along progressive updates can sometimes be faster that computing directly the exact result. This point was surprising to us (and to some extent, to reviewers as well), and may not be obvious to grasp at first. In our opinion, this observation should drive us to try to consider progressive ways of solving issues in science, and especially in data analysis and visualization.

Finally, we would like to highlight the difficulty of progressively refining *global* topological structures. In Chapter 3, we managed to efficiently update the progressive critical points from a level of the hierarchy to another. Critical points are indeed characterized locally, which means that a localized change in the data occurring as the resolution increases only entails minor recomputation. In contrast, we were not able to efficiently maintain and update a data structure only a bit more complex such as integral lines, as mentioned in Section 3.4.2. As a result, our approach for the persistence diagram computation is not completely progressive. More

global structures, such as the merge tree or the Morse-Smale complex, would represent an even greater challenge.

7.3 Perspectives

Progressive approaches with guarantees

A natural perspective for future work involves the development of progressive approaches with theoretical guarantees on the quality of the intermediate results, and the convergence of the method. We started working towards this goal with the introduction of our approximation methods for persistence diagrams with guarantees on the Bottleneck error. We would like to find ways to build a full progressive computation method with the same guarantees, and to work up to error bounds in terms of different metrics, notably the Wasserstein distance to the exact result.

Progressive complete data analysis pipelines

Although we presented an example of an overall time-constrained data analysis pipeline (Section 5.8) combining our time-constrained diagram estimation (Section 3.4) with our time-constrained clustering computation (Section 5.5), we have not developed overall *progressive* analysis pipelines. In theory, our progressive barycenter computation could manage progressive inputs, where each input persistence diagram would be streamed, starting from the highest persistent pairs to the lowest. Unfortunately, our progressive approach for the persistence diagram computation is not guaranteed to deliver diagrams in this fashion. In the future, we would like to seek ways to combine progressive approaches to build complete progressive topological data analysis pipelines.

Since a progressive pipeline could be given a time-budget constraint and still be resumed afterwards if needed, we would also like to investigate how such preemptable data analysis methods could help for optimizing scheduling policies in high-performance environments, where data analysis is often run at the same time as data production and where the allocation of computation resources need to be finely optimized.

Generalization to other topological abstractions

In general, we firmly believe that the development of progressive and approximative approaches for data analysis and visualization is an important and exciting research direction. Indeed, these methods offer a

versatile solution to the challenge of processing data of ever-increasing size and complexity, in a context where the saving of time and power resources is confirmed to be an important stake for the future. They are especially relevant in the field of Topological Data Analysis, as most of the computational workload is typically spent in practice on the capture of small scale features (as illustrated throughout our work, for instance in Figure 4.1), whereas they usually are less relevant in the applications, and post-process simplification techniques are often applied to eliminate them anyway. In this context, a natural and thrilling avenue for future work would be the development of progressive and approximative methods to revisit existing topological data representations (merge trees, Reeb graphs, Morse-Smale complexes) in the light of progressive and degraded computation. In that perspective, the generalization of topologically invariant vertices to Discrete Morse Theory [For98] looks promising.

7.4 Final Word

This thesis was started in early 2018, at the beginning of the VESTEC project. At the time, I remember thinking that the project painted quite an apocalyptic picture of the future: fires destroying lands, and satellites falling from the skies, as mosquitoes roam over the Earth, transmitting deadly diseases. Three years later, as I write this manuscript, it almost looks like a bad joke. It has been almost two years since the beginning of the coronavirus disease pandemic, and it continues to dramatically impact the world. This last summer only, the effects of man-made climate change have showed in unprecedented ways. Enormous fires ravaged lands in Greece, Turkey, California, Canada, Siberia, and Central Africa. Deadly unprecedented floods took place in Europe and the United States.

Unfortunately, this motivates once more the relevance of urgent computing and the development of interactive techniques for the analysis and visualization of large ensembles of data. Once more, I believe that a key stake for the future would be to find approximate and progressive ways to solve these issues, in order to reduce the time and power consumption of such algorithms. While this was not the subject of this thesis, and although it is hard to quantify, the environmental impact of data sciences should in my opinion become an important topic and could drive research towards the development of low-impact algorithms and hardware.

Appendix: Data List

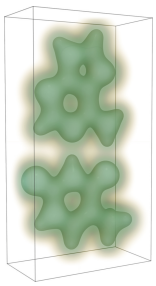
A

This appendix introduces an extensive list of the datasets used in this thesis, with information about the type of scalar data considered, its origin, and the interpretation of the salient features of the data. All the data has been made available through the Github repository related to each chapter:

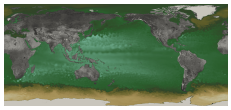
- https://github.com/julesvidal/progressive-topology
- https://github.com/julesvidal/persistence-diagram-approximation
- https://github.com/julesvidal/wasserstein-pd-barycenter

Single-field data (Chapters 3 and 4)

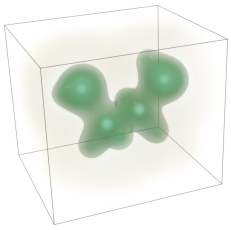
AT: Simulation of the electronic density on the adhenine-thymine molecular complex (Nucleobases A and T of DNA). The persistent maxima capture the atoms of the molecules.
Regular grid of dimensions $177 \times 95 \times 48$.



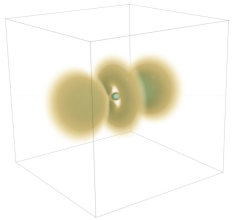
SeaSurfaceHeight: Height of the sea surface acquired through altimetry satellites. The features of interest are the centers of the eddies in the oceans, that are characterized by persistent height extrema.
2D regular grid of dimensions 1440×720.



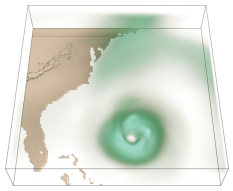
EthaneDiol: Simulation of the electronic density in an ethanediol molecule. The persistent maxima capture the atoms of the molecules.
Regular grid of dimensions $115 \times 116 \times 134$.



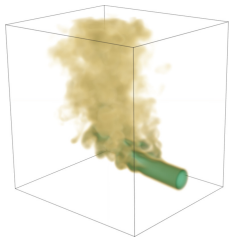
Hydrogen: Simulation of the spatial probability distribution of the electron in an hydrogen atom. The persistent maxima capture the localization of the main electronic orbitals.
Regular grid of dimensions $128 \times 128 \times 128$.
From [Kla20].



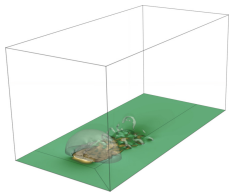
Isabel: Magnitude of the wind velocity in a simulation of the hurricane Isabel, that hit the east coast of the USA in 2003. High-persistence maxima capture the zones of high wind velocity, typically the eye-wall of the hurricane.
Regular grid of dimensions $250 \times 250 \times 50$.



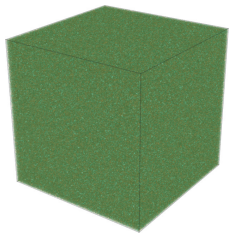
Combustion: Pressure simulation of a gas in a combustion chamber.
Regular grid of dimensions $170 \times 160 \times 140$.



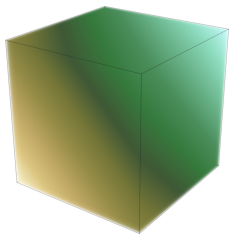
Boat: Simulation of the aerodynamics of a boat. Persistent extrema of the wind velocity capture the turbulent structures forming behind the boat.
Regular grid of dimensions $256 \times 256 \times 128$.



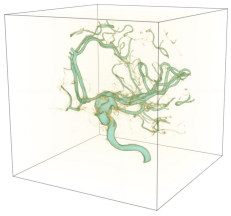
Random: Synthetic dataset. Random field on a cube.
Regular grid of dimensions: $250 \times 250 \times 250$.



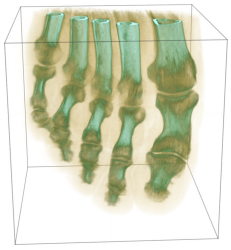
MinMax: Synthetic dataset. Elevation function on a cube.
Regular grid of dimensions: $255 \times 255 \times 255$.



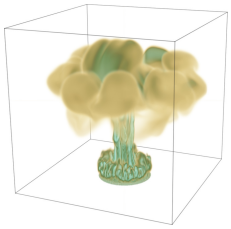
Aneurism: X-ray scan of the arteries of the right half of a human head. A contrast agent was injected into the blood and an aneurism is present.
Regular grid of dimensions $256 \times 256 \times 256$.
From [Kla20].



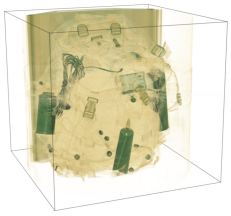
Foot: CT-scan of a human foot. The considered scalar field is the material density. Salient extrema correspond to bones and allow to capture the whole toes or different phalanges.
Regular grid of dimensions $256 \times 256 \times 256$.
From [TTK20].



Heptane: Pressure simulation of a jet of heptane gas undergoing combustion.
Regular grid of dimensions $302 \times 302 \times 302$.
From [Kla20].

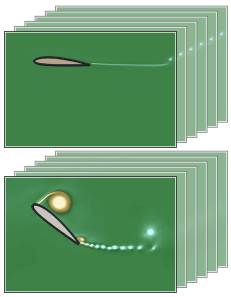


Backpack: Density in the CT-scan of a backpack filled with items. Persistent maximas capture the localization of items in the bag.
Regular grid of dimensions $512 \times 512 \times 373$.
From [Kla20].

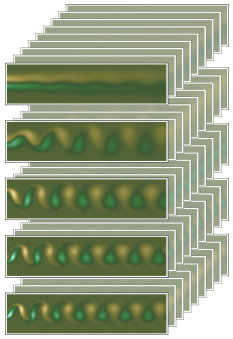


Ensemble data (Chapter 5)

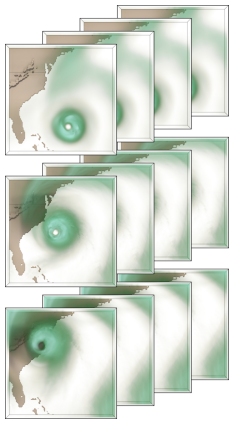
Starting Vortex: Simulated ensemble data set of the aerodynamics of a plane wing, for two distinct wing configurations. The data set contains 6 runs of a 2D simulation of a vortex behind the wing in each case. The considered scalar field is the orhogonal component of the air velocity curl. Vortices are captured by the extrema of the field.
Regular grid of dimensions 1500×1000



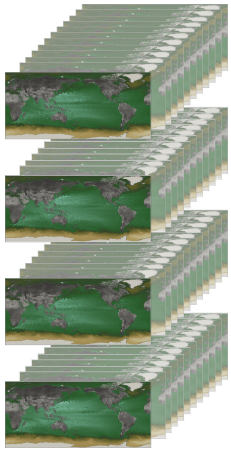
Vortex Street: Ensemble dataset containing 45 runs of a 2D simulation of flow turbulence behind an obstacle (Von Karman vortex street). The scalar field is the orthogonal component of the velocity curl, for 5 fluids of different viscosity, with 9 runs per fluid. Salient extrema of the field capture the center of vortices.
Regular grid of dimensions 300×100



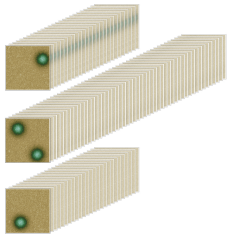
Isabel: Simulation of the Isabel hurricane. The ensemble contains 12 time steps of the simulation showing 3 different phases of the hurricane (formation, drift, and landfall), with 4 time steps per phase. The considered scalar field is the magnitude of the wind velocity. Salient maxima characterize the zones of high wind velocity, typically the eye-wall of the hurricane.
Regular grid of dimensions $250 \times 250 \times 50$.



SeaSurfaceHeight: Climate dataset showing the height of the sea surface, acquired by satellite at different moments of a year. 12 observations were made for each season, respectively in April, July, October and December, totalizing 48 members. Salient extrema of the surface height characterize the eddies.
Regular grid of dimensions 1440×720.



Gaussians: Synthetic ensemble data set containing 100 2D noisy members, classified in 3 clusters containing respectively 25, 50 and 25 members. Each cluster exhibits a different pattern of Gaussians, captured by the persistent maximas of the field.
Regular grid of dimensions 512×512



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. (Cited page 5.)

[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. (Cited pages 90 and 130.)

[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*, 2017. (Cited page 96.)

[ADB+16] Muhammet M. Akiner, Berna Demirci, Giorgi Babuadze, Vincent Robert, and Francis Schaffner. Correction: Spread of the invasive mosquitoes aedes aegypti and aedes albopictus in the black sea region increases risk of chikungunya, dengue, and zika outbreaks in europe. *PLOS Neglected Tropical Diseases*, 10(5):1–2, 05 2016. (Cited page 127.)

[AE13] Tushar Athawale and Alireza Entezari. Uncertainty quantification in linear interpolation for isosurface extraction. *IEEE Transactions on Visualization and Computer Graphics*, 2013. (Cited page 94.)

[AGL05] James Ahrens, Berk Geveci, and Charles Law. ParaView: An End-User Tool for Large-Data Visualization. *The Visualization Handbook*, pages 717–731, 2005. (Cited pages 3 and 65.)

[AJ19] T. Athawale and C. R. Johnson. Probabilistic asymptotic decider for topological ambiguity resolution in level-set ex-

traction for uncertain 2d data. *IEEE Transactions on Visualization and Computer Graphics*, 2019. (Cited page 94.)

[ASE16] T. Athawale, E. Sakhaee, and A. Entezari. Isosurface visualization of data with nonparametric models for uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 2016. (Cited page 94.)

[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. *Comp. Grap. For.*, 2016. (Cited pages 90 and 95.)

[Ban70] T. F. Banchoff. Critical points and curvature for embedded polyhedral surfaces. *The American Mathematical Monthly*, 1970. (Cited pages 21, 36, 47, 60, and 61.)

[BC91] Dimitri P. Bertsekas and David Castañon. Parallel synchronous and asynchronous implementations of the auction algorithm. *Parallel Computing*, 1991. (Cited page 99.)

[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 (Proc. of IEEE VIS)*, 2018. (Cited page 5.)

[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. (Cited page 36.)

[Bel66] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966. (Cited page 7.)

[Ber81] Dimitri P. Bertsekas. A new algorithm for the assignment problem. *Mathematical Programming*, 1981. (Cited pages 83, 91, 96, 97, 98, 99, 100, 102, 103, and 119.)

[Bey98] Jurgen Bey. Simplicial grid refinement: on Freudenthal's algorithm and the optimal number of congruence classes. *Numer. Math.*, 85:1–29, 1998. (Cited pages 36, 37, 38, 40, and 41.)

[BFS00] Silvia Biasotti, Bianca Falcidieno, and Michela Spagnuolo. Extended Reeb Graphs for Surface Understanding and Description. In *Discrete Geometry for Computer Imagery*, 2000. (Cited page 36.)

[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*, 2018. (Cited pages 5 and 22.)

[BGSF08] S. Biasotti, D. Giorgio, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *TCS*, 2008. (Cited pages 5 and 31.)

[BGW14] U. Bauer, X. Ge, and Y. Wang. Measuring distance between Reeb graphs. In *S. o. C. G.*, 2014. (Cited page 96.)

[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. (Cited page 94.)

[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*, 2012. (Cited page 94.)

[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. (Cited pages 3 and 80.)

[BMW15] Ulrich Bauer, Elizabeth Munch, and Yusu Wang. Strong equivalence of the interleaving and functional distortion metrics for Reeb graphs. In *S. o. C. G.*, 2015. (Cited page 96.)

[BR63] R. L. Boyell and H. Ruston. Hybrid techniques for real-time radar simulation. In *Proc. of the IEEE Fall Joint Computer Conference*, 1963. (Cited page 5.)

[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*, 2011. (Cited pages 5, 36, and 50.)

[BYM+14] Kenes Beketayev, Damir Yeliussizov, Dmitriy Morozov, Gunther H. Weber, and Bernd Hamann. Measuring the distance between merge trees. In *TopoInVis*. 2014. (Cited page 96.)

[CCG+09] Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas J. Guibas, and Steve Oudot. Proximity of persistence modules and their diagrams. In *S. o. C. G.*, 2009. (Cited page 27.)

[CCO17] Mathieu Carrière, Marco Cuturi, and Steve Oudot. Sliced Wasserstein Kernel for Persistence Diagrams. *ICML*, 2017. (Cited pages 5 and 96.)

[CD14] Marco Cuturi and Arnaud Doucet. Fast computation of wasserstein barycenters. In *ICML*, 2014. (Cited page 96.)

[CEH05] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. In *S. o. C. G.*, 2005. (Cited pages 27, 30, 76, and 83.)

[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. (Cited pages 107 and 117.)

[CLLR05] Yi-Jen Chiang, Tobias Lenz, Xiang Lu, and Günter Rote. Simple and optimal output-sensitive construction of contour trees using monotone paths. *Comput. Geom.*, 2005. (Cited page 52.)

[CLRS09] T. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2009. (Cited pages 47, 52, and 54.)

[CRM+20] Beniamino Caputo, Gianluca Russo, Mattia Manica, Francesco Vairo, Piero Poletti, Giorgio Guzzetta, Stefano Merler, Carolina Scagnolari, and Angelo Solimini. A comparative analysis of the 2007 and 2017 italian chikungunya outbreaks and implication for public health response. *PLOS Neglected Tropical Diseases*, 14(6):1–12, 06 2020. (Cited page 126.)

[CSA00] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. In *Symp. on Dis. Alg.*, 2000. (Cited pages 5, 31, and 36.)

[CSEH05] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. In *S. o. C. G.*, 2005. (Cited page 96.)

[CSEHM10] David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Yuriy Mileyko. Lipschitz functions have lp-stable persistence. *Foundations of Computational Mathematics*, 2010. (Cited page 27.)

[CSvdP04] Hamish A. Carr, Jack Snoeyink, and Michiel van de Panne. Simplifying Flexible Isosurfaces Using Local Geometric Measures. In *IEEE VIS*, 2004. (Cited page 5.)

[Cut13] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NIPS*. 2013. (Cited page 96.)

[CWSA16] Hamish A. Carr, Gunther H. Weber, Christopher M. Sewell, and James P. Ahrens. Parallel peak pruning for scalable SMP contour tree computation. In *IEEE Symposium on Large Data Analysis and Visualization*, 2016. (Cited page 52.)

[DBvK97] Mark De Berg and Marc van Kreveld. Trekking in the alps without freezing or getting tired. *Algorithmica*, 1997. (Cited page 5.)

[DFFIM15] Leila De Floriani, Ulderico Fugacci, Federico Iuricich, and Paola Magillo. Morse complexes for shape segmentation and homological analysis: discrete models and algorithms. *Comp. Grap. For.*, 2015. (Cited pages 5 and 32.)

[DHLZ02] Peter Diggle, Patrick Heagerty, K.-Y. Liang, and Scott Zeger. *The Analysis of Longitudinal Data*. Oxford University Press, 2002. (Cited page 95.)

[DM98] Leonardo Dagum and Ramesh Menon. OpenMP: an industry standard API for shared-memory programming. *IEEE computational science and engineering*, 5(1):46–55, 1998. (Cited page 61.)

[EH09] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2009. (Cited pages 5, 12, 17, 18, 22, 23, 25, 26, 31, 50, 53, and 96.)

[EHZ01] Herbert Edelsbrunner, John Harer, and Afra Zomorodian. Hierarchical morse complexes for piecewise linear 2-manifolds. In *S. o. C. G.*, 2001. (Cited page 36.)

[Elk03] C. Elkan. Using the triangle inequality to accelerate k-means. In *ICML*, 2003. (Cited page 107.)

[ELZ02] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Disc. Compu. Geom.*, 2002. (Cited pages 5 and 7.)

[EM90] Herbert Edelsbrunner and Ernst P Mucke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. on Graph.*, 1990. (Cited page 19.)

[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*, 2016. (Cited page 95.)

[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*, 2018. (Cited pages 95, 108, and 116.)

[FKRW16] F. Ferstl, M. Kanzler, M. Rautenhaus, and R. Westermann. Visual analysis of spatial variability and global correlations in ensembles of iso-contours. *Comp. Grap. For.*, 2016. (Cited page 95.)

[For98] Robin Forman. A User's Guide to Discrete Morse Theory. *Advances in Mathematics*, 1998. (Cited page 142.)

[FP16] Jean-Daniel Fekete and Romain Primet. Progressive Analytics: A Computation Paradigm for Exploratory Data Analysis. *arXiv*, 2016. (Cited page 6.)

[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 (Proc. of IEEE VIS)*, 2014. (Cited pages 5 and 22.)

[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. *Comp. Graph. For.*, 2014. (Cited page 5.)

[GBHP08] A. Gyulassy, P. T. Bremer, B. Hamann, and V. Pascucci. A practical approach to Morse-Smale complex computation: Scalability and generality. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2008. (Cited page 5.)

[GBP18] 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 (Proc. of IEEE VIS)*, 2018. (Cited page 5.)

[GDL+02] Benjamin F. Gregorski, Mark A. Duchaineau, Peter Lindstrom, Valerio Pascucci, and Kenneth I. Joy. Interactive view-dependent rendering of large isosurfaces. In *Proc. of IEEE VIS*, pages 475–482, 2002. (Cited page 36.)

[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 (Proc. of IEEE VIS)*, 2007. (Cited page 5.)

[GFJT17] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Task-based Augmented Merge Trees with Fibonacci Heaps,. In *IEEE LDAV*, 2017. (Cited page 108.)

[GFJT19] Charles Gueunet, Pierre Fortin, Julien Jomier, and Julien Tierny. Task-Based Augmented Contour Trees with Fi-

bonacci Heaps. *IEEE Trans. Parallel Distrib. Syst.*, 2019. (Cited pages 5, 31, 60, 61, 81, and 82.)

[GKL+15] 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 (Proc. of IEEE VIS)*, 2015. (Cited page 5.)

[GP00] Thomas Gerstner and Renato Pajarola. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. In *Proc. of IEEE VIS*, pages 259–266, 2000. (Cited page 36.)

[GRP+12] David Guenther, Jan Reininghaus, Steffen Prohaska, Tino Weinkauf, and Hans-Christian Hege. Efficient computation of a hierarchy of discrete 3d gradient vector fields. In *Proc. of TopoInVis*, pages 15–29, 2012. (Cited page 36.)

[GST14] David Günther, Joseph Salmon, and Julien Tierny. Mandatory critical points of 2D uncertain scalar fields. *Comp. Graph. For.*, 2014. (Cited page 94.)

[H. 42] H. Freudenthal. Simplizialzerlegungen von beschrankter Flachheit. *Annals of Mathematics*, 43:580–582, 1942. (Cited pages 36, 37, 38, 39, and 65.)

[HC20] M. Hesse and P. A. Cassak. Magnetic reconnection in the space sciences: Past, present, and future. *Journal of Geophysical Research: Space Physics*, 125(2):e2018JA025935, 2020. e2018JA025935 2018JA025935. (Cited page 128.)

[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. *Comp. Grap. For.*, 2016. (Cited page 5.)

[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*, 2013. (Cited page 95.)

[HSKK01a] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *Proc. of ACM SIGGRAPH*, 2001. (Cited page 96.)

[HSKK01b] Masaki Hilaga, Yoshihisa Shinagawa, Taku Komura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *Proc. of ACM SIGGRAPH*, 2001. (Cited page 36.)

[IF17] Federico Iuricich and Leila De Floriani. Hierarchical forman triangulation: A multiscale model for scalar field analysis. *Comput. Graph.*, 66:113–123, 2017. (Cited page 36.)

[J. 95] J. Bey. Tetrahedral grid refinement. *Computing*, 55:355–378, 1995. (Cited pages 36, 37, 38, 39, 41, and 65.)

[JS03] C. R. Johnson and A. R. Sanderson. A next step: Visualizing errors and uncertainty. *IEEE Computer Graphics and Applications*, 2003. (Cited page 94.)

[JS06] Guangfeng Ji and Han-Wei Shen. Feature Tracking using Earth Mover's Distance and Global Optimization. In *Proc. of IEEE PacificVis*, 2006. (Cited page 50.)

[JSF20] Jaemin Jo, Jinwook Seo, and Jean-Daniel Fekete. PANENE: A Progressive Algorithm for Indexing and Querying Approximate k-Nearest Neighbors. *IEEE Transactions on Visualization and Computer Graphics*, 2020. (Cited page 6.)

[Kan42] Leonid Kantorovich. On the translocation of masses. *AS URSS*, 1942. (Cited pages 28 and 96.)

[KE07] Thomas Klein and Thomas Ertl. Scale-Space Tracking of Critical Points in 3D Vector Fields. In *Topology-based Methods in Visualization*, Mathematics and Visualization. Springer, 2007. (Cited page 50.)

[Kep19] Johannes Kepler. 1619. (Cited page 2.)

[Kla20] Pavol Klacansky. Open Scientific Visualization Data Sets. `https://klacansky.com/open-scivis-datasets/`, 2020. (Cited pages 55, 56, 80, 144, and 145.)

[KMN16] Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams. *ACM Journal of Experimental Algorithmics*, 2016. (Cited pages 83, 91, 96, 97, 98, 99, 100, 102, 108, 110, 111, 112, 113, and 119.)

[Kra10] Martin Kraus. Visualization of uncertain contour trees. In *International Conference on Information Visualization Theory and Applications*, 2010. (Cited page 95.)

[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*, 2011. (Cited pages 5 and 22.)

[Kuh60] H W Kuhn. Some combinatorial lemmas in topology. *IBM Journal of Research and Development*, 45:518–524, 1960. (Cited pages 16, 36, 37, 38, 40, and 41.)

[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 (Proc. of IEEE VIS)*, 2006. (Cited page 5.)

[LCO18] Théo Lacombe, Marco Cuturi, and Steve Oudot. Large Scale computation of Means and Clusters for Persistence Diagrams using Optimal Transport. In *NIPS*, 2018. (Cited pages 96, 108, and 110.)

[Llo82] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 1982. (Cited page 100.)

[Loo87] Charles Loop. Smooth Subdivision Surfaces Based on Triangles. Master's thesis, University of Utah, 1987. (Cited pages 36, 37, 38, 39, and 65.)

[LS16] T. Liebmann and G. Scheuermann. Critical points of gaussian-distributed scalar fields on simplicial grids. *Computer Graphics Forum (Proc. of EuroVis)*, 2016. (Cited page 94.)

[LVLM04] Thomas Lewiner, Luiz Velho, Hélio Lopes, and Vinícius Mello. Hierarchical isocontours extraction and compression. In *SIBGRAPHI*, pages 234–241, 2004. (Cited page 36.)

[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. *Cartography and Geographic Information Science*, 2005. (Cited page 94.)

[MDN12] Senthilnathan Maadasamy, Harish Doraiswamy, and Vijay Natarajan. A hybrid parallel algorithm for computing and tracking level set topology. In *Proc. of HiPC*, 2012. (Cited page 52.)

[Mil63] J. Milnor. *Morse Theory*. Princeton University Press, 1963. (Cited pages 21 and 26.)

[Mil68] Robert B. Miller. Response time in man-computer conversational transactions. In *Fall Joint Computer Conference*, 1968. (Cited page 5.)

[MLRu10] Stefano Markidis, Giovanni Lapenta, and Rizwan-uddin. Multi-scale simulations of plasma with ipic3d. *Mathematics and Computers in Simulation*, 80(7):1509–1519, 2010. Multiscale modeling of moving interfaces in materials. (Cited page 131.)

[Mon81] Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Académie Royale des Sciences de Paris*, 1781. (Cited pages 28 and 96.)

[Mor10] Dmitriy Morozov. Dionysus. `http://www.mrzv.org/software/dionysus`, 2010. Accessed: 2019-03-01. (Cited pages 28 and 102.)

[MOT10] H.-C. Hege M. Otto, T. Germer and H. Theisel. Uncertain 2D vector field topology. *Comp. Graph. For.*, 29:347–356, 2010. (Cited page 94.)

[MOT11] T. Germer M. Otto and H. Theisel. Uncertain topology of 3D vector fields. *Proc. of IEEE Pacific Vis*, 2011. (Cited page 94.)

[MSA+19] Luana Micallef, Hans-Jörg Schulz, Marco Angelini, Michaël Aupetit, Remco Chang, Jörn Kohlhammer, Adam Perer, and Giuseppe Santucci. The Human User in Progressive Visual Analytics. In *EuroVis 2019 Short Papers*, 2019. (Cited page 135.)

[Mun57] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 1957. (Cited pages 28, 91, 96, 97, and 102.)

[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*, 2014. (Cited page 95.)

[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*, 2019. (Cited pages 5 and 22.)

[PB00] Valerio Pascucci and Chandrajit L. Bajaj. Time critical isosurface refinement and smoothing. In *Proc. of the Volume Visualization and Graphics Symposium*, pages 33–42, 2000. (Cited page 36.)

[PCMS04] Valerio Pascucci, Kree Cole-McLaughlin, and Giorgio Scorzelli. Multi-resolution computation and presentation of contour trees. In *Proc. IASTED conference on visualization, imaging, and image processing*, 2004. (Cited page 36.)

[PGA13] K. Potter, S. Gerber, and E. W. Anderson. Visualization of uncertainty without a mean. *IEEE Computer Graphics and Applications*, 2013. (Cited page 94.)

[PH11] K. Pöthkow and H.-C. Hege. Positional uncertainty of isocontours: Condition analysis and probabilistic measures. *IEEE Transactions on Visualization and Computer Graphics*, 2011. (Cited page 94.)

[PH13] Kai Pöthkow and Hans-Christian Hege. Nonparametric models for uncertainty visualization. *Comp. Grap. For.*, 2013. (Cited page 94.)

[PK12] Johnson CR Potter K, Rosen P. From quantification to visualization: A taxonomy of uncertainty visualization approaches. *IFIP Advances in Information and Communication Technology*, 2012. (Cited page 94.)

[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*, 2013. (Cited page 94.)

[PPH12] Christoph Petz, Kai Pöthkow, and Hans-Christian Hege. Probabilistic local features in uncertain vector fields with spatial correlation. *Comp. Graph. For.*, 2012. (Cited page 94.)

[PPH13] Kai Pöthkow, Christoph Petz, and Hans-Christian Hege. Approximate level-crossing probabilities for interactive visualization of uncertain isocontours. *Int. J. Uncert. Quantif.*, 2013. (Cited page 94.)

[PRW11] Tobias Pfaffelmoser, Matthias Reitinger, and Rüdiger Westermann. Visualizing the positional and geometrical variability of isosurfaces in uncertain scalar fields. *Comp. Graph. For.*, 2011. (Cited page 94.)

[PSBM07] V Pascucci, G Scorzelli, P T Bremer, and A Mascarenhas. Robust on-line computation of Reeb graphs: simplicity and speed. *ACM Trans. on Graph.*, 2007. (Cited pages 5, 36, and 37.)

[PW12] Tobias Pfaffelmoser and Rüdiger Westermann. Visualization of global correlation structures in uncertain 2D scalar fields. *Comp. Grap. For.*, 2012. (Cited page 94.)

[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 International Conference on Data Mining Workshops*, 2009. (Cited page 95.)

[PWH11] Kai Pöthkow, Britta Weber, and Hans-Christian Hege. Probabilistic marching cubes. In *Comp. Graph. For.*, 2011. (Cited page 94.)

[PWL97] A. T. Pang, C. M. Wittenbrink, and S. K. Lodha. Approaches to uncertainty visualization. *The Visual Computer*, 1997. (Cited page 94.)

[RCL+21] Martin Royer, Frédéric Chazal, Clément Levrard, Yuhei Umeda, and Yuichi Ike. ATOL: Measure Vectorization for

Automatic Topologically-Oriented Learning. In *The 24th International Conference on Artificial Intelligence and Statistics (AISTATS 2021)*, The 24th International Conference on Artificial Intelligence and Statistics, Virtual conference, France, April 2021. (Cited page 117.)

[R.E83] R.E. Bank, and A.H. Sherman, and A. Weiser. Refinement algorithms and data structures for regular local mesh refinement. *Scientific Computing*, pages 3–17, 1983. (Cited pages 36, 37, 38, 39, 41, and 65.)

[Ree46] Georges Reeb. Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique. *Comptes Rendus des séances de l'Académie des sciences*, 222(847-849):76, 1946. (Cited pages 5 and 31.)

[RHBK15] Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A stable multi-scale kernel for topological machine learning. In *IEEE CVPR*, 2015. (Cited pages 5 and 96.)

[RKWH12] Jan Reininghaus, Jens Kasten, Tino Weinkauf, and Ingrid Hotz. Efficient Computation of Combinatorial Feature Flow Fields. *IEEE Transactions on Visualization and Computer Graphics*, 2012. (Cited page 50.)

[RSL17] B. Rieck, F. Sadlo, and H. Leitte. Topological machine learning with persistence indicator functions. In *Proc. of TopoInVis*, 2017. (Cited pages 5 and 96.)

[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.*, 2011. (Cited page 5.)

[S. 95] S. Zhang. Successive subdivision of tetrahedra and multigrid methods on tetrahedral meshes. *Houston Journal of Mathematics*, 21:541–556, 1995. (Cited pages 36, 37, 38, 39, 41, and 65.)

[SBB18] Francesca Samsel, Lyn Bartram, and Annie Bares. Art, affect and color: Creating engaging expressive scientific visualization. In *2018 IEEE VIS Arts Program (VISAP)*, pages 1–9, 2018. (Cited pages 19 and 20.)

[Sci04] I.E.E.E SciVisContest. Simulation of the isabel hurricane. http://sciviscontest-staging.ieeevis.org/2004/data.html, 2004. (Cited pages 109 and 118.)

[SDGP+15] Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. Convolutional wasserstein distances. *ACM Transactions on Graphics*, 2015. (Cited page 96.)

[SK08] Johannes Singler and Benjamin Konsik. The GNU libstdc++ Parallel Mode: Software Engineering Considerations. In *Proc. of International Workshop on Multicore Software Engineering*, 2008. (Cited pages 54 and 79.)

[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 (Proc. of IEEE VIS)*, 2012. (Cited page 94.)

[SM17] Dmitriy Smirnov and Dmitriy Morozov. Triplet Merge Trees. In *TopoInVis*, 2017. (Cited pages 5 and 52.)

[Sou11] T. Sousbie. The persistent cosmic web and its filamentary structure: Theory and implementations. *Royal Astronomical Society*, 2011. http://www2.iap.fr/users/sousbie/web/html/indexd41d.html. (Cited pages 5 and 23.)

[SPCT18a] M. Soler, M. Plainchault, B. Conche, and J. Tierny. Topologically controlled lossy compression. In *2018 IEEE Pacific Visualization Symposium (PacificVis)*, 2018. (Cited page 130.)

[SPCT18b] Maxime Soler, Mélanie Plainchault, Bruno Conche, and Juilen Tierny. Lifted Wasserstein Matcher for Fast and Robust Topology Tracking. In *IEEE Symposium on Large Data Analysis and Visualization*, 2018. (Cited pages 29, 95, 96, 97, 102, 108, and 110.)

[SPCT18c] Maxime Soler, Melanie Plainchault, Bruno Conche, and Julien Tierny. Lifted Wasserstein matcher for fast and robust topology tracking. In *IEEE Symposium on Large Data Analysis and Visualization*, 2018. (Cited page 50.)

[SPD+19] Maxime Soler, Martin Petitfrere, Gilles Darche, Melanie Plainchault, Bruno Conche, and Julien Tierny. Ranking Viscous Finger Simulations to an Acquired Ground Truth with Topology-Aware Matchings. In *IEEE Symposium on Large Data Analysis and Visualization*, 2019. (Cited page 5.)

[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*, 2016. `http://vgl.serc.iisc.ernet.in/felix/index.html`. (Cited pages 5 and 23.)

[SSW14] Himangshu Saikia, Hans-Peter Seidel, and Tino Weinkauf. Extended branch decomposition graphs: Structural comparison of scalar data. *Comp. Graph. For.*, 2014. (Cited page 96.)

[ST83] D. Sleator and R. Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 1983. (Cited pages 47 and 48.)

[SW17] Himangshu Saikia and Tino Weinkauf. Global Feature Tracking and Similarity Estimation in Time-Dependent Scalar Fields. *Comp. Graph. For.*, 2017. (Cited page 50.)

[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*, 2010. (Cited page 95.)

[Szy13] A. Szymczak. Hierarchy of stable Morse decompositions. *IEEE Transactions on Visualization and Computer Graphics*, 2013. (Cited page 94.)

[TFL+17] Julien Tierny, Guillaume Favelier, Joshua A. Levine, Charles Gueunet, and Michael Michaux. The Topology ToolKit. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2017. `https://topology-tool-kit.github.io/`. (Cited pages 3, 31, 34, 55, 60, 61, 63, 65, 70, 80, 81, 92, 108, 130, and 137.)

[Tie18] Julien Tierny. *Topological Data Analysis for Scientific Visualization*. Springer, 2018. (Cited page 12.)

[TMMH14] Katharine Turner, Yuriy Mileyko, Sayan Mukherjee, and John Harer. Fréchet Means for Distributions of Persistence Diagrams. *Disc. Compu. Geom.*, 2014. (Cited pages 9, 91, 97, 100, 101, 102, 103, 108, 110, 111, 112, 113, 117, and 119.)

[TN13] Dilip Mathew Thomas and Vijay Natarajan. Detecting symmetry in scalar fields using augmented extremum graphs. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2013. (Cited page 96.)

[TN14] D. M. Thomas and V. Natarajan. Multiscale symmetry detection in scalar fields by clustering contours. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2014. (Cited page 96.)

[TTK20] TTK Contributors. TTK Data. `https://github.com/topology-tool-kit/ttk-data/tree/dev`, 2020. (Cited pages 3, 23, 27, 31, 55, 80, 118, and 145.)

[TV98] S. Tarasov and M. Vyali. Construction of contour trees in 3d in o(n log n) steps. In *S. o. C. G.*, 1998. (Cited page 5.)

[unc08] ISO/IEC Guide 98-3:2008 uncertainty of measurement - part 3: Guide to the expression of uncertainty in measurement (GUM). 2008. (Cited page 94.)

[VBT19] J. Vidal, J. Budin, and J. Tierny. Progressive wasserstein barycenters of persistence diagrams. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2019. (Cited page 91.)

[VGT21] J. Vidal, P. Guillou, and J. Tierny. A progressive approach to scalar field topology. *IEEE Transactions on Visualization and Computer Graphics*, 2021. (Cited page 34.)

[VT21] J. Vidal and J. Tierny. Fast approximation of persistence diagrams with guarantees. *IEEE Symposium on Large Data Analysis and Visualization*, 2021. (Cited page 70.)

[WAG+19] A. Park Williams, John T. Abatzoglou, Alexander Gershunov, Janin Guzman-Morales, Daniel A. Bishop, Jennifer K. Balch, and Dennis P. Lettenmaier. Observed impacts of anthropogenic climate change on wildfire in california. *Earth's Future*, 7(8):892–910, 2019. (Cited page 125.)

[WF09a] Kenneth Weiss and Leila De Floriani. Diamond hierarchies of arbitrary dimension. *Comput. Graph. Forum*, 28(5):1289–1300, 2009. (Cited page 36.)

[WF09b] Kenneth Weiss and Leila De Floriani. Supercubes: A high-level primitive for diamond hierarchies. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 15(6):1603–1610, 2009. (Cited page 36.)

[WM04] Matt Williams and Tamara Munzner. Steerable, Progressive Multidimensional Scaling. In *Proc. of IEEE InfoVis*, 2004. (Cited page 6.)

[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*, 2013. (Cited page 95.)

[WZ13] Keqin Wu and Song Zhang. A contour tree based visualization for exploring data with uncertainty. *International Journal for Uncertainty Quantification*, 2013. (Cited page 95.)

[ZGC+17] Emanuel Zgraggen, Alex Galakatos, Andrew Crotty, Jean-Daniel Fekete, and Tim Kraska. How progressive visualizations affect exploratory analysis. *IEEE Transactions on Visualization and Computer Graphics*, 2017. (Cited page 6.)

[ZSC+17] Qian Zhang, Kaiyuan Sun, Matteo Chinazzi, Ana Pastore y Piontti, Natalie E. Dean, Diana Patricia Rojas, Stefano Merler, Dina Mistry, Piero Poletti, Luca Rossi, Margaret Bray, M. Elizabeth Halloran, Ira M. Longini, and Alessandro Vespignani. Spread of zika virus in the americas. *Proceedings of the National Academy of Sciences*, 114(22):E4334–E4343, 2017. (Cited page 126.)

Progressivité en Analyse Topologique de Données

L'analyse topologique de donnés forme une famille d'outils qui permettent l'extraction générique et efficace de caractéristiques structurelles dans les données. Cependant, bien que ces techniques aient des complexités asymptotiques connues et raisonnables, elles sont rarement interactives en pratique sur des jeux de données réels, ce qui limite leur utilisation pour l'analyse et la visualisation interactives de données. Dans cette thèse, nous avons cherché à développer des méthodes *progressives* pour l'analyse topologique de données scalaires scientifiques, qui peuvent être interrompues pour fournir rapidement un résultat approché exploitable, et sont capables de l'affiner ensuite. Dans un premier temps, nous présentons une représentation hiérarchique des données d'entrée, qui permet de définir des algorithmes topologiques *coarse-to-fine* efficaces. En conséquence, nous introduisons deux algorithmes progressifs pour le calcul des *points critiques* et du *diagramme de persistance* d'un champ scalaire. Ces méthodes fournissent des sorties interprétables en cas d'interruption, offrent un retour visuel continu tout au long du calcul et sont plus rapides en pratique que leurs homologues non progressifs. Ensuite, nous revisitons ce cadre progressif pour introduire un algorithme pour le calcul approché du diagramme de persistance d'un champ scalaire, avec des garanties sur l'erreur d'approximation associée. Enfin, afin d'effectuer une analyse visuelle de *données d'ensemble*, nous présentons un nouvel algorithme progressif pour le calcul du barycentre de Wasserstein d'un ensemble de diagrammes de persistance, une tâche notoirement coûteuse en calcul. Notre approche progressive permet d'approcher le barycentre de manière interactive. Nous étendons cette méthode à un algorithme de classification topologique de données d'ensemble, qui est progressif et capable de respecter une contrainte de temps. Nous présentons un cas d'application de ces travaux, dans le contexte de l'analyse et la visualisation interactives de données pour l'aide à la prise de décision urgente en cas de situations de crises, dans le cadre du projet européen VESTEC.

Progressivity in Topological Data Analysis

Topological Data Analysis (TDA) forms a collection of tools that enable the generic and efficient extraction of features in data. However, although most TDA algorithms have practicable asymptotic complexities, these methods are rarely interactive on real-life datasets, which limits their usability for interactive data analysis and visualization. In this thesis, we aimed at developing *progressive* methods for the TDA of scientific scalar data, that can be interrupted to swiftly provide a meaningful approximate output and that are able to refine it otherwise. First, we present a hierarchical representation of the data that enables the definition of efficient coarse-to-fine topological algorithms. As a result we introduce two progressive algorithms for the computation of the *critical points* and the extremum-saddle *persistence diagram* of a scalar field. These methods provide interpretable outputs upon interruption, offer a continuous visual feedback along the computation, and are faster in practice than their non-progressive counterpart. Next, we revisit this progressive framework to introduce an approximation algorithm for the persistence diagram of a scalar field, with strong guarantees on the related approximation error. Finally, in a effort to perform visual analysis of *ensemble data*, we present a novel progressive algorithm for the computation of the discrete Wasserstein barycenter of a set of persistence diagrams, a notoriously computationally intensive task. Our progressive approach enables the approximation of the barycenter within interactive times. We extend this method to a progressive, time-constraint, topological ensemble clustering algorithm. We present an application use-case of all this work in the context of supercomputing and interactive data analysis and visualization to help the urgent decision-making process during crisis events, in the scope of the European project VESTEC.