DOCTORAL THESIS
SORBONNE UNIVERSITÉ

Speciality: **COMPUTER SCIENCE**

Presented by

MOHAMED KISSI

to obtain the degree of

Ph.D. of Sorbonne Université

Persistence Optimization for Data Visualization

Defended on October 28th, 2025, before the committee:

Georges Pierre BONNEAU	Professor	Grenoble Universities	**Reviewer**
Guillaume LAVOUÉ	Professor	Ecole Centrale de Lyon	**Reviewer**
Isabelle BLOCH	Professor	Sorbonne University	**Examiner**
Mathieu CARRIÈRE	Research Scientist	INRIA	**Examiner**
Joshua A. LEVINE	Associate Professor	University of Arizona	**Guest**
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

The order of names within each group is presented randomly.

I sincerely thank Georges Pierre BONNEAU and Guillaume LAVOUÉ for dedicating their time to review this thesis and offering invaluable feedback. My gratitude also extends to the entire jury for their active participation and commitment throughout the evaluation process.

Julien, je tiens à te remercier sincèrement pour l'encadrement exceptionnel dont tu m'as fait bénéficier tout au long de cette thèse. Dès le premier jour, tu as su m'accompagner, me guider et me prodiguer des conseils précieux. Ta disponibilité, ton écoute et ta bienveillance ont été inestimables. Merci pour ta gentillesse et ton soutien constant. Je n'aurais pas pu espérer un encadrement de thèse aussi enrichissant, à la fois sur le plan scientifique et humain.

Josh, I warmly thank you for your guidance and support throughout my thesis.

Je tiens à exprimer ma profonde reconnaissance envers Amaury, Emeric, Mamy, Loic, Marco, Francesco, Milla, Mattéo, Eve, Keanu, Sébastien, Sylvain, Thomas et Pierre pour leur précieux soutien, leur collaboration et leur camaraderie tout au long de ma thèse. Un remerciement tout particulier à Mathieu Pont, dont l'aide inestimable a été déterminante lors des premières étapes de ce voyage scientifique.

Je souhaite également remercier Thomas, Aminata et Joo-won, que j'ai eu la chance de connaître durant mon école d'ingénieur.

Réussir, c'est rendre hommage aux sacrifices et à l'amour inconditionnel de ses parents. J'espère que cette thèse pourra vous honorer et refléter l'impact de votre soutien dans mon parcours.

Je tiens à exprimer ma gratitude infinie à mes parents, qui ont été ma force motrice tout au long de ma scolarité. Leur amour et leurs encouragements constants m'ont poussé à me dépasser et à croire en mes rêves, même lorsque cela semblait difficile.

À mes sœurs, je dis un grand merci pour leur soutien indéfectible au fil des années. Vous m'avez accompagné avec patience et m'avez ouvert

les portes du monde en me faisant découvrir les voyages dès mon plus jeune âge. Ces expériences ont profondément marqué ma personnalité et m'ont aidé à grandir. Je remercie mon beau-frère.

Une pensée spéciale va à mon oncle Abdelharim, dont la présence et les encouragements tout au long de mon parcours scolaire ont été une source inestimable de motivation. Merci également à Malika et Hamida, qui ont été des piliers dans ma vie depuis mon enfance, ne manquant jamais une seule de mes représentations ou événements marquants.

Je souhaite enfin honorer la mémoire de mes grands-parents, qui auraient été si heureux de me voir obtenir ce titre. Je me remémore avec émotion les paroles de ma grand-mère, qui me répétait souvent : « Tu seras docteur. » Même si elle pensait à la médecine, je suis convaincu qu'elle serait fière aujourd'hui de me voir obtenir ce doctorat.

First-Author Publications

Journal Papers

A Practical Solver for Scalar Data Topological Simplification
Mohamed Kissi, Mathieu Pont, Joshua A. Levine, Julien Tierny
IEEE Transactions on Visualization and Computer Graphics, 2024
Presented at IEEE VIS 2024

Workshop Papers

Topology Aware Neural Interpolation of Scalar Fields
Mohamed Kissi, Keanu Sisouk, Joshua A. Levine, Julien Tierny
Under review for the IEEE Workshop on Topological Data Analysis and Visualization (TopoInVis), 2025

Contents

1 Introduction

Contents

As computational resources and acquisition devices are becoming more efficient and precise, modern datasets are growing in resolution and level of details. This results in a general growth in the size of datasets which creates challenges for their storage, processing and analysis. To address this issue, *data reduction* is commonly considered, either by employing lossy compression schemes [Lin14, BRLP19] or by storing reduced data representations, which concisely, yet precisely, only encode the core features of the data, possibly during data production (i.e., *in-situ* [BAA+16, ABG+15]).

Topological Data Analysis (TDA) [EH09] precisely focuses on the robust encoding of the structural patterns of a dataset into concise *topological descriptors*, whose successful practical applications have been documented in a variety of domains and contexts [HLH+16].

Topological descriptors are often used as a mean of data reduction, typically by storing to disk these descriptors instead of the data itself. For instance, for time-varying datasets, a typical strategy [BWT+11, BNP+21, FPF+23] consists in storing the actual data at a low frequency (resulting in the storage of a small number n of *keyframes*), while storing topological

descriptors at a higher frequency (e.g., for a large number $N \gg n$ of *non-keyframe* time steps). Then, the produced descriptors can be directly post-processed by dedicated statistical frameworks [TMMH14, LCO18, VBT20, YWM+19, PVDT22, PVT23, PT24], tailored to the analysis of topological representations. However, in many scenarios, a complete investigation might require going back to the original dataset which initially generated a given topological descriptor, for instance, for further visual inspection, interpretation and analysis. Then, the following question arises: *how can we reliably* "invert" *the construction of a topological descriptor?* (i.e., retrieve the dataset which generated it).

This thesis explores these challenges and contributes novel methods at the intersection of data reduction, topological simplification, and reconstruction, with the goal of making topological analysis more practical and impactful for large-scale scientific data.

1.1 General Context and Motivations

1.1.1 Data analysis and visualization

Topological Data Analysis (TDA) [EH09, Zom10] operates at the intersection of computer science and applied mathematics, providing generic, robust, and multi-scale techniques for the extraction, quantification, and comparison of hidden structural features within complex datasets. It leverages foundational concepts in topology that are well-established in the scientific literature and offers a variety of powerful topological data abstractions, such as critical points [Ban67], persistence diagrams [ELZ02, BKR14, GVT23], merge [CWSA16, GFJT17, LWW+24] and contour trees [CSA00, GFJT19a], Reeb graphs [BGSF08, PSBM07, GFJT19b], and Morse-Smale complexes [GBHP08, RWS11, SN12, GBP19, MLT+23].

In recent years, TDA has seen a rapid growth, driven by its notable successes in data science and machine learning, as well as by the availability of modern open-source tools that simplify its implementation. It has been successfully applied to numerous data analysis problems across various application domains [HLH+16], including turbulent combustion [BWT+11, GBG+14], fluid dynamics [KRHH11, NVBB+22], material sciences [GKL+16, SPD+19], nuclear energy [MWR+16], bioimaging [BDSS18, AAPW18], quantum chemistry [BGL+18, OGT19, OT23], and astrophysics [Sou11, SPN+16]. TDA's adaptability and effectiveness in capturing structural insights make it an invaluable approach in these diverse fields.

1.1.2 Thesis Environment - The TORI Project

This thesis was carried out as part of a collaboration between the CNRS and the University of Arizona, in connection with the European TORI project, introduced below. The thesis was supervised by Julien Tierny (CNRS research director, expert in topological methods for data analysis and visualization, https://julien-tierny.github.io/), with occasional collaborations with Josh Levine (professor at the University of Arizona, specializing in neural networks for visualization and expert in topological methods for data analysis and visualization).

The TORI project (Topological Reduction of Information) is a research initiative funded by the European Research Council (ERC), hosted by the French National Center for Scientific Research (CNRS) at Sorbonne

University, and launched in October 2020.

Scheduled to run for six years, until September 30, 2026, TORI brings together researchers, students, and engineers focused on advancing computational methods within the field of Topological Data Analysis (TDA). The project's primary objective is to design novel algorithms for interactive analysis of large-scale datasets, emphasizing concise and meaningful structural representations. Additionally, TORI aims to develop innovative techniques for comparing these topological data abstractions and efficiently computing them at scale.

The algorithms and tools developed by the TORI project are implemented and distributed through the open-source software library, Topology ToolKit (TTK), described in the next section.

1.1.3 The Topology ToolKit (TTK)

The **Topology ToolKit** (TTK) is an open-source library designed for data analysis and visualization with topological methods. It is efficient, versatile, and user-friendly. It is particularly well-suited for processing scalar data defined on regular grids or triangulations in spaces of low dimension, and provides a rich collection of robust, efficient, and generic algorithms for topological data analysis. Developed in C++, TTK offers VTK/C++ and Python bindings as well as standalone command-line tools, making it highly flexible and easy to integrate into diverse workflows. Its modular design allows users to extend its functionality by creating custom analysis modules, making it straightforward to adapt TTK to specific research or industrial needs. Notably, the algorithm discussed in Chapter 3 is implemented directly within the TTK library, and the examples presented in that chapter are also available among the official TTK example datasets on its website. Furthermore, with dedicated plugins for ParaView, TTK facilitates intuitive exploration and processing of data in various formats, bringing advanced topological analysis within easy reach. Distributed under the BSD license, it can be freely used in both open-source and commercial projects, and is supported by comprehensive documentation, detailed tutorials, and numerous examples that encourage its adoption and customization across both academic and industrial domains.

1.2 Problem Formulation

In the previously described application context, persistence diagrams alone are often insufficient for performing advanced interpretation, analysis, or visualization tasks. Indeed, fully leveraging the information encoded in persistence diagrams frequently requires reverting back to the original scalar data or at least to a faithful approximation of it. This requirement is particularly critical in scenarios involving time-varying datasets, where scalar fields are stored only at selected keyframes, while persistence diagrams are computed and stored at a significantly higher temporal resolution.

Consequently, reliably reconstructing scalar fields from persistence diagrams becomes an essential issue. This general problem can be further divided into two specific subproblems:

- **Problem 1:** Existing approaches for optimizing the topology of scalar fields relying on persistence diagrams are computationally expensive; how can these optimization methods be significantly accelerated to make topological simplification optimization practical for real-world applications? In particular, how can we efficiently remove noisy persistence pairs while precisely preserving the important pairs, including saddle pairs in three-dimensional scalar datasets?
- **Problem 2:** How can we interpolate or reconstruct a plausible scalar field from one or several persistence diagrams; specifically, how can we ensure that the reconstructed scalar field accurately respects the topological constraints encoded in the persistence diagram?

This manuscript proposes novel methodological solutions specifically addressing these two challenges, with the goal of expanding the practical applicability of topological analysis tools for managing and analyzing large-scale scientific datasets.

1.3 Contributions

This thesis addresses the two main challenges identified in Sec. 1.2 by proposing new methods that advance both the simplification and the interpolation of scalar data under topological constraints.

To tackle **Problem 1**, which concerns the prohibitive computational cost of existing optimization-based approaches for topological simplification of scalar fields guided by persistence diagrams, we propose a practical solver specifically designed for this task. Our approach introduces two dedicated accelerations: a fast update strategy for the persistence diagram during the optimization process, which avoids the need to recompute it entirely at every iteration, and a rapid update of the pair assignments between the target diagram and the current diagram of the evolving scalar field. These improvements yield substantial speed-ups compared to existing persistence optimization frameworks, making the optimization of topological simplification practical even for large three-dimensional datasets. This enables direct applications such as the visualization of simplified isosurfaces with fewer connected components and handles, the extraction of prominent filament structures, and explicit surface genus repair. To support reproducibility and further research, we provide a C++ implementation of our algorithm.

To address **Problem 2**, namely how to interpolate scalar fields from one or several persistence diagrams while ensuring that the resulting fields respect the intended topological constraints, we develop a novel framework that integrates topological information directly into a neural generative model. Our method combines a time-parameterized neural network for scalar field interpolation with specialized loss functions based on persistence diagrams, explicitly enforcing topological and geometric consistency in the interpolated outputs. Implemented using TTK and PyTorch, this approach offers a practical and reproducible solution for time-varying scalar field interpolation tasks, significantly improving over baseline techniques by embedding topological awareness directly into the learning process.

Taken together, these contributions provide complementary solutions to the challenges of simplifying and interpolating scalar data within topological pipelines. They help bridge the gap between abstract persistence

diagrams and the rich, detailed scalar fields they summarize, thereby extending the practical reach of topological data analysis tools for large-scale scientific data exploration.

1.4 Outline

The remainder of this manuscript is structured as follows:

- **Chapter 2** provides the required theoretical foundations of Topological Data Analysis along with a general overview.
- **Chapter 3** presents our practical solver for scalar data topological simplification.
- **Chapter 4** describes a topology aware neural interpolation of scalar fields.
- **Chapter 5** summarizes the contributions of this thesis, discusses current limitations, and explores potential future directions.

Theoretical Background

2

Contents

This section provides the theoretical foundations necessary to understand the methods and concepts used throughout this work. We begin by introducing the nature of the input data and the types of scalar fields considered (Sec. 2.1). We then present persistence diagrams, a central tool in topological data analysis that captures the birth and death of topological features across scales (Sec. 2.2). In Section Sec. 2.3, we detail the Wasserstein distance, a commonly used metric to compare persistence diagrams. This is followed by a presentation of persistence optimization (Section Sec. 2.4), which aims to refine or simplify topological features while preserving relevant structures. Finally, Section Sec. 2.5 provides a general introduction to neural networks, setting the stage for their later use in data-driven tasks within this thesis.

2.1 Input data

The input data is provided as a piecewise-linear (PL) scalar field $f : \mathcal{K} \rightarrow \mathbb{R}$ defined on a d-dimensional simplicial complex $\mathcal{K}$ (with $d \leq 3$ in our applications). If the data is provided on a regular grid, we consider for $\mathcal{K}$ the implicit Freudenthal triangulation of the grid [H. 42, H.W60].

In practice, the data values are defined on the n_v vertices of $\mathcal{K}$, in the form of a *data vector*, noted $v_f \in \mathbb{R}^{n_v}$. f is assumed to be injective on the vertices (i.e., the entries of v_f are all distinct), which can be easily obtained in practice via a variant of simulation of simplicity [EM90]. To facilitate parameter tuning and comparisons, scalar values are normalized between 0 and 1 in all experimental setups.

2.2 Persistence Diagrams

Persistent homology has been developed independently by several research groups [Bar94, FL99, Rob99, ELZ02]. Intuitively, persistent homology considers a sweep of the data (i.e., a *filtration*) and estimates at each step the corresponding topological features (i.e., *homology generators*), as well as maps to the features of the previous step. This enables the identification of the topological features, along with their lifespan, during the sweep.

In this work we consider the *lexicographic filtration* (as described in [GVT23]), which we briefly recall here for completeness. Given the input data vector $v_f \in \mathbb{R}^{n_v}$, one can sort the vertices of $\mathcal{K}$ by increasing data values, yielding a *global vertex order*. Based on this order, each d'-simplex $\sigma \in \mathcal{K}$ (with $d' \in [0, d]$) can be represented by the sorted list (in decreasing values) of the $(d'+1)$ indices in the global vertex order of its $(d'+1)$ vertices. Given this simplex representation, one can now compare two simplices σ_i and σ_j via simple lexicographic comparison, which induces a *global lexicographic order* on the simplices of $\mathcal{K}$. This order induces a nested sequence of simplicial complexes $\emptyset = \mathcal{K}_0 \subset \mathcal{K}_1 \subset \cdots \subset \mathcal{K}_{n_\sigma} = \mathcal{K}$ (where n_σ is the number of simplices of $\mathcal{K}$), which we call the *lexicographic filtration* of $\mathcal{K}$ by f.

At each step i of the filtration, one can characterize the p^{th} homology group of $\mathcal{K}_i$, noted $\mathcal{H}_p(\mathcal{K}_i)$, for instance by counting its number of *homology classes* [EH09, GVT23] (i.e., the *order* of the group) or its number of *homology generators* (i.e., the *rank* of the group, a.k.a. the p^{th} *Betti number*, noted β_p). Intuitively, in 3D, the first three Betti numbers (β_0, β_1, and β_2)

respectively provide the number of connected components, of independent cycles and voids of the complex $\mathcal{K}_i$. For two consecutive steps of the filtration i and j, the corresponding simplicial complexes are nested ($\mathcal{K}_i \subset \mathcal{K}_j$). This inclusion induces homomorphims between the homology groups $\mathcal{H}_p(\mathcal{K}_i)$ and $\mathcal{H}_p(\mathcal{K}_j)$, mapping homology classes at step i to homology classes at step j. Intuitively, for the 0^{th} homology group, one can precisely map a connected component at step i to a connected component at step j because the former is included in the latter. In general, a p-dimensional homology class γ_i at step i can be mapped to a class γ_j at step j if the p-cycles of γ_i and γ_j are *homologous* in $\mathcal{K}_j$ [EH09, GVT23]. Then, one can precisely track the homology generators between consecutive steps of the filtration. In particular, a *persistent generator* is born at step j (with $j = i + 1$) if it is not the image of any generator by the homomorphims mapping $\mathcal{H}_p(\mathcal{K}_i)$ to $\mathcal{H}_p(\mathcal{K}_j)$. Symmetrically, a persistent generator dies at step j if it merges with another, *older* homology class, which was born before it (this is sometimes called the *Elder rule* [EH09]). Each p-dimensional persistent generator is associated to a *persistence pair* (σ_b, σ_d), where σ_b is the p-simplex introduced at the *birth* of the generator (at step b) and where σ_d is the $(p+1)$-simplex introduced at its *death* (at step d).

A p-simplex which is involved in the birth or the death of a generator is called a *critical simplex* and, in 3D, we call it a minimum, a 1-saddle, a 2-saddle, or a maximum if p equals 0, 1, 2, or 3 [RWS11, GVT23], respectively. The *persistence* of the pair (σ_b, σ_d), noted $p(\sigma_b, \sigma_d)$, is given by $p(\sigma_b, \sigma_d) = v_f(v_d) - v_f(v_b)$, where v_b and v_d (the *birth* and *death vertices* of the pair) are the vertices with highest global vertex order of σ_b and σ_d.

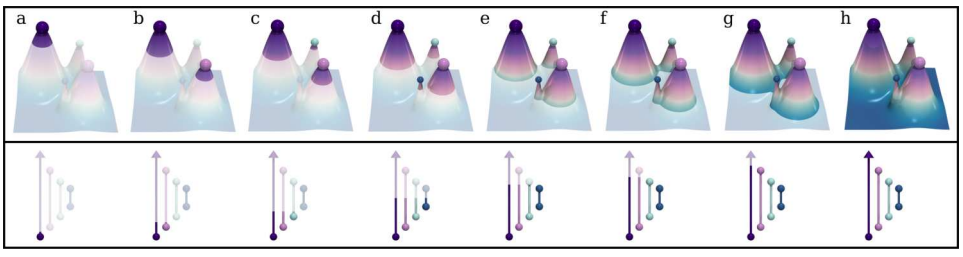

Figure 2.1 – *Filtration of the opposite elevation function (going downward, purple: low values, cyan: high values) on a toy terrain example (in transparent). From left to right, simplices are progressively added in the filtered simplicial complex for increasing values of opposite elevation. Each local minimum triggers the* birth *of a connected component in the complex (sub-figures* a *to* d*). Connected components are represented by growing bars of matching color in the diagram (bottom). When a component merges with an* older *one (sub-figures* e*,* f *and* g*), its corresponding bar terminates its growth in the diagram and the corresponding topological feature is said to* die *at the corresponding value.*

Note that the generators characterizing the homology groups of the input complex $\mathcal{K}$ are said to have *infinite persistence*. This process is illustrated in Fig. 2.1 for the specific case of the 0-dimensional persistent homology (representing persistent connected components).

We call *zero-persistence pairs* the pairs with $v_b = v_d$. Some p-simplices of $\mathcal{K}$ may be involved in no persistence pair. These mark the birth of persistent generators with *infinite persistence* (i.e., which never die during the filtration) and they characterize the homology groups of the final step of the filtration ($\mathcal{K}_{n_\sigma} = \mathcal{K}$).

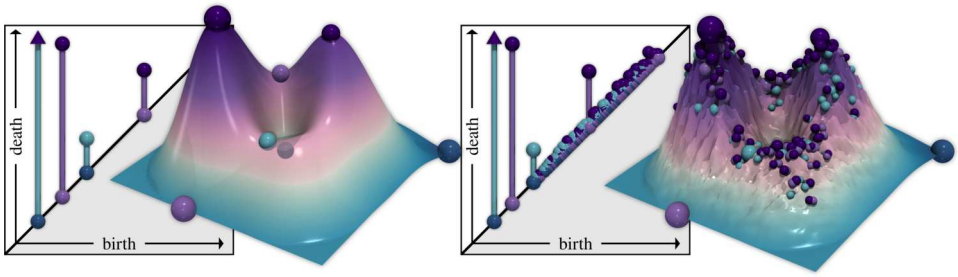

Figure 2.2 – *Persistent diagrams for the lexicographic filtration from bottom to top of a clean (left) and a noisy (right) terrain example. Minimum-saddle persistence pairs are show with cyan bars in the birth-death space, while saddle-maximum pairs are shown with purple bars. Generators with infinite persistence are marked with an upward arrow. The persistence of each topological feature is given by the height of its bar. Critical simplices are shown in the data with spheres, with a radius proportional to their persistence.*

The set of persistence pairs induced by the lexicographic filtration of $\mathcal{K}$ by f can be organized in a concise representation called the *persistence diagram* (Fig. 2.2), noted $\mathcal{D}(f)$, which embeds each non zero-persistence pair (σ_b, σ_d) as a point in a 2D space (called the birth-death space), at coordinates $\big(v_f(v_b), v_f(v_d)\big)$. By convention, generators with infinite persistence are reported at coordinates $\big(v_f(v_b), v_f(v_{max})\big)$, where v_{max} is the last vertex in the global vertex order.

2.3 Wasserstein distance between persistence diagrams

Two diagrams $\mathcal{D}(f)$ and $\mathcal{D}(g)$ can be reliably compared in practice with the notion of *Wasserstein distance*. For this, the two diagrams $\mathcal{D}(f)$ and $\mathcal{D}(g)$ need to undergo an *augmentation* pre-processing phase. This step ensures that the two diagrams admit the same number of points, which will facilitate their comparison. Given a point $p = (p_b, p_d) \in \mathcal{D}(f)$, we note $\Delta(p)$ its diagonal projection: $\Delta(p) = \big(\frac{1}{2}(p_b + p_d), \frac{1}{2}(p_b + p_d)\big)$. Let Δ_f and Δ_g be the sets of the diagonal projections of the points of $\mathcal{D}(f)$

and $\mathcal{D}(g)$ respectively. Then, $\mathcal{D}(f)$ and $\mathcal{D}(g)$ are *augmented* by appending to them the set of diagonal points Δ_g and Δ_f respectively. After this augmentation, we have $|\mathcal{D}(f)| = |\mathcal{D}(g)|$.

Then, given two augmented persistence diagrams $\mathcal{D}(f)$ and $\mathcal{D}(g)$, the L^q Wasserstein distance between them is defined as:

$$\mathcal{W}_q(\mathcal{D}(f), \mathcal{D}(g)) = \min_{\phi \in \Phi} \Big(\sum_{p \in \mathcal{D}(f)} c(p, \phi(p))^q \Big)^{\frac{1}{q}}, \tag{2.1}$$

where Φ is the set of all bijective maps between the augmented diagrams $\mathcal{D}(f)$ and $\mathcal{D}(g)$, which specifically map points of finite (respectively infinite) r-dimensional persistent generators to points of finite (respectively infinite) r-dimensional persistent generators. For this distance, the cost $c(p, p')$ is set to zero when both p and p' lie on the diagonal (i.e., matching dummy features has no impact on the distance). Otherwise, it is set to the Euclidean distance in birth-death space $||p - p'||_2$.

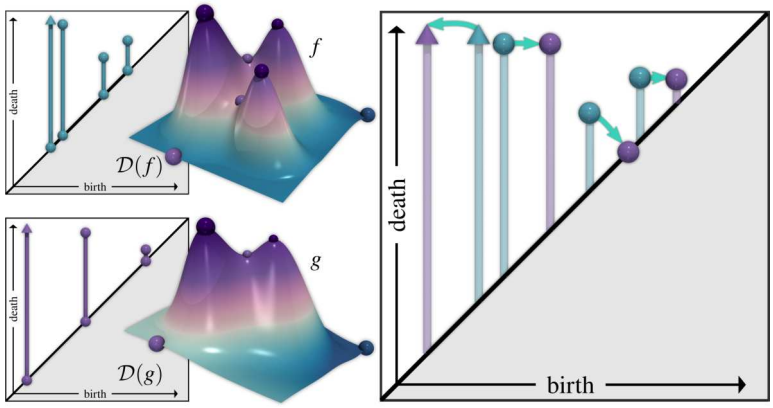

Figure 2.3 – *The Wasserstein distance $\mathcal{W}_2$ between $\mathcal{D}(f)$ (top) and $\mathcal{D}(g)$ (bottom) is computed by assignment optimization (Eq. 2.1) in the 2D birth-death space (right). The optimal assignment ϕ^* (arrows) encodes a minimum cost transformation of $\mathcal{D}(f)$ into $\mathcal{D}(g)$, which displaces persistence pairs in the birth-death space or* cancel *them by sending them to the diagonal.*

The Wasserstein distance induces an optimal assignment ϕ^* from $\mathcal{D}(f)$ to $\mathcal{D}(g)$ (Fig. 2.3), which depicts how to minimally transform $\mathcal{D}(f)$ into $\mathcal{D}(g)$ (given the considered cost). This transformation may induce point displacements in the birth-death space, as well as projections to the diagonal (encoding the *cancellation* of a persistence pair).

2.4 Persistence optimization

Several frameworks have been introduced for persistence optimization (Sec. 3.1.1). We review a recent, efficient, and generic framework [CCG+21].

Given a scalar data vector $v_f \in \mathbb{R}^{n_v}$ (Sec. 2.1), the purpose of persistence optimization is to modify v_f such that its persistence diagram $\mathcal{D}(f)$ minimizes a certain loss $\mathcal{L}$, specific to the considered problem. Then the solution space of the optimization problem is $\mathbb{R}^{n_v}$.

Let $\mathcal{F} : \mathbb{R}^{n_v} \to \mathbb{R}^{n_\sigma}$ be the *filtration map*, which maps a data vector v_f from the solution space $\mathbb{R}^{n_v}$ to a filtration represented as a vector $\mathcal{F}(v_f) \in \mathbb{R}^{n_\sigma}$, where the i^{th} entry contains the index of the i^{th} simplex σ_i of $\mathcal{K}$ in the global lexicographic order (Sec. 2.2). For convenience, we maintain a *backward filtration map* $\mathcal{F}^+ : \mathbb{R}^{n_\sigma} \to \mathbb{R}^{n_\sigma}$, which maps a filtration vector $\mathcal{F}(v_f)$ to a vector in $\mathbb{R}^{n_\sigma}$, whose i^{th} entry contains the index of the highest vertex (in global vertex order) of the i^{th} simplex in the global lexicographic order.

Given a persistence diagram $\mathcal{D}(f)$, the *critical simplex persistence order* can be introduced as follows. First, the points of $\mathcal{D}(f)$ are sorted by increasing birth and then, in case of birth ties, by increasing death. Let us call this order the *diagram order*. Then the set of persistence pairs can also be sorted according to the diagram order, by interleaving the birth and death simplices corresponding to each point. This results in an ordering of the critical simplices, called the *critical simplex persistence order*, where the $(2i)^{th}$ and $(2i+1)^{th}$ entries correspond respectively to the birth and death simplices of the i^{th} point p_i in the diagram order. Critical simplices which are not involved in a persistence pair (i.e., corresponding to homology classes of infinite persistence) are appended to this ordering, in increasing order of birth values.

Let us now consider the *persistence map* $\mathcal{P} : \mathbb{R}^{n_\sigma} \to \mathbb{R}^{n_\sigma}$, which maps a filtration vector $\mathcal{F}(v_f)$ to a persistence image $\mathcal{P}\big(\mathcal{F}(v_f)\big)$, whose i^{th} entry contains the *critical simplex persistence order* (defined above) for the i^{th} simplex in the global lexicographic order. For convenience, the entries corresponding to filtration indices which do not involve critical simplices are set to -1.

Now, to evaluate the relevance of a given diagram $\mathcal{D}(f)$ for the considered optimization problem, one needs to define a loss term. Let $\mathcal{E} : \mathbb{R}^{n_\sigma} \to \mathbb{R}$ be an energy function, which evaluates the diagram energy given its critical simplex persistence order. Then, given an input data

vector v_f, the associated loss $\mathcal{L} : \mathbb{R}^{n_v} \to \mathbb{R}$ is given by:

$$\mathcal{L}(v_f) = \mathcal{E} \circ \mathcal{P} \circ \mathcal{F}(v_f).$$

Since distinct functions can admit the same persistence diagram, the global minimizer of the above loss may not be unique. However, given the search space ($\mathbb{R}^{n_v}$), the search for a global minimizer is not tractable anyway in practice and local minimizers will be searched instead.

If $\mathcal{E}$ is locally Lipschitz and a definable function of persistence, then the composition $\mathcal{E} \circ \mathcal{P} \circ \mathcal{F}$ is also definable and locally Lipschitz [CCG+21]. This implies that $\mathcal{L}$ is differentiable almost everywhere and admits a well defined sub-differential. Then, a stochastic sub-gradient descent algorithm [KB15] converges almost surely to a critical point of $\mathcal{L}$ [DDKL20]. In practice, this means that the loss can be decreased by displacing each diagram point p_i in the diagram $\mathcal{D}(f)$ according to the sub-gradient. Assuming a constant global lexicographic order, this displacement can be back-propagated into modifications in data values in the vector v_f, by identifying the vertices v_{i_b} and v_{i_d} corresponding to the birth and death (Sec. 2.2) of the i^{th} point in the diagram order:

$$\begin{aligned} v_{i_b} &= \mathcal{F}^+\big(\mathcal{P}^{-1}(2i)\big) \\ v_{i_d} &= \mathcal{F}^+\big(\mathcal{P}^{-1}(2i+1)\big), \end{aligned} \tag{2.2}$$

and by updating their data values $v_f(v_{i_b})$ and $v_f(v_{i_d})$ accordingly.

2.5 Neural Networks

This section presents the essential Machine Learning principles and techniques relevant to the neural architectures used in this thesis. We specifically focus on deep learning methods suitable for analyzing and generating complex volumetric data, introducing convolutional neural networks, normalization methods, residual learning, upsampling techniques, and activation functions.

2.5.1 Convolutional Neural Networks (CNNs)

CNNs constitute a class of deep neural networks specifically designed to handle structured spatial or volumetric data such as images or three-dimensional scalar fields [LBH15]. The key component of CNNs is the convolutional layer, which applies filters (kernels) across an input volume to capture local spatial correlations.

In particular, 3D convolutional layers (Conv3D) employ volumetric kernels sliding across the spatial dimensions, producing a structured set of feature maps representing different spatial features. Formally, a convolution operation is expressed as follows:

$$(V * K)(i,j,k) = \sum_{p=-a}^{a} \sum_{q=-b}^{b} \sum_{r=-c}^{c} V(i-p, j-q, k-r) \cdot K(p,q,r), \tag{2.3}$$

where V denotes the input volume indexed by spatial coordinates (i,j,k), and K is the convolutional kernel. By stacking multiple convolutional layers, CNNs progressively build hierarchical representations capable of capturing increasingly complex patterns in data.

2.5.2 Normalization Techniques

Normalization is widely employed to improve training stability and speed up convergence of neural networks by controlling the internal distribution of activations. Different normalization methods exist, including Batch Normalization [IS15], Layer Normalization [BKH16], and Instance Normalization [UVL16].

Instance Normalization (InstanceNorm3D), extensively used in generative tasks, normalizes each individual instance separately. It operates independently across spatial dimensions for each feature map and each sample, defined as follows:

$$y = \frac{x - \mu(x)}{\sqrt{\sigma^2(x) + \epsilon}}, \tag{2.4}$$

where x denotes the input activation tensor, $\mu(x)$ and $\sigma^2(x)$ are the mean and variance computed per-instance and per-channel, and ϵ is a small constant for numerical stability. Here, a *channel* refers to a feature map dimension, typically representing different learned filters in the output of a convolutional layer. For example, in a 3D convolution, a tensor may have dimensions (batch, channels, depth, height, width).

Instance normalization normalizes each sample independently, which reduces the impact of variations in contrast, illumination, or texture. This effect is often described as improving *style invariance*, meaning the network becomes less sensitive to superficial style changes and focuses more on structural content, which helps the model generalize better across different data distributions.

2.5.3 Residual Blocks (ResBlocks)

Residual networks were introduced to overcome the challenges of training very deep neural networks, notably the vanishing gradient problem [HZRS16]. The central concept is the skip connection or residual connection, enabling information to bypass certain layers and directly flow into deeper parts of the network. Formally, a residual block computes:

$$y = \mathcal{R}(x, W) + x, \tag{2.5}$$

where x and y are the input and output tensors of the residual block, respectively, and $\mathcal{R}(x, W)$ denotes the learned residual mapping, typically consisting of convolutional, activation, and normalization layers.

Such residual connections simplify optimization, allow training of deeper networks, and generally lead to improved performance in a wide variety of tasks.

2.5.4 Upsampling Operations

Upsampling operations increase the spatial or volumetric resolution of feature maps and are essential for tasks such as data generation, segmentation, or reconstruction. Several methods are commonly used for upsampling, including interpolation (nearest-neighbor, bilinear, trilinear) and transposed convolution (also called fractionally-strided convolution).

In this thesis, we use interpolation-based upsampling followed by convolutional refinement to progressively reconstruct high-resolution volumetric data. The choice of interpolation method affects the smoothness and quality of reconstructed features and is crucial for preserving spatial details in generated or interpolated scalar fields.

2.5.5 Activation Functions

Activation functions introduce nonlinear transformations within neural networks, which are crucial for modeling complex patterns:

ReLU (Rectified Linear Unit)

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2.6}$$

ReLU is preferred due to its computational simplicity, ease of optimization, and reduced likelihood of vanishing gradients.

Sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.7}$$

Sigmoid functions map outputs between 0 and 1, making it suitable for tasks involving probabilistic interpretations or normalized predictions, such as final-layer activations in volumetric data reconstruction.

2.5.6 Integration within the Proposed Architecture

The methods presented above are integrated into the neural architecture exploited in Chapter 4. Specifically, convolutional blocks, normalization layers, residual connections, upsampling operations, and activation functions are strategically combined to construct a powerful generative model capable of accurately reconstructing missing temporal snapshots in time-varying scalar fields. The detailed description and evaluation of this architecture are provided in chapter 4.

3 A practical solver for scalar data topological simplification

Contents

This chapter presents a practical approach for the optimization of topological simplification, a central pre-processing step for the analysis and visualization of scalar data. Given an input scalar field f and a set of *"signal"* persistence pairs to maintain, our approaches produces an output field g that is close to f and which optimizes *(i)* the cancellation of

"non-signal" pairs, while *(ii)* preserving the *"signal"* pairs. In contrast to pre-existing simplification algorithms, our approach is not restricted to persistence pairs involving extrema and can thus address a larger class of topological features, in particular saddle pairs in three-dimensional scalar data. Our approach leverages recent generic persistence optimization frameworks and extends them with tailored accelerations specific to the problem of topological simplification. Extensive experiments report substantial accelerations over these frameworks, thereby making topological simplification optimization practical for real-life datasets. Our approach enables a direct visualization and analysis of the topologically simplified data, e.g., via isosurfaces of simplified topology (fewer components and handles). We apply our approach to the extraction of prominent filament structures in three-dimensional data. Specifically, we show that our pre-simplification of the data leads to practical improvements over standard topological techniques for removing filament loops. We also show how our approach can be used to repair genus defects in surface processing. Finally, we provide a C++ implementation for reproducibility purposes.

The work presented in this chapter has been published in the IEEE Transactions on Visualization and Computer Graphics as part of the proceedings of the IEEE VIS 2024 conference [KPLT24]. It has been certified as replicable by the Graphics Replicability Stamp Initiative (http://www.replicabilitystamp.org/). Our implementation is available at: https://github.com/MohamedKISSI/Code-Paper-A-Pratical-Solver-for-Scalar-Data-Topological-Simplification.git, and it is also integrated into the Topology ToolKit (TTK) [TFL+17]. The examples presented in this section are also available on the TTK website.

Our Contributions in one Image

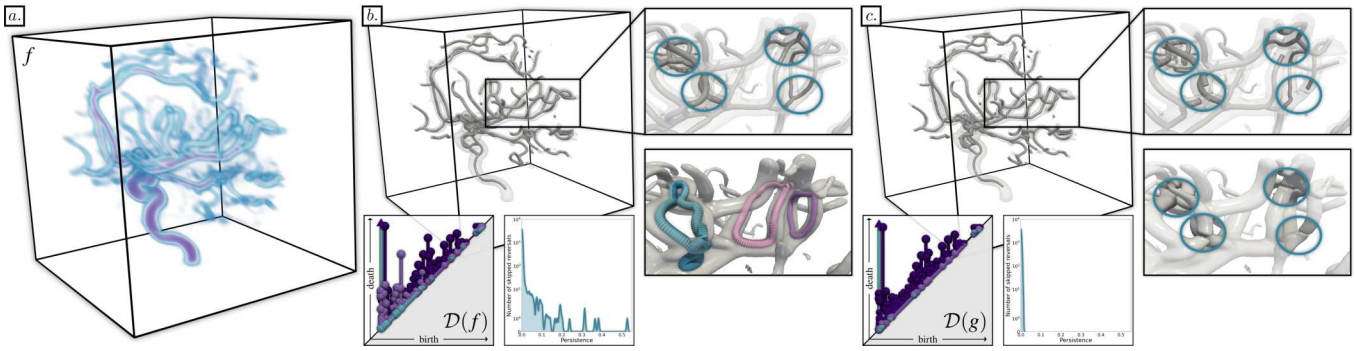

Figure 3.1 – *Given an acquired scalar field f of a network of arteries (a), the core structure of the blood vessels can be extracted (grey filaments, (b)) with upward discrete integral lines, started at* 2*-saddles above* 0.1 *(isovalue representing the geometry of the vessels, transparent isosurface). However, as shown in the persistence diagram $\mathcal{D}(f)$, f contains many saddle-pairs (light purple bars), corresponding to persistent* 1*- dimensional generators [GVT23, Iur21] (curves, colored by persistence, bottom zoom, (b)), yielding incorrect loops in the filament structures (top zoom). In this example, standard techniques for gradient field simplification (i.e., saddle connector reversal) cannot simplify these spurious loops while maintaining a valid gradient (b), as shown in the bottom histogram (number of* skipped *reversals as a function of persistence). Our approach efficiently generates a function g which is close to f and which optimizes saddle pair cancellation while maintaining the other features $(\mathcal{D}(g))$. This enables the direct visualization and analysis of the simplified data (c), where isosurface handles have been cut (bottom zoom) and most spurious filament loops have been simplified (top zoom).*

3.1 Context

As acquisition devices and computational resources are getting more sophisticated and efficient, modern datasets are growing in size. Consequently, the features of interest contained in these datasets gain in geometrical complexity, which challenge their interpretation and analysis. This motivates the design of tools capable of robustly extracting the structural patterns hidden in complex datasets. This task is the purpose of Topological Data Analysis (TDA) [EH09, Zom10], which provides a family of techniques for the generic, robust and multi-scale extraction of structural features. It has been successfully applied in a number of data analysis problems [HLH+16], in various applications, including turbulent combustion [BWT+11, GBG+14], material sciences [GKL+16, SPD+19], nuclear energy [MWR+16], fluid dynamics [KRHH11, NVBB+22], bioimaging [BDSS18, AAPW18], quantum chemistry [BGL+18, OGT19, OT23], or astrophysics [Sou11, SPN+16]. TDA provides a variety of *topological data abstractions*, which enable the extraction of specific types of features of interest. These abstractions include critical points [Ban67], persistence diagrams [ELZ02, BKR14, GVT23], merge [CWSA16, GFJT17, LWW+24] and contour trees [CSA00, GFJT19a], Reeb graphs [BGSF08, PSBM07, GFJT19b], or Morse-Smale complexes [GBHP08, RWS11, SN12, GBP19, MLT+23]. A central aspect of TDA is its ability to analyze data at multiple scales. Thanks to various importance measure [ELZ02, CSvdP04], these abstractions can be iteratively simplified, to reveal the prominent structures in a dataset.

In practice, this topological simplification can be achieved in two fashions: either by *(i)* a post-process simplification of the abstractions, or by *(ii)* a pre-process simplification of the data itself. While the post-process approach *(i)* requires specific simplification mechanisms tailored to the abstraction at hand [CSvdP04, PSBM07, Gyu08], the pre-process strategy offers a generic framework which is independent of the considered abstraction. This generic aspect eases software design, as simplification needs to be implemented only once [TFL+17, BMBF+19]. Pre-process simplification has also the advantage of being reusable by multiple abstractions when these are combined within a single data analysis pipeline (see [TTK22] for real-life examples). Also, pre-process simplification enables the direct visualization of the simplified data itself (e.g. with isosurfaces). Finally, it is also compatible with further post-process simplification if needed. For these reasons, we focus on pre-process simplification in this work.

Several combinatorial approaches [Soi04, EMP06, AAY06, AGLM09, BLW12, TP12, LGMT20] have been proposed for the pre-simplification of persistence pairs involving extrema. However, no efficient combinatorial algorithm has been proposed for the pre-simplification of saddle pairs, hence preventing a more advanced simplification of 3D datasets. In fact, scalar data simplification in 3D is NP-hard in general [ABD+13]. Then, there may not even exist polynomial time algorithms for directly solving this problem. This theoretical limitation requires a shift in strategy. A recent alternative consists in considering persistence optimization frameworks [CCG+21, SWB21, NM22], which optimize the data in a *best effort* manner, given criteria expressed with persistence diagrams. However, while one could leverage these frameworks for data pre-simplification (i.e., to cancel noisy features while preserving the features of interest, *as much as possible*), current frameworks can require up to days of computation for regular grids of standard size (Sec. 3.2), making them impractical for real-life datasets.

This chapter addresses this issue and introduces a practical solver for the optimization of the topological simplification of scalar data. Our approach relies on a number of pragmatic observations from which we derived specific accelerations, for each sub-step of the optimization (Secs. 3.3.2 and 3.3.3). Our accelerations are simple and easy to implement, but result in significant gains in terms of runtimes. Extensive experiments (Sec. 3.4.1) report $\times 60$ accelerations on average over state-of-the-art frameworks (with both fewer and faster iterations), thereby making topological simplification optimization practical for real-life datasets. We illustrate the utility of our contributions in two applications. First, our work enables the direct visualization and analysis of topologically simplified data (Sec. 3.4.2). This reduces visual clutter in isosurfaces by simplifying their topology (fewer components and handles). We also investigate filament extraction in three-dimensional data, where we show that our approach helps standard topological techniques for removing filament loops. Second, we show how to use our approach to repair genus defects in surface processing (Sec. 3.4.3).

3.1.1 Related work

Beyond post-process simplification schemes tailored for specific topological abstractions (e.g. for merge/contour trees [Car04], Reeb graphs [PSBM07] or Morse-Smale complexes [Gyu08, GBHP09, GRSW13]), the

literature related to pre-process simplification can be classified into two categories.

Combinatorial methods: the first combinatorial approach for the topological simplification of scalar data on surfaces has been proposed by Edelsbrunner et al. [EMP06]. This work can be seen as a generalization of previous approaches in terrain modeling where only persistence pairs involving minima were removed [Soi04, AAY06]. Attali et al. [AGLM09] extended this framework to generic filtrations, while Bauer et al. [BLW12] extended it to discrete Morse theory [For98]. Tierny et al. presented a generalized approach [TP12], supporting a variety of simplification criteria, which was later extended by Lukasczyk et al. [LGMT20] with an efficient shared-memory parallel algorithm. Such combinatorial simplification algorithms can be used directly within optimization procedures [NKSM24], to remove noise in the solution at each iteration. While most of the above approaches were specifically designed for scalar data on surfaces, they can be directly applied to domains of higher dimensions. However, they can only simplify persistence pairs involving extrema. For instance, this means that they cannot remove saddle pairs in three-dimensional scalar fields, thus preventing an advanced simplification of this type of datasets. In general, scalar data simplification in 3D has been shown to be NP-hard [ABD$^+$13]. Then, a polynomial time algorithm solving this problem may not even exist. This theoretical limitation requires a shift in strategy.

Numerical methods: in contrast to combinatorial methods, which come with strong guarantees on the result, numerical approaches aim at providing an approximate solution in a best effort manner. In other words, these methods may not *fully* simplify three-dimensional scalar fields up to the desired tolerance either, but they will do their best to provide a result as close as possible to the specified simplification. As such, this type of approaches appear as a practical alternative overcoming the theoretical limitation of combinatorial approaches discussed above. In geometric modeling, several techniques have been described to generate smooth scalar fields on surfaces, with a minimal number of critical points [NGH04, GZ07, PF09]. Bremer et al. [BEHP04] proposed a method based on Laplacian smoothing to reconstruct a two-dimensional scalar field corresponding to a pre-simplified Morse-Smale complex. This work has been extended by Weinkauf et al. [WGS10] to bi-Laplacian optimization, with an additional enforcement of gradient continuity across the separatrices of the Morse-Smale complex. While an extension of this work has been documented for the 3D case [GJR$^+$14], it only addresses the simplifica-

tion of persistence pairs involving extrema, without explicit control on the saddle pairs. Recently, a new class of methods dedicated to *persistence optimization* has been documented. Specifically, these approaches introduce a framework for optimizing a dataset, according to criteria expressed with persistence diagrams, with applications in various tasks including surface matching [PSO18], point cloud processing [GGSG20], classification [CCG+21] and more. Solomon et al. [SWB21] presented an approach based on stochastic subsampling applied to 2D images. Carriere et al. [CCG+21] presented an efficient and generic persistence optimization framework, supporting a wide range of criteria and applications, exploiting the convergence properties of stochastic sub-gradient descent [KB15] for tame functions [DDKL20]. Nigmetov et al. [NM22] presented an alternative method, drastically reducing the number of optimization iterations, but at the cost of significantly more computationally expensive steps. As described in Sec. 3.2, one can leverage these frameworks for the problem of topological simplification, however, with impractical runtimes for three-dimensional datasets of standard size (e.g. up to days of computation). We address this issue in this work by proposing a practical approach for topological simplification optimization, with substantial accelerations over state-of-the-art frameworks for persistence optimization [CCG+21].

3.1.2 Contributions

This chapter makes the following new contributions:

1. **Algorithm:** We introduce a practical solver for the optimization of topological simplification for scalar data (Sec. 3.3). Our algorithm is based on two accelerations, which are tailored to the specific problem of topological simplification:

   - We present a simple and practical procedure for the fast update of the persistence diagram of the data along the optimization (Sec. 3.3.2), hence preventing a full re-computation at each step.
   - We describe a simple and practical procedure for the fast update of the pair assignments between the diagram specified as target, and the persistence diagram of the optimized data (Sec. 3.3.3), also preventing a full re-computation at each step.

   Overall, the combination of these accelerations makes topological simplification optimization tractable for real-life datasets.

2. **Applications:** Thanks to its practical time performance, our work sets up ready-to-use foundations for several concrete applications:

   - *Visualization of topologically simplified data (Sec. 3.4.2):* we illustrate the utility of our framework for the direct visualization and analysis of topologically simplified data. Our approach reduces visual clutter in isosurfaces by simplifying connected components as well as, in contrast to previous work, surface handles. We also investigate prominent filament extraction in 3D data, where we show that our approach helps standard topological techniques for removing filament loops.
   - *Surface genus repair (Sec. 3.4.3):* we show how to use our framework to repair genus defects in surface processing, with an explicit control on the employed primitives (cutting or filling).

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

3.2 Approach

This section describes our overall approach for the optimization of the topological simplification of scalar data.

Given the diagram $\mathcal{D}(f)$ of the input field f, we call a *signal* pair a persistence pair of $\mathcal{D}(f)$ which is selected by the user for preservation. Symmetrically, we call a *non-signal* pair a persistence pair of $\mathcal{D}(f)$ which is selected by the user for cancellation. Note that this distinction between *signal* and *non-signal* pairs is application dependent. In practice, the user can be aided by several criteria, such as persistence [ELZ02], geometrical measures [CSvdP04], etc. Then, topological simplification can be expressed as an optimization problem, with the following objectives:

1. Penalizing the persistence of the *non-signal* pairs;
2. Enforcing the precise preservation of the *signal* pairs.

In short, we wish to penalize the undesired features (objective *1*), and, at the same time, enforce the precise preservation of the features of the input which are deemed relevant (objective *2*). The latter objective is important in practice to preserve the accuracy of the features of interest. As later discussed in Sec. 3.3.3 (and illustrated in Fig. 3.2), *non-signal* and *signal* pairs do interact during the optimization, thereby perturbing *signal*

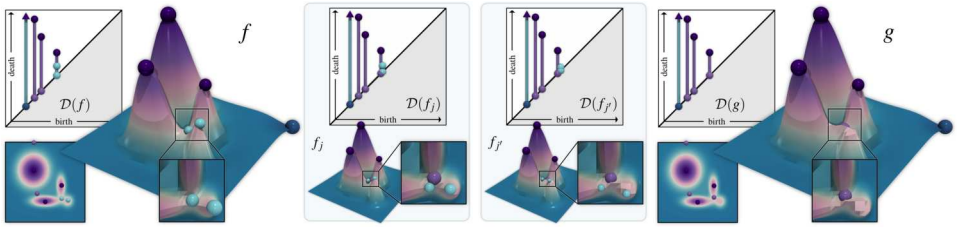

Figure 3.2 – *Interactions between* non-signal *and* signal *pairs during the optimization. A multi-saddle vertex can be involved in both a* non-signal *pair* p *(cyan bar in* $\mathcal{D}(f)$*) and a* signal *pair* p' *(vertically aligned purple bar in* $\mathcal{D}(f)$*). At iteration* j*, the update of the* non-signal *pair* p *unfolds the multi-saddle into multiple simple saddles of distinct values, effectively* perturbing *the birth of the* signal *pair* p' *and making it* non-still*. In real-life data, especially in 3D, such configurations occur often, and cascade. In our experiments (Sec. 3.4), at each iteration,* 11% *of the* signal *pairs are perturbed this way by* non-signal *pairs (on average, and up to* 32%*). This is addressed by our loss (Sec. 3.2) which enforces* signal *pair preservation.*

pairs. In our experiments (Sec. 3.4), at each optimization iteration, 11% of the *signal* pairs are perturbed by *non-signal* pairs (on average, and up to 32%). In certain configurations, this can drastically alter the persistence of *signal* pairs. Hence, to address this issue, the precise preservation of the *signal* pairs should be explicitly constrained.

In the following, we formalize this specific optimization problem based on the generic framework described in Sec. 2.4. Our novel solver (for efficiently solving it) is presented in Sec. 3.3.

Let $\mathcal{D}_T$ be the *target diagram*. It can be obtained by copying the diagram $\mathcal{D}(f)$ of the input field f, and by removing the *non-signal* pairs. $\mathcal{D}_T$ encodes the two objectives of our problem: it describes the constraints for the cancellation of the noisy features of f (objective *1*) as well as the lock constraints for its features of interest (objective *2*).

In general, a perfect *reconstruction* (i.e., a scalar field g such that $\mathcal{D}(g) = \mathcal{D}_T$) may not exist in 3D (deciding on its existence is NP-hard [ABD+13]). Thus, a practical strategy consists in optimizing the scalar field f such that its diagram $\mathcal{D}(f)$ gets *as close as possible* to $\mathcal{D}_T$. For this, we consider the following *simplification energy*:

$$\mathcal{E}\big(\mathcal{D}(f)\big) = \mathcal{W}_2\big(\mathcal{D}(f), \mathcal{D}_T\big)^2. \tag{3.1}$$

Since the Wasserstein distance is locally Lipschitz and a definable function of persistence [CCG+21], the optimization framework of Sec. 2.4 can be used to optimize $\mathcal{L} = \mathcal{E} \circ \mathcal{P} \circ \mathcal{F}$ with guaranteed convergence.

Specifically, at each iteration, given the optimal assignment ϕ^* induced by the Wasserstein distance between $\mathcal{D}(f)$ and $\mathcal{D}_T$ (Sec. 2.3), one can dis-

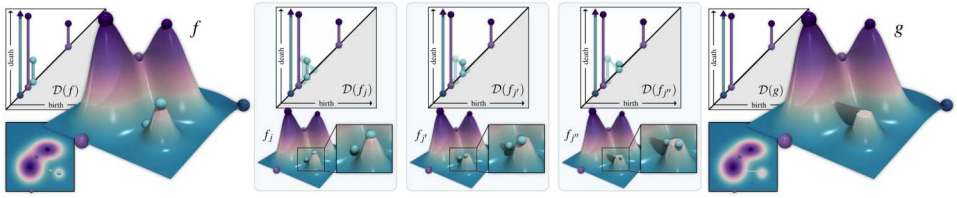

Figure 3.3 – *Optimizing the simplification of an input scalar field $f = f_0$ into a field $g = f_{final}$ for the removal of a user selected saddle-maximum pair (cyan). At each iteration ($j < j' < j''$), given the point $p_i \in \mathcal{D}(f_j)$ to cancel, the data values of its birth and death vertices v_{i_b} and v_{i_d} (cyan spheres in the data) are modified to project p_i to the diagonal. In this example, this results in a scalar field g which is close to f, with the prescribed topology.*

Algorithm 1 Baseline optimization approach for topological simplification.

Input: Input scalar field $f = f_0 : \mathcal{K} \to \mathbb{R}$.
Input: Target diagram $\mathcal{D}_T$.
Input: Stopping conditions $s \in [0,1]$, $j_{max} \in \mathbb{N}$.
Output: Topologically simplified scalar field $g = f_{final} : \mathcal{K} \to \mathbb{R}$.

$j \leftarrow 0$
$\mathcal{D}(f_j) \leftarrow PersistenceDiagramComputation(v_{f_j})$
$(\mathcal{L}(v_{f_j}), \phi_j^*) \leftarrow WassersteinDistanceComputation(\mathcal{D}(f_j), \mathcal{D}_T)$
do
    $j \leftarrow j + 1$
    $v_{f_j} \leftarrow GradientDescentStep(\phi_{j-1}^*, v_{f_{j-1}})$
    $\mathcal{D}(f_j) \leftarrow PersistenceDiagramComputation(v_{f_j})$
    $(\mathcal{L}(v_{f_j}), \phi_j^*) \leftarrow WassersteinDistanceComputation(\mathcal{D}(f_j), \mathcal{D}_T)$
while $\mathcal{L}(v_{f_j}) > s\mathcal{L}(v_{f_0})$ and $j < j_{max}$

place each point p_i in $\mathcal{D}(f)$ towards its individual target $\phi^*(p_i)$ by adjusting accordingly the corresponding scalar values $v_f(v_{i_b})$ and $v_f(v_{i_d})$ (Eq. 2.2). In practice, the generic optimization framework reviewed in Sec. 2.4 computes this displacement (given ϕ^*) via automatic differentiation [CCG+21] and by using Adam [KB15] for gradient descent.

However, depending on the employed step size, a step of gradient descent on v_f may change the initial filtration order (Sec. 2.2). Thus, after a step of gradient descent, the persistence diagram of the optimized data needs to be recomputed and, thus, so does its optimal assignment ϕ^* to the target $\mathcal{D}_T$. This procedure is then iterated, until the loss at the current iteration is lower than a user-specified fraction s of the loss at the first iteration (or until a maximum number j_{max} of iterations). We call this overall procedure the *baseline optimization for topological simplification*. It is summarized in Alg. 1 and illustrated in Fig. 3.3.

As shown in Alg. 1, each iteration j of the optimization involves a

step of gradient descent, the computation of the diagram $\mathcal{D}(f_j)$ and the computation of its Wasserstein distance to $\mathcal{D}_T$. While the first of these three steps has linear time complexity, the other two steps are notoriously computationally expensive and both have cubic theoretical worst case time complexity, $\mathcal{O}(n_\sigma^3)$. In practice, practical implementations for persistence diagram computation tend to exhibit a quadratic behavior [BKRW17, BKR14, GVT23]. Moreover, the exact optimal assignment algorithm [Mun57] can be approximated in practice to improve runtimes, for instance with Auction-based [BC91, KMN17, VBT20] or sliced approximations [CCO17].

However, even when using the above practical implementations for persistence computation and assignment optimization, the baseline optimization approach for topological simplification has impractical runtimes for datasets of standard size. Specifically, for the simplifications considered in our experiments (Sec. 3.4), this approach can require up to days of computations per dataset. When it completes within 24 hours, the computation spends 20% of the time in persistence computation and 75% in assignment optimization.

3.3 Algorithm

This section describes our algorithm for topological simplification optimization. It is based on a number of practical accelerations of the baseline optimization (Alg. 1), which are particularly relevant for the problem of topological simplification.

3.3.1 Direct gradient descent

Instead of relying on automatic differentiation and the Adam optimizer [KB15] as done in the generic framework reviewed in Sec. 2.4, similar to [PSO18], we can derive the analytic expression of the gradient of our energy on a per-iteration basis (Eq. 3.1) and perform at each iteration a direct step of gradient descent, in order to improve performance. Specifically, at the iteration j (Alg. 1), given the current persistence diagram $\mathcal{D}(f_j)$, if the assignments between diagonal points are ignored (these have zero cost, Sec. 2.3), Eq. 3.1 can be re-written as:

$$\mathcal{E}\big(\mathcal{D}(f_j)\big) = \min_{\phi \in \Phi} \sum_{p_i \in \mathcal{D}(f_j)} ||p_i - \phi(p_i)||_2^2.$$

As the optimal assignment ϕ_j^* (i.e., minimizing the energy for a fixed $\mathcal{D}(f_j)$) is constant at the iteration j, the energy can be re-written as:

$$\mathcal{E}\big(\mathcal{D}(f_j)\big) = \sum_{p_i \in \mathcal{D}(f_j)} \big(p_{i_b} - \phi_j^*(p_i)_b\big)^2 + \big(p_{i_d} - \phi_j^*(p_i)_d\big)^2.$$

Then, given Eq. 2.2, for the iteration j, the overall optimization loss $\mathcal{L}(v_{f_j})$ can be expressed as a function of the input data vector v_{f_j}:

$$\mathcal{L}(v_{f_j}) = \sum_{p_i \in \mathcal{D}(f_j)} \big(v_{f_j}(v_{i_b}) - \phi_j^*(p_i)_b\big)^2 + \big(v_{f_j}(v_{i_d}) - \phi_j^*(p_i)_d\big)^2.$$

For the iteration j, for a fixed assignment ϕ_j^*, this energy is convex with v_{f_j} (in addition to being locally Lipschitz) and gradient descent can be considered. Specifically, let $\nabla v_{i_b} \in \mathbb{R}^{n_v}$ be a vector with zero entries, except for the ${i_b}^{th}$ entry, which is set to 1. Let $\nabla v_{i_d} \in \mathbb{R}^{n_v}$ be the vector constructed similarly for v_{i_d}. Then, by the chain rule, we have:

$$\begin{aligned}\nabla\mathcal{L}(v_{f_j}) \;=\; & \sum_{p_i \in \mathcal{D}(f_j)} \Big(2\big(v_{f_j}(v_{i_b}) - \phi_j^*(p_i)_b\big)\nabla v_{i_b} \\ & + \; 2\big(v_{f_j}(v_{i_d}) - \phi_j^*(p_i)_d\big)\nabla v_{i_d}\Big).\end{aligned}$$

We now observe that the gradient can be split into two terms, a *birth gradient* (noted $\nabla\mathcal{L}(v_{f_j})_b$) and a *death gradient* (noted $\nabla\mathcal{L}(v_{f_j})_d$):

$$\begin{aligned}\nabla\mathcal{L}(v_{f_j})_b &= \textstyle\sum_{p_i \in \mathcal{D}(f_j)} 2\big(v_{f_j}(v_{i_b}) - \phi_j^*(p_i)_b\big)\nabla v_{i_b} \\ \nabla\mathcal{L}(v_{f_j})_d &= \textstyle\sum_{p_i \in \mathcal{D}(f_j)} 2\big(v_{f_j}(v_{i_d}) - \phi_j^*(p_i)_d\big)\nabla v_{i_d}.\end{aligned}$$

Then, given the above gradient expressions, a step of gradient descent is obtained by:

$$v_{f_{j+1}} = v_{f_j} - \big((\alpha_b \nabla\mathcal{L}(v_{f_j})_b + \alpha_d \nabla\mathcal{L}(v_{f_j})_d\big),$$

where $\alpha_b, \alpha_d \in \mathbb{R}$ are the gradient step sizes for the birth and death gradients respectively. Such individual step sizes enable an explicit control over the evolution of the persistence pairs to cancel (see Sec. 3.4.3).

3.3.2 Fast persistence update

As described in Sec. 3.2, each optimization iteration j involves the computation of the persistence diagram of the data vector v_{f_j}, which is computationally expensive (20% of the computation time on average). Subsequently, for each persistence pair p_i, the data values of its vertices v_{i_b} and v_{i_d} will be updated given the optimal assignment ϕ_j^*.

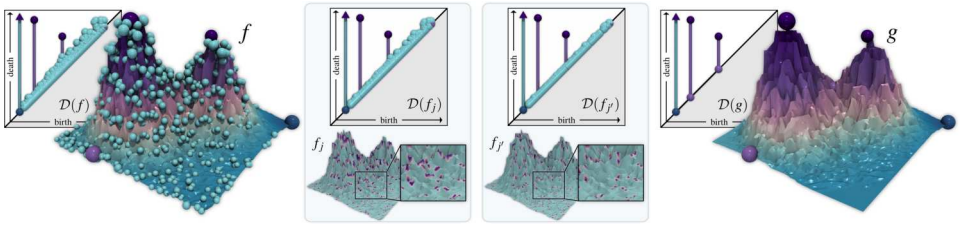

Figure 3.4 – *Updated vertices (dark purple vertices, center insets) along the topological simplification optimization of a noisy terrain (*non-signal *pairs, to simplify, are shown in cyan). In this example, only 20% of the vertices are updated per iteration on average. The discrete gradient (at the core of a recent, fast persistence computation algorithm [GVT23]) only needs to be recomputed for these, yielding a x2 speedup for persistence computation.*

A key observation can be leveraged to improve the performance of the persistence computation stage. Specifically, the updated data vector $v_{f_{j+1}}$ only contains updated data values for the subset of the vertices of $\mathcal{K}$ which are the birth and death vertices v_{i_b} and v_{i_d} of a persistence pair p_i. Then, only a small fraction of the vertices are updated from one iteration to the next, as shown in Fig. 3.4. In practice, for the simplifications considered in our experiments (Sec. 3.4), 90% of the vertices of $\mathcal{K}$ do *not* change their data values between consecutive iterations (on average over our datasets, with the baseline optimization). This indicates that a procedure capable of quickly updating the persistence diagram $\mathcal{D}(f_{j+1})$ based on $\mathcal{D}(f_j)$ has the potential to improve performance in practice.

Several approaches focus on updating a persistence diagram based on a previous estimation [CEM06, LN24], with a time complexity that is linear for each vertex order transposition between the two scalar fields. However, this number of transpositions can be extremely large in practice.

Instead, we derive a simple procedure based on recent work for computing persistent homology with Discrete Morse Theory (DMT) [For98, RWS11, GVT23], which we briefly review here for completeness. Specifically, the *Discrete Morse Sandwich* (DMS) approach [GVT23] revisits the seminal algorithm *PairSimplices* [ELZ02] within the DMT setting, with specific accelerations for volume datasets. This algorithm is based on two main steps. First, a *discrete gradient field* is computed, for the fast identification of zero-persistence pairs. Second, the remaining persistence pairs are computed by restricting the algorithm *PairSimplices* to the critical simplices (with specific accelerations for the persistent homology groups of dimension 0 and $d-1$).

The first key practical insight about this algorithm is that its first step, discrete gradient computation, is documented to represent in practice, in

3D, 66% of the persistence computation time on average [GVT23] (in sequential mode). This indicates that, if one could quickly update the discrete gradient between consecutive iterations, the overall persistence computation step could be accelerated by up to a factor of 3 in practice.

The second key practical insight about this algorithm is that the discrete gradient computation is a completely *local* operation, specifically to the lower star of each vertex v [RWS11] (i.e., the co-faces of v containing no higher vertex than v in the global vertex order).

Thus, we leverage the above two observations to expedite the computation of the diagram $\mathcal{D}(f_j)$, based on the diagram $\mathcal{D}(f_{j-1})$. Specifically, we mark as *updated* all the vertices of $\mathcal{K}$ for which the data value is updated by gradient descent at iteration $j-1$ (Sec. 3.3.1). Then, the discrete gradient field at step j is copied from that at step $j-1$ and the local discrete gradient computation procedure [RWS11] is only re-executed for these vertices for which the lower star may have changed from step $j-1$ to step j, i.e. the vertices marked as *updated* or which contain *updated* vertices in their star. This localized update guarantees the computation of the correct discrete gradient field at step j, with a very small number of local re-computations. Next, the second step of the DMS algorithm [GVT23] (i.e., the computation of the persistence pairs from the critical simplices) is re-executed as-is.

3.3.3 Fast assignment update

As described in Sec. 3.2, each optimization iteration j involves the computation of the optimal assignment ϕ_j^* from the current diagram $\mathcal{D}(f_j)$ to the target $\mathcal{D}_T$, which is computationally expensive.

However, for the problem of simplification, a key practical observation can be leveraged to accelerate this assignment computation. In practice, an important fraction of the pairs of $\mathcal{D}(f_j)$ to optimize (among *signal* and *non-signal* pairs) may only move slightly in the domain from one iteration to the next (as illustrated in Fig. 3.2), and some do not move at all. For these pairs which do not move at step j, the assignment can be re-used from the step $j-1$, hence reducing the size of the assignment problem (Sec. 2.3), and hence reducing its practical runtime.

Given two persistence diagrams $\mathcal{D}(f_j)$ and $\mathcal{D}(f_{j-1})$, we call a *still* persistence pair a pair of points (p_i, p'_i) with $p_i \in \mathcal{D}(f_j)$ and $p'_i \in \mathcal{D}(f_{j-1})$ such that $v_{i_b} = v_{i'_b}$ and $v_{i_d} = v_{i'_d}$. In other words, a *still* persistence pair is a pair which does not change its birth and death vertices from one optimization iteration to the next. In practice, for the simplifications considered in our

experiments (Sec. 3.4), 84% of the persistence pairs of $\mathcal{D}(f_j)$ are still (on average over the iterations and our test datasets, Sec. 3.4). This indicates that a substantial speedup could be obtained by expediting the assignment computation for still pairs.

Let $\mathcal{S}$ be the set of still pairs between the iteration j and $j-1$. Then, for each pair $(p_i, p'_i) \in \mathcal{S}$, we set $\phi^*_j(p_i) \leftarrow \phi^*_{j-1}(p'_i)$. Concretely, we re-use at step j the assignment at step $j-1$ for all the still pairs.

Next, let $\overline{\mathcal{D}(f_j)}$ be the *reduced* diagram at step j, i.e., the subset of $\mathcal{D}(f_j)$ which does not contain still pairs: $\overline{\mathcal{D}(f_j)} = \mathcal{D}(f_j) - \{p_i \in \mathcal{D}(f_j), (p_i, p'_i) \in \mathcal{S}\}$. Similarly, let $\overline{\mathcal{D}_{Tj}}$ be the *reduced* target at step j, i.e., the subset of $\mathcal{D}_T$ which has not been assigned to still pairs: $\overline{\mathcal{D}_{Tj}} = \mathcal{D}_T - \{p''_i \in \mathcal{D}_T, p''_i = \phi^*_j(p_i), (p_i, p'_i) \in \mathcal{S}\}$. Then, we finally complete the assignment between $\mathcal{D}(f_j)$ and $\mathcal{D}_T$ by computing the Wasserstein distance between $\overline{\mathcal{D}(f_j)}$ and $\overline{\mathcal{D}_{Tj}}$, as documented in Sec. 2.3.

Note that, in the special case where the reduced target $\overline{\mathcal{D}_{Tj}}$ is empty (i.e., all *signal* pairs are *still*), the reduced diagram $\overline{\mathcal{D}(f_j)}$ only contains *non-signal* pairs. Then, the optimal assignment can be readily obtained (without any assignment optimization) by simply assigning each point p_i in $\overline{\mathcal{D}(f_j)}$ to its diagonal projection $\Delta(p_i)$. However, from our experience, such a perfect scenario never occurs on real-life data, at the notable exception of the very first iteration (before the data values are actually modified by the solver). For the following iterations, many *signal* pairs are not still in practice. Fig. 3.2 illustrates this with a simple 2D example involving a multi-saddle vertex. However, in real-life data, such configurations occur very often, and cascade. Also, these configurations get significantly more challenging in 3D. For instance, the birth and death vertices of a given *signal* pair can both be multi-saddles, themselves possibly involved with *non-signal* pairs to update (hence yielding perturbations in the *signal* pair). In certain configurations, this can drastically alter the persistence of the *signal* pairs affected by such perturbations. This is addressed by our loss (Sec. 3.2) which enforces the preservation of the *signal* pairs via assignment optimization.

3.4 Results

This section presents experimental results obtained on a computer with two Xeon CPUs (3.0 GHz, 2x8 cores, 64GB of RAM). We implemented our algorithm (Sec. 3.3) in C++ (with OpenMP) as a module for TTK [TFL+17, BMBF+19]. We implemented the baseline optimization ap-

proach (Sec. 3.2) by porting the original implementation by Carriere et al. [CCG+21] from TensorFlow/Gudhi [AAB+15, MBGY14] to PyTorch/TTK [PGM+19] and by applying it to the loss described in Sec. 3.2. We chose this approach as a baseline, since its implementation is simple and publicly available, and since it provides performances comparable to alternatives [NM22]. In our implementations, we use the DMS algorithm [GVT23] for persistence computation (as it is reported to provide the best practical performance for scalar data) and the Auction algorithm [BC91, KMN17] for the core assignment optimization, with a relative precision of 0.01, as recommended in the literature [KMN17]. Persistence computation with DMS [GVT23] is the only step of our approach which leverages parallelism (see [GVT23] for a detailed performance analysis). Experiments were performed on a selection of 10 (simulated and acquired) 2D and 3D datasets extracted from public repositories [TTK20, Kla20], with an emphasis on 3D datasets containing large filament structures (and thus possibly, many persistent saddle pairs). The 3D datasets were resampled to a common resolution (256^3), to better observe runtime variations based on the input topological complexity. Moreover, for each dataset, the data values were normalized to the interval $[0, 1]$, to facilitate parameter tuning across distinct datasets.

Our algorithm is subject to two meta-parameters: the gradient step sizes α_b and α_d. To adjust them, we selected as default values the ones which minimized the runtime for our test dataset with the largest diagram. This resulted in $\alpha_b = \alpha_d = 0.5$ (which coincides, given a persistence pair to cancel, to a displacement of its birth and death vertices halfway towards the other, in terms of function range). For the baseline optimization approach (Sec. 3.2), we set the initial learning rate of Adam [KB15] to the largest value which still enabled practical convergence for all our datasets (specifically, 10^{-4}). For both approaches, we set the maximum number of iterations j_{max} to $1,000$ (however, it has never been reached in our performance experiments).

3.4.1 Quantitative performance

The time complexity of each iteration of the baseline optimization is cubic in the worst case, but quadratic in practice (Sec. 3.2). As discussed in Sec. 3.3, our approach has the same worst case complexity, but behaves more efficiently in practice thanks to our accelerations.

Tab. 3.1 provides an overall comparison between the baseline optimiza-

Table 3.1 – *Time performance comparison between the baseline optimization approach (Sec. 3.2) and our solver (Sec. 3.3), for two simplification scenarios: a mild simplification and an aggressive simplification, where the* non-signal *pairs are the input pairs less persistent than* 1% *(white lines) and* 45% *(grey lines) of the function range, respectively. The column N.S.S.P. reports the average percentage of non-still* signal *pairs (per iteration) for our solver. The stopping condition is set to* $s = 0.01$.

Dataset	d	$\lvert\mathcal{D}(f)\rvert$	$\lvert\mathcal{D}_T\rvert$	Baseline (Sec. 3.2)			Our solver (Sec. 3.3)				
				#It.	Time/It (s.)	Time (s.)	N.S.S.P.	#It.	Time/It (s.)	Time (s.)	Speedup
Cells	2	7,676	2,635	89	0.58	52	0.07%	10	0.20	2	**26**
Ocean Vortices	2	12,069	2,781	87	0.61	53	8.02%	12	0.25	3	**18**
Aneurysm	3	38,490	24,725	80	29.89	2,391	3.70%	8	11.63	93	**26**
Bonsai	3	168,489	55,464	67	56.73	3,801	3.90%	10	13.50	135	**28**
Foot	3	754,965	474,271	60	914.47	54,868	11.28%	4	104.25	417	**132**
Neocortical Layer Axon	3	765,406	483,791	89	735.36	65,447	32.04%	8	263.38	2,107	**31**
Dark Sky	3	1,140,653	774,793	NA	NA	**> 24h**	9.08%	6	122.00	732	**> 118**
Backpack	3	1,331,362	84,402	66	305.58	20,168	21.46%	9	62.22	560	**36**
Head Aneurysm	3	1,345,168	234,672	NA	NA	**> 24h**	6.68%	5	83.80	419	**> 206**
Chameleon	3	3,641,961	32,578	55	210.51	11,578	18.22%	8	74.75	598	**19**
Cells	2	7,676	21	755	0.26	193	0.00%	167	0.19	32	**6**
Ocean Vortices	2	12,069	80	474	0.26	126	0.70%	269	0.14	38	**8**
Aneurysm	3	38,490	231	329	32.65	10,743	1.38%	17	8.73	148	**72**
Bonsai	3	168,489	2	483	72.47	35,002	49.23%	65	12.26	797	**44**
Foot	3	754,965	18	108	36.52	3,944	5.05%	11	29.22	321	**12**
Neocortical Layer Axon	3	765,406	174	177	12.54	2,219	1.87%	8	15.08	121	**18**
Dark Sky	3	1,140,653	120,332	NA	NA	**> 24h**	0.38%	7	30.73	215	**> 402**
Backpack	3	1,331,362	97	329	40.36	13,278	2.21%	28	23.05	646	**21**
Head Aneurysm	3	1,345,168	2	97	119.90	11,630	5.26%	19	35.99	684	**17**
Chameleon	3	3,641,961	2	NA	NA	**> 24h**	0.00%	81	28.70	2,325	**> 37**

Table 3.2 – *Quality comparison between the baseline optimization approach (Sec. 3.2) and our solver (Sec. 3.3) for the parameters used in Tab. 3.1.*

Dataset	d	Baseline (Sec. 3.2)			Our solver (Sec. 3.3)		
		$\mathcal{L}(v_g)$	$\lvert\lvert f-g\rvert\rvert_2$	$\lvert\lvert f-g\rvert\rvert_\infty$	$\mathcal{L}(v_g)$	$\lvert\lvert f-g\rvert\rvert_2$	$\lvert\lvert f-g\rvert\rvert_\infty$
Cells	2	0.0003	0.2939	0.0054	0.0003	0.2996	0.0070
Ocean Vortices	2	0.0006	0.3710	0.0055	0.0005	0.3769	0.0074
Aneurysm	3	0.0007	0.3481	0.0063	0.0006	0.3174	0.0117
Bonsai	3	0.0064	1.0243	0.0060	0.0053	1.0541	0.0117
Foot	3	0.0326	2.0526	0.0058	0.0297	2.0483	0.0127
Neocortical Layer Axon	3	0.0271	2.0876	0.0085	0.0279	2.1435	0.0154
Dark Sky	3	NA	NA	NA	0.0166	1.8617	0.0148
Backpack	3	0.0339	2.4438	0.0070	0.0312	2.1991	0.0159
Head Aneurysm	3	NA	NA	NA	0.0750	3.7571	0.0130
Chameleon	3	0.1679	5.1264	0.0055	0.1736	4.7773	0.0143
Cells	2	0.0627	10.0828	0.1991	0.0628	9.8787	0.2397
Ocean Vortices	2	0.0547	4.8040	0.1370	0.0541	12.7392	0.4404
Aneurysm	3	1.1040	23.2228	0.2586	1.0278	14.1333	0.4205
Bonsai	3	0.6679	34.3607	0.2014	0.6651	19.0419	0.3712
Foot	3	3.3992	25.7129	0.0967	2.9832	23.7273	0.2375
Neocortical Layer Axon	3	5.5482	29.2017	0.1602	4.6222	28.1093	0.3424
Dark Sky	3	NA	NA	NA	87.4867	116.2727	0.5851
Backpack	3	1.0060	23.0536	0.2397	1.1978	13.8104	0.4043
Head Aneurysm	3	0.4880	16.7538	0.0873	0.4912	10.3221	0.2386
Chameleon	3	NA	NA	NA	0.4223	10.8036	0.2949

tion (Sec. 3.2) and our solver (Sec. 3.3). Specifically, it compares both approaches in terms of runtime, for two realistic simplification scenarios: a mild simplification and an aggressive simplification, where the *non-signal* pairs are the input pairs less persistent than 1% (white lines) and 45% (grey lines) of the function range, respectively. For both approaches, we set the stopping criterion s to 0.01, such that both methods reach a similar residual loss at termination (and hence produce results of comparable quality). This table shows that for the mild simplification scenario (white lines), our approach produces results within minutes (at most 35). In contrast, the baseline approach does not produce a result after 24 hours of computation for the largest examples. Otherwise, it still exceeds hours of computation for diagrams of modest size. Overall, our approach results in an average $\times 64$ speedup for this simplification scenario. This acceleration can be explained by several factors. First, the direct gradient descent (Sec. 3.3.1) requires *fewer* iterations than the baseline approach (we discuss this further in the next paragraph, presenting Tab. 3.2). Second, our approach also results in *faster* iterations, given the accelerations presented in Sec. 3.3.

In practice, the overall runtime for our solver is a function of the size of the input and target diagrams (large diagrams will lead to large assignment problems). The size of the topological features in the geometrical domain also plays a role (larger features will require more iterations). Finally, the number of still *signal* pairs also plays a role given our fast assignment update procedure (Sec. 3.3.3, a large number of still *signal* pairs leads to faster assignments). For instance, the total number of pairs (input plus target) for the *Neocortical Layer Axon* dataset is about 20 times larger than that of the *Aneursym* dataset, and the ratio between their respective runtime is also about 20. Moreover, the *Foot* and *Neocortical Layer Axon* datasets have comparable overall sizes (input plus target). However, the latter dataset results in a computation time that is $\times 5$ larger. This can be partly explained by the fact that the topological features are larger in this dataset, yielding twice more iterations (hence explaining a $\times 2$ slowdown). Moreover, the per-iteration runtime is also $\times 2.5$ slower (hence explaining the overall $\times 5$ slowdown), due the higher percentage of non-still *signal* pairs, which increase the size of the assignment problem.

Similar observations can be made for the aggressive simplification scenario (grey lines). Our solver compute these simplifications within minutes (at most 39), for the same average speedup over the baseline ($\times 64$). Note that, for both the baseline and our solver, while the number of iter-

Table 3.3 – *Individual gains (in percentage of runtime) for each of our accelerations for the topological simplification parameters used in Tab. 3.1.*

Dataset	d	$\lvert\mathcal{D}(f)\rvert$	$\lvert\mathcal{D}_T\rvert$	Persistence Update (Sec. 3.3.2)	Assignment Update (Sec. 3.3.3)
Cell	2	7,676	2,635	5.6	79.2
Ocean Vortices	2	12,069	2,781	−0.4	79.8
Aneurysm	3	38,490	24,725	41.4	24.8
Bonsai	3	168,489	55,464	12.1	80.6
Foot	3	754,965	474,271	0.2	90.9
Neocortical Layer Axon	3	765,406	483,791	0.0	74.6
Dark Sky	3	1,140,653	774,793	3.2	96.7
Backpack	3	1,331,362	84,402	1.1	74.5
Head Aneurysm	3	1,345,168	234,672	1.3	93.4
Chameleon	3	3,641,961	32,578	1.5	61.3
Cell	2	7,676	21	17.9	21.9
Ocean Vortices	2	12,069	80	14.9	41.4
Aneurysm	3	38,490	231	54.9	7.6
Bonsai	3	168,489	2	44.8	1.5
Foot	3	754,965	18	30.0	−21.5
Neocortical Layer Axon	3	765,406	174	0.1	−28.9
Dark Sky	3	1,140,653	120,332	−1.3	91.6
Backpack	3	1,331,362	97	17.2	27.9
Head Aneurysm	3	1,345,168	2	8.8	67.0
Chameleon	3	3,641,961	2	9.0	92.8

ations increases in comparison to the mild simplification (since more and larger features need to be simplified), iterations are significantly faster as the assignment problems are much smaller.

The runtime gains provided by our individual accelerations is presented in Tab. 3.3. For mild simplifications (white lines), our procedure for fast Persistence update (Sec. 3.3.2) can save up to 41.4% of overall computation time, and 6.6% on average. These numbers increase to 54.9% and 19.6% respectively for aggressive simplifications (grey lines). As more iterations are required to simplify persistent features (Tab. 3.1), less and less vertices are updated along the iterations (since low-persistence features are cancelled in the early iterations), hence advantaging our fast persistence update procedure. For mild simplifications (white lines), our procedure for fast assignment update (Sec. 3.3.3) provides the most substantial gains, saving up to 97% of the overall computation time for the largest target diagram, and 76% on average. For aggressive simplifications (grey lines), the average gain decreases to 30% since assignment problems get smaller (and so does their importance in the overall computation). Negative entries in Tab. 3.3 indicate cases where the acceleration actually degrades runtimes. For the fast persistence update, this happens when the number of updated vertices is so large that their identification overweights the gradient computation for the non-updated vertices. Similar remarks can be made for the fast assignment update, where the identification of the still pairs can penalize runtime for small assignment problems. Overall, both our accelerations (fast persistence update, Sec. 3.3.2, and fast assignment update, Sec. 3.3.3) improve performance in both simplification

scenarios, with the fast assignment update being more important for mild simplifications, and the fast persistence update for aggressive ones.

Tab. 3.2 compares the quality of the output obtained with the baseline optimization (Sec. 3.2) and our algorithm (Sec. 3.3), for the simplification parameters used in Tab. 3.1. The quality is estimated based on the value of the loss at termination ($\mathcal{L}(v_g)$), which assesses the quality of the topological simplification. To estimate the proximity of the solution g to the input f, we also evaluate the distances $||f-g||_2$ (giving a global error for the entire dataset) and $||f-g||_\infty$ (giving a pointwise worst case error). We refer the reader to Appendix A for complementary quality statistics. Overall, Tab. 3.2 shows that our approach provides comparable losses to the baseline approach (sometimes marginally better). In terms of data fitting, our approach also provides comparable global distances $||f-g||_2$ (sometimes marginally better). For the pointwise worst case error ($||f-g||_\infty$), our approach can result in degraded values (by a factor 2). This can be explained by the fact that, when tuning the parameters of our approach, we optimized the gradient step size to minimize running time, hence possibly triggering in practice bigger pointwise shifts in data values. In contrast, the baseline approach uses the Adam [KB15] algorithm, which optimizes step sizes along the iterations, possibly triggering milder pointwise shifts in data values. In principle, the $||f-g||_\infty$ distance could be improved for our solver by considering smaller step sizes, but at the expense of more iterations.

We provide complementary quality statistics, where we now evaluate the preservation of the *features of interest* after our simplification. For this, Tab. 3.4 reports statistics (minimum, average, maximum) of the displacement in the birth-death space (between 0 and 1) for the *signal* pairs, both for a mild (white lines) and an aggressive (grey lines) simplification based on persistence (1% and 45% of the function range, respectively). Specifically, displacements are evaluated given the optimal assignment (achieved by the Wasserstein distance) between $\mathcal{D}_T$ and $\mathcal{D}(g)$. Overall, this table shows that the position of the *signal* pairs in the birth-death space is well constrained by our solver, with a worst displacement of 2.25×10^{-02} for a challenging example (aggressive simplification of the *Dark Sky* dataset, where many multi-saddles are involved in both *signal* and *non-signal* pairs). For all datasets, the achieved worst displacement is negligible with regard to the employed persistence threshold (by an order of magnitude).

Table 3.4 – *Statistics (minimum, average, maximum) of displacement in the birth-death space (between* 0 *and* 1*) for the* signal *pairs. The employed simplification parameters are those used in the Table 1 of the main manuscript (white lines: mild simplification, grey lines: aggressive one).*

Dataset	d	Min.	Avg.	Max.
Cells	2	0	0	0
Ocean Vortices	2	0	0	0
Aneurysm	3	0	1.22×10^{-07}	8.27×10^{-04}
Bonsai	3	0	4.03×10^{-07}	3.92×10^{-03}
Foot	3	0	3.28×10^{-09}	4.19×10^{-03}
Neocortical Layer Axon	3	0	1.14×10^{-06}	4.38×10^{-03}
Dark Sky	3	0	1.84×10^{-06}	2.01×10^{-03}
Backpack	3	0	2.85×10^{-06}	1.25×10^{-03}
Head Aneurysm	3	0	6.24×10^{-07}	1.20×10^{-03}
Chameleon	3	0	3.14×10^{-06}	1.43×10^{-03}
Cells	2	0	0	0
Ocean Vortices	2	0	2.99×10^{-07}	2.40×10^{-05}
Aneurysm	3	0	5.51×10^{-05}	1.27×10^{-02}
Bonsai	3	0	9.42×10^{-21}	1.88×10^{-20}
Foot	3	0	5.09×10^{-21}	9.17×10^{-20}
Neocortical Layer Axon	3	0	4.22×10^{-21}	7.35×10^{-19}
Dark Sky	3	0	1.38×10^{-04}	2.25×10^{-02}
Backpack	3	0	8.67×10^{-22}	8.41×10^{-20}
Head Aneurysm	3	0	1.06×10^{-22}	2.12×10^{-22}
Chameleon	3	0	0	0

3.4.2 Analyzing topologically simplified data

Our approach enables the direct visualization and analysis of topologically simplified data. This is illustrated in Fig. 3.1, which shows the processing of an acquired dataset (*"Aneurysm"*) representing a network of arteries. As documented in the literature [MK16, HBMK22], this network exhibits a typical tree-like structure, whose accurate geometrical extraction is relevant for medical analysis. The filament structure of the arteries can be simply extracted by considering the *discrete integral lines* [GVT23] (a.k.a. *v-paths* [For98]) which connect 2-saddles to maxima and which have a minimum function value above 0.1 (scalar fields are normalized). This value 0.1 generates an isosurface (transparent surfaces, Fig. 3.1) which accurately captures the geometry of the blood vessels. Hence, selecting the discrete integral lines above that threshold guarantees the extraction of the filament structures within the vessels.

As shown in Fig. 3.1, the diagram $\mathcal{D}(f)$ contains several saddle pairs, corresponding to persistent 1-dimensional generators [GVT23, Iur21] (curves colored by persistence in the inset zooms), which yields incorrect loops in the filament structure (which is supposed to have a tree-like

structure [MK16, HBMK22]). To remove loops in networks of discrete integral lines, an established topological technique, relying on standard discrete Morse theory [For98], consists in reversing the discrete gradient [GRSW13] along *saddle connectors*. We recap this procedure here for completeness. Given the persistence diagram $\mathcal{D}(f)$, we process its *non-signal* saddle pairs in increasing order of persistence. For each saddle pair (σ_b, σ_d), its *saddle connector* is constructed by following the discrete gradient of f from σ_d down to σ_b. Next, the pair of critical simplices (σ_b, σ_d) is cancelled, in the discrete sense, by simply reversing the discrete gradient along its saddle connector [For98] (i.e., each discrete vector is reversed to point to the preceding co-face). Such a reversal is marked as *valid* if it does not create any cycle in the discrete gradient field. The validity of a reversal is important since invalid reversals result in discrete vector fields which no longer describe valid scalar fields, and from which the subsequent extraction of integral lines can generate further loops (which we precisely aim to remove). The cancellation of a saddle pair (σ_b, σ_d) is *skipped* if the reversal of its saddle connector is not valid, or if its saddle connector does not exist. The latter case occurs for instance for nested saddle pairs, when an invalid reversal of a small persistence pair prevents the subsequent reversal of a larger one. Finally, when all the *non-signal* saddle pairs have been processed, the simplified filament structures are simply obtained from the simplified discrete gradient, by initiating integral lines from 2-saddles up to maxima.

However, in the example of Fig. 3.1, this saddle connector reversal procedure fails at simplifying the spurious loops in the filament structures, while maintaining a valid discrete gradient (Fig. 3.1(b)). As discussed in the literature [GRSW13], integral line reversal is indeed not guaranteed to completely simplify saddle pairs (*v-path* co-location [IFF15] as well as specific cancellation orderings [GNP$^+$05, GDN$^+$07] can challenge reversals, the latter issue being a manifestation of the NP-hardness of the problem [ABD$^+$13]). This is evaluated in the bottom left histogram, which reports the number of *skipped* saddle connector reversals as a function of the persistence of the corresponding pair. Specifically, this histogram shows that the reversal of several high-persistence saddle pairs could not be performed, hence the presence of large loops in the extracted filament structures.

Our approach can be used to efficiently generate a function g which is close to the input f and from which the removal of saddle pairs has been optimized, while maintaining intact the rest of the features (see the result-

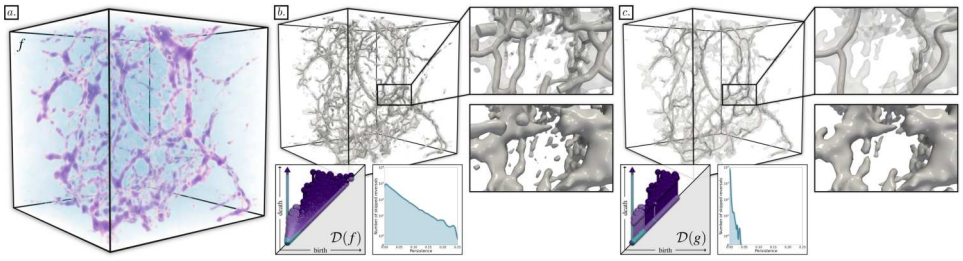

Figure 3.5 – *Topological simplification optimization for a challenging dataset (*"Dark Sky"*: dark matter density in a cosmology simulation,* (a), signal pairs*: pairs with a persistence larger than* 0.25*). The geometry of the* cosmic web *[Sou11, SPN+16] is captured* (b) *by an isosurface (at isovalue* 0.4*) and its core filament structure is extracted by the upward discrete integral lines, started at* 2*-saddles above* 0.4*. The latter structure contains many small-scale loops as many, persistent saddle connector reversals could not be performed (bottom left histogram). The local minimum* g *of the simplification energy (Eq. 3.1) found by our solver* (c) *has a number of* non-signal *pairs reduced by* 92%*. This results in a less cluttered visualization, as the cosmic web has a less complicated topology (noisy connected components are removed and small scale handles are cut, inset zooms). This also induces fewer skips of persistent saddle connector reversals (bottom right histogram), hence simplifying more loops and revealing the main filament structure.*

ing diagram $\mathcal{D}(g)$, Fig. 3.1). Specifically, we set as *non-signal* pairs *all* the saddle pairs of the input, and we set as *signal* pairs all the others (irrespective of their persistence). This enables a direct visualization and analysis of the topologically simplified data, where isosurface handles have been cut (Fig. 3.1c, bottom-right zoom VS Fig. 3.1b, bottom-right zoom) and where most spurious filament loops have been consequently simplified (Fig. 3.1, top zoom). Note that, as shown in the bottom right histogram, our optimization modifies the input data f into a function g where reversal skips still occur. This is due to the fact that our solver identifies a *local* minimum of the simplification energy (Eq. 3.1) and that, consequently, a few saddle pairs, with low persistence, may still remain (we recall that the problem is NP-hard [ABD+13], see Sec. 3.4.4 for further discussions). However, the skipped reversals which remain after our optimization (Fig. 3.1, bottom right histogram) only involve very low persistence pairs, hence allowing the cancellation of the largest loops overall.

Fig. 3.5 illustrates our topological simplification optimization for a challenging dataset (*"Dark Sky"*: dark matter density in a cosmology simulation). As shown in the inset zooms, the isosurface capturing the *cosmic web* [Sou11, SPN+16] has a complicated topology (many noisy connected components and handles), which challenges its visual inspection. Its core filament structure also contains many small-scale loops since many persis-

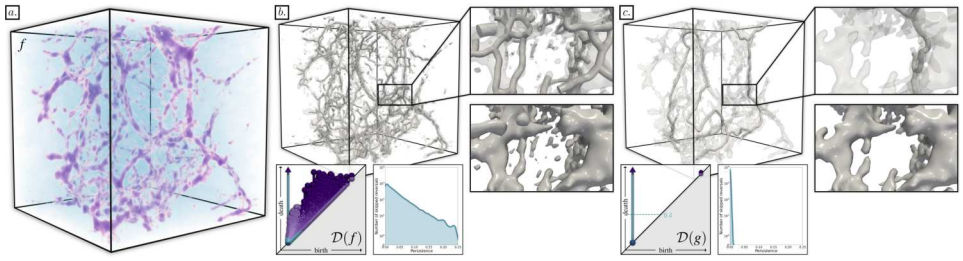

Figure 3.6 – Extreme *simplification optimization for a challenging dataset (*"Dark Sky"*: dark matter density in a cosmology simulation,* (a)*,* only one signal pair*: the bar involving the global minimum). The geometry of the* cosmic web *[Sou11, SPN+16] is captured* (b) *by an isosurface (at isovalue* 0.4*) and its core filament structure is extracted by the upward discrete integral lines, started at* 2*-saddles above* 0.4*. The latter structure contains many small-scale loops as many, persistent saddle connector reversals could not be performed (bottom left histogram). The local minimum* g *of the simplification energy found by our solver* (c) *has a number of* non-signal *pairs reduced by* 95%*. This results in a less cluttered visualization, as the cosmic web has a drastically simplified topology (noisy connected components are removed and most handles are cut, inset zooms). This also induces fewer skips of persistent saddle connector reversals (bottom right histogram), here simplifying all filament loops and revealing the core filament structure.*

tent saddle connector reversals could not be performed (Fig. 3.5, bottom left histogram). Our solver provides a local minimum g to the simplification energy (Eq. 3.1) with a number of *non-signal* pairs reduced by 92%. This results in a less cluttered visualization, as the resulting cosmic web (Fig. 3.5(c)) has a less complicated topology (noisy connected components are removed and small scale handles are cut, inset zooms). Moreover, our optimization modifies the data in a way that is more conducive to persistent saddle connector reversals (bottom right histogram), hence simplifying more loops and, thus, better revealing overall the large-scale filament structure of the cosmic web.

We consider the challenging *Dark Sky* dataset (large input diagrams, many saddle pairs, intricate geometry) and specify an *extreme* simplification. In particular, all the *finite* persistence pairs are considered as *non-signal* (bars marked with spheres at their extremities, Fig. 3.6) and only the *infinite* bar involving the global minimum (cropped by convention at the globally maximum data value, bar with an upward arrow, Fig. 3.6) is considered as a *signal pair*. The corresponding results are shown in Fig. 3.6. Specifically, this figure shows that, despite this challenging dataset and extreme simplification criterion, our approach still manages to simplify 95% of the *non-signal* pairs, which is a slight improvement over the original experiment reported in the Figure Fig. 3.5 of the main manuscript (92%

for a persistence threshold of 0.25). Moreover, from a qualitative point of view, all the *filament* loops have been simplified: the persistence diagram does not contain any *finite* persistence pairs whose life-span crosses the death isovalue 0.4 (dashed horizontal line). Only the *infinite* bar related to the global minimum (bar with an upward arrow) crosses it. In other words, this means that the cosmic web volume (i.e. the sublevel set for the isovalue 0.4) is made of only one connected component and contains no topological handles.

3.4.3 Repairing genus defects in surface processing

Our work can also be used to repair genus defects in surface processing, where surface models, in particular when they are acquired, can include spurious handles due to acquisition artifacts. While several approaches have been proposed to address this issue [CJL+18, ZCLJ20, ZCLJ22], they typically rely on intensive automatic optimizations, aiming at selecting at each iteration the *best* local simplification primitive (i.e. *cutting* or *filling*). In contrast, our approach relies on a simpler and lightweight procedure, which provides control to the user over the primitives to use. For this, we consider the three-dimensional signed distance field f to the input surface S, computed on a regular grid (i.e., f encodes for each grid vertex v the distance to the closest point on the surface S, multiplied by -1 if v is located within the volume enclosed by S). For such a field, the zero level set $f^{-1}(0)$ coincides with S. Then, the removal of a handle in S can be performed by creating a simplified signed distance field g, where the corresponding saddle pair has been canceled. Finally, the zero level set $g^{-1}(0)$ provides the simplified surface S'.

This process is illustrated in Fig. 3.7 where the handle of a torus is removed. Note that, from a topological point of view, this operation can be performed in two ways: either by cutting the handle (Fig. 3.7(b)), or by filling it (Fig. 3.7(c)). This can be controlled in our solver by simply adjusting the step sizes for the birth and death gradients (Sec. 3.3.1). Specifically, given a saddle pair to remove $p_i \in \mathcal{D}(f)$, handle cutting is obtained by setting α_d to zero. Then, the death vertex v_{i_d} will not be modified (above the zero level set), while only the birth vertex v_{i_b} (located in the star of the 1-saddle creating the handle) will increase its value above 0, effectively disconnecting the handle in the output surface S'. Handle filling is obtained symmetrically, by setting α_b to zero (effectively forcing the 2-saddle to decrease its value below 0).

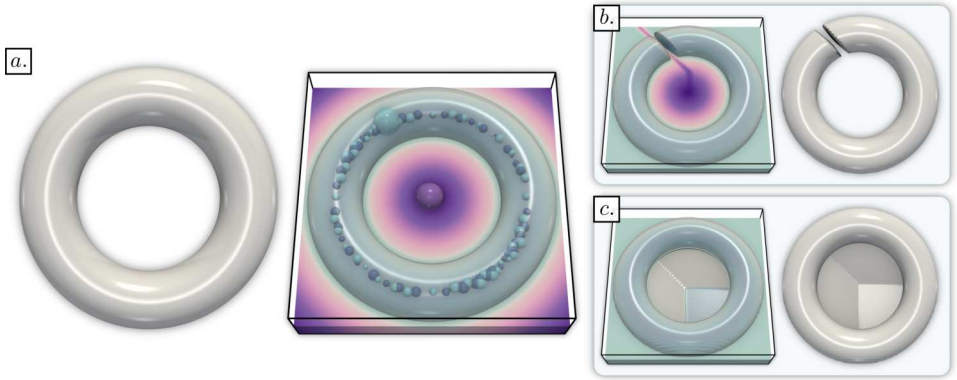

Figure 3.7 – *Handle removal on a torus example.* (a) *The input surface* S *(left) is used to compute a 3D signed distance field* f *(right, color map).* f *contains a persistent saddle pair (large spheres) encoding the handle of the torus and many low-persistence minimum-saddle pairs (smaller spheres, radius scaled by persistence) which are artifacts (located on the medial axis of* S*) of the sampling of the distance field (which has discontinuous derivatives). The handle can be removed in the output surface* S' (b, c) *by considering the zero level set of a simplified field* g *obtained with our approach (in this example, only the persistent generator of* f *with infinite persistence has been maintained). The handle can be removed either by* cutting (b) *(by only using the* birth gradient*, Eq. 3.2) or by* filling (c) *(by only using the* death gradient*, Eq. 3.2).*

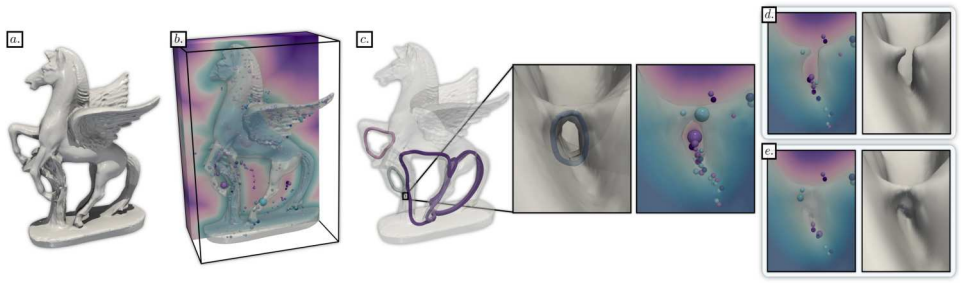

Figure 3.8 – *Removal of a spurious handle from an acquired surface* S (a)*. First, the signed distance field* f *is computed from* S (b)*.* f *is shown with a color map on the clipped volume, with its critical simplices colored per dimension, with a sphere with a radius proportional to their persistence. The extraction of the* 1*-dimensional persistent generators [GVT23, Iur21]* $((c)$*, colored by persistence) reveals the existence of a short generator in* f*, corresponding to a small handle defect in* S *(under the Pegasus front left hoof, see inset zooms). Our framework can repair this defect by simplifying the corresponding saddle pair, either by* cutting $((d)$*, by only using the* birth *gradient, Eq. 3.2) or by* filling $((e)$*, by only using the* death *gradient, Eq. 3.2).*

Fig. 3.8 presents a realistic example of an acquired surface from a public repository [TTK20], which contains a spurious handle, due to acquisition artifacts. First, the signed distance field is computed and its 1-dimensional persistent generators [Iur21, GVT23] are extracted. The shortest generator corresponds to a small handle, which happens to be a genus defect in this example. Then, the user can choose to repair this defect via cutting or filling, resulting in a repaired surface S' which is close to the input S, and from which the spurious handle has been removed.

3.4.4 Limitations

Our approach is essentially numerical and, thus, suffers from the same limitations as previous numerical methods for topological simplification (Sec. 3.1.1). Specifically, the *non-signal* pairs are canceled by our approach by decreasing their persistence to a target value of zero. However, this decrease is ultimately limited by the employed numerical precision (typically, 10^{-6} for single-precision floating point values). From a strictly combinatorial point of view, this can result in *residual pairs* with an arbitrarily small persistence (i.e., in the order of the numerical precision). In principle, this drawback is common to all numerical methods (although sometimes mitigated via smoothing). Then, when computing topological abstractions, these residual pairs need to be removed from the computed abstraction (e.g., with integral line reversal, Sec. 3.4.2). However, as discussed in the literature [GBHP09, GRSW13], post-process mechanisms for simplifying topological abstractions may not guarantee a complete simplification of the abstractions either (this is another concrete implication of the NP-hardness of the problem [ABD$^+$13]). However, our experiments (Sec. 3.4.2) showed that our numerical optimization precisely helped such combinatorial mechanisms, by pre-processing the data in a way that resulted eventually in fewer persistent reversal skips (Figs. 3.1 and 3.5, right versus left histograms).

Similar to previous persistence optimization frameworks, our approach generates a *local* minimum of the simplification energy (Eq. 3.1), and thus it is not guaranteed to reach the global minimum. As a reminder, in 3D, a perfect reconstruction (i.e., $\mathcal{D}(g) = \mathcal{D}_T$) may not exist and finding the global minimizer of the simplification energy is NP-hard [ABD$^+$13]. However, our experiments (Sec. 3.4.1) showed that our approach still generated solutions whose quality was on par with the state-of-the-art (comparable losses and distances to the input), while providing

substantial accelerations. Moreover, as shown in Sec. 3.4.2, these solutions enabled the direct visualization of isosurfaces whose topology was indeed simplified (fewer components and handles) and they were also conducive to improved saddle connector reversals.

3.5 Summary

This chapter introduced a practical solver for topological simplification optimization. Our solver is based on tailored accelerations, which are specific to the problem of topological simplification. Our accelerations are simple and easy to implement, but result in significant gains in terms of runtime, with $\times 60$ speedups on average on our datasets over state-of-the-art persistence optimization frameworks (with both fewer and faster iterations), for comparable output qualities. This makes topological simplification optimization practical for real-life three-dimensional datasets. We showed that our contributions enabled a direct visualization and analysis of the topologically simplify data, where the topology of the extracted isosurfaces was indeed simplified (fewer connected components and handles). We applied our approach to the extraction of prominent filament structures in 3D data, and showed that our pre-simplification of the data led to practical improvements for the removal of spurious loops in filament structures. We showed that our contributions could be used to repair genus defects in surface processing, where handles due to acquisition artifacts could be easily removed, with an explicit control on the repair primitives (cutting or filling).

4 Topology Aware Neural Interpolation of Scalar Fields

Contents

This chapter presents a neural scheme for the topology-aware interpolation of time-varying scalar fields. Given a time-varying sequence of

persistence diagrams, along with a sparse temporal sampling of the corresponding scalar fields, denoted as keyframes, our interpolation approach aims at "inverting" the non-keyframe diagrams to produce plausible estimations of the corresponding, missing data. For this, we rely on a neural architecture which learns the relation from a time value to the corresponding scalar field, based on the keyframe examples, and reliably extends this relation to the non-keyframe time steps. We show how augmenting this architecture with specific topological losses exploiting the input diagrams both improves the geometrical and topological reconstruction of the non-keyframe time steps. At query time, given an input time value for which an interpolation is desired, our approach instantaneously produces an output, via a single propagation of the time input through the network. Experiments interpolating 2D and 3D time-varying datasets show our approach superiority, both in terms of data and topological fitting, with regard to reference interpolation schemes.

The work presented in this chapter is currently under review for the IEEE Workshop on Topological Data Analysis and Visualization (TopoInVis).

Our Contributions in one Image

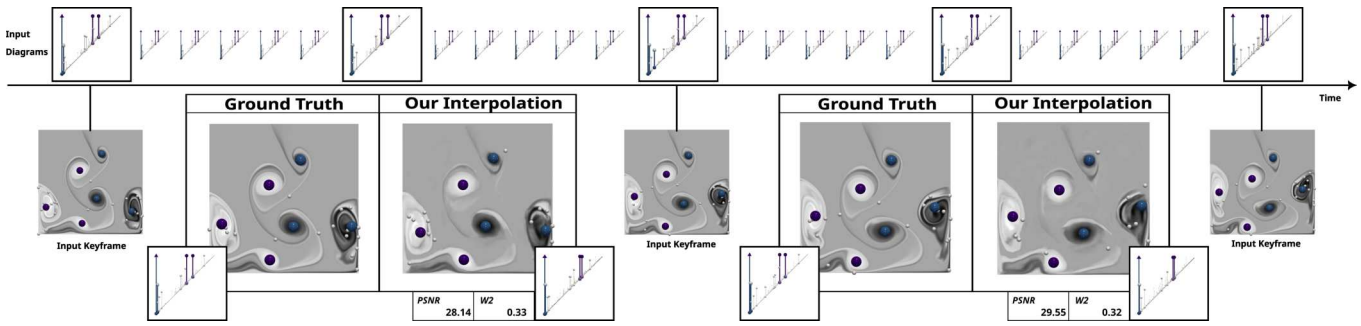

Figure 4.1 – *Given a time-varying sequence of persistence diagrams (top), along with a sparse set of corresponding keyframe scalar fields ("Input Keyframe" labels), our interpolation approach aims at "inverting" the non-keyframe diagrams and generating plausible estimations of the missing, non-keyframe scalar fields ("Our Interpolation" labels) enabling their visualization and analysis.*

4.1 Context

As computational resources and acquisition devices are becoming more efficient and precise, modern datasets are growing in resolution and level of details. This results in a general growth in the size of datasets which creates challenges for their storage, processing and analysis. To address this issue, *data reduction* is commonly considered, either by employing lossy compression schemes [Lin14, BRLP19] or by storing reduced data representations, which concisely, yet precisely, only encode the core features of the data, possibly during data production (i.e., *in-situ* [BAA+16, ABG+15]).

Topological Data Analysis (TDA) [EH09] precisely focuses on the robust encoding of the structural patterns of a dataset into concise *topological descriptors*, whose successful applications have been documented in a variety of domains and contexts [HLH+16].

Topological descriptors are often used as a mean of data reduction, typically by storing to disk these descriptors instead of the data itself. For instance, for time-varying datasets, a typical strategy [BWT+11, BNP+21, FPF+23] consists in storing the actual data at a low frequency (resulting in the storage of a small number n of *keyframes*), while storing topological descriptors at a higher frequency (e.g., for a large number $N \gg n$ of *non-keyframe* time steps). Then, the produced descriptors can be directly post-processed by dedicated statistical frameworks [TMMH14, LCO18, VBT20, YWM+19, PVDT22, PVT23, PT24], tailored to the analysis of topological representations. However, in many scenarios, a complete investigation might require going back to the original dataset which initially generated a given topological descriptor, for instance, for further visual inspection, interpretation and analysis. Then, the following question arises: *how can we reliably* "invert" *the construction of a topological descriptor?* (i.e., retrieve the dataset which generated it). Unfortunately, this inverse problem is ill-posed as many distinct datasets can generate the same topological descriptor (see Fig. 4.2). Then, further constraints need to be considered to exploit the available data, e.g., the stored *keyframes*.

This work addresses this issue by presenting a topology-aware interpolation approach. Given a reduced representation of an input time-varying scalar field (i.e., the persistence diagram of each time-step and a few *keyframes*), the overall goal of our work is to *invert* the non-keyframe diagrams. To achieve this goal, we exploit a generative neural architecture to interpolate the scalar field for a given time step. This network is trained

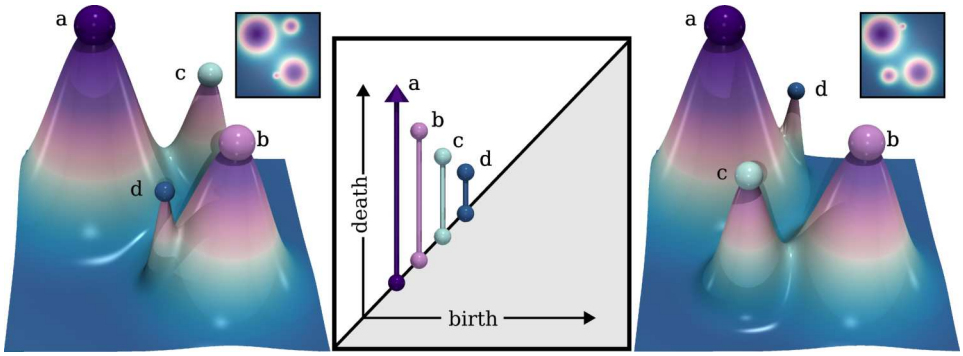

Figure 4.2 – *Scalar fields (height opposite) admitting a common persistence diagram (center). Each hill is encoded in the diagram by a vertical bar whose length encodes the* persistence *of the corresponding topological feature in the data (arrows indicate generators with infinite persistence). While the diagram encodes the list of topological features with their birth, death and persistence, it forgets their geometrical realization in the data.*

to learn the relation from the time parameter to the input time varying scalar field, based on the keyframe examples, in order to reliably extend this relation to the non-keyframes. Also, we show how augmenting this architecture with specific topology-aware losses exploiting the input diagrams both improves the geometrical and topological reconstruction of the non-keyframe time steps. At query time, our approach only requires as an input the time value for which an interpolation is desired and it instantaneously produces an interpolated scalar field, via a single propagation of the time through the neural network. Experiments interpolating 2D and 3D time-varying ground-truth datasets demonstrate the superiority of our model, both in terms of data and topological fitting, with regard to previous, baseline and neural interpolation schemes providing comparable query times.

4.1.1 Related work

This section reviews the literature related to our work, which can be classified into the following main categories.

Topological methods in visualization: The visualization community has been investigating Topological Data Analysis (TDA) [EH09] for more than two decades [HLH+16], with applications to a variety of domains, including combustion [BWT+11, GBG+14] fluid dynamics [KRHH11, NVBB+22] material sciences [GND+07, SPD+19], chemistry [BGL+18, OT23], or astrophysics [Sou11, SPN+16] to name a few. A key feature of TDA is its ability to robustly extract the structural patterns present in complex datasets and to efficiently represent them into concise represen-

tations. Such representations include persistence diagrams [ELZ02, BKR14, GVT23], merge [LWW+24] and contour trees [CSA00, GFJT19a], Reeb graphs [BGSF08, PSBM07, GFJT19b], or Morse-Smale complexes [GBHP08, RWS11, SN12, GBP19]. Moreover, another critical aspect of TDA is its ability to provide multi-scale hierarchies of the above topological data representations, enabling in consequence a multi-scale visualization, exploration and analysis of the topological features of the data. In that context, *topological persistence* [ELZ02] is an established importance measure which can be directly read from the persistence diagram and which can be used to drive the simplification of the above topological representations.

In many application scenarios, when handling time-varying data in scientific computing [BWT+11, BNP+21, FPF+23] (in particular *in-situ* [BAA+16, ABG+15]), these topological descriptors are often used as *proxies* to the data for the purpose of data reduction. For instance, in the context of simulating mosquito-borne disease spread, Brown et al. [BNP+21] store time steps of the data at a low frequency to reduce IO, while the persistence diagram (which is orders of magnitude smaller than the original data) is stored permanently at a higher frequency. Similar data reduction strategies based on the merge tree have also been documented [BWT+11]. Then, in a post-process, the resulting ensemble of topological proxies can be exploited by statistical frameworks [TMMH14, LCO18, VBT20, YWM+19, PVDT22, PVT23, PT24] which are tailored to the analysis of topological descriptors. In this work, we focus on exploiting these topological data representations (in particular persistence diagrams), in conjunction with a sparse temporal sampling of the actual data, to generate plausible visualizations of the missing, unstored data, in a way that favors topological feature preservation. For this, we rely on a neural scheme which integrates topological constraints, thanks to persistence optimization [PSO18, GGSG20, CCG+21, SWB21, NM22, KPLT24], as described in Sec. 4.2. Note that several schemes have been investigated for achieving topology-aware compression [SPCT18, LLW+25]. However, compression is a problem that is *orthogonal* to the setup studied in our work. First, compressors do have access to the *full* input data, which eases several aspects dealing with topological constraint enforcement and data value preservation. In contrast, our approach does *not* have access to the full input data but only to a reduced representation (the persistence diagram of each time step, as well as a few *keyframes*). Second, compression could be used in

conjuction to our work, e.g., by using topology-aware compressors to store the *keyframes*.

Neural methods for interpolation: The problem of interpolating fields in time appears frequently in both vision and visualization.

In computer vision, the idea of interpolating video frames from a sequence of images is a well known research topic, showing up applications such as producing slow-motion videos, temporal upsampling, and video compression. A recent survey by Dong et al. [DOD23] categorizes approaches into two high level categories: flow-based methods (which rely on an estimation of optical flow [LK81]) and kernel-based methods (which rely on evaluating differences in a fixed neighborhood of each pixel). This dichotomy is conceptually similar to Lagrangian vs. Eulerian methods, as observed by Meyer et al. [MWZ+15] who also propose looking at phase-based methods for video interpolation.

When working in the setting of video, there are sharp differences than one might consider for field data. Typically, video footage is assumed to be objects moving around in a scene, and thus it is reasonable to use an optical-flow based model that tracks the trajectories and velocities of individual pixels (as object samples). Techniques to compute optical flow have seen significant advances when using deep learning methods such as FlowNet [DFI+15, IMS+17], PWC-Net [SYLK18], and RAFT [TD20]. While these form an impressive backbone to many video interpolation techniques [JSJ+18, BLM+19, LXS+20, NL20, PKLK20, PLK21, HZH+22, RKT+22], they also can come at the cost of complex optimization procedures. Because of the framing of bidirectional flow, often these methods are limited to synthesizing single (or a fixed number of) frames between two input frames, and they also make relatively strong assumptions (reasonable for videos of objects) that pixel movement should be captured with flow.

Kernel-based methods [LKC+20, NML17, CKH+20, KPCT23], focus instead on localized difference in pixel values within a neighborhood, but as a result often cannot estimate motion accurately if between-frame movement is larger than the kernel size. The method VideoINR considers interpolating videos in both space and time simultaneously [CCL+22]. Closely related to our approach is the idea of modeling a video as a function in time, as proposed by Chen et al. in NeRV [CHW+21]. We also consider the idea of modeling a time-varying field as a network conditioned to produce an entire frame given a time step; however, for our work we model the "frame" as a 3D volume, and we guide the optimization of the net-

work with a topological loss. Notably, recent work in visualization has also built upon NeRV, such as NeRVI [GCW23] and FCNR [LGW24], but these works focus on building neural models for rendered images (2D) of time-varying volumes, rather than modeling the volume itself.

While most applications in computer vision are limited to 2D + time, some of these methods have been extended to 3D + time images, particularly coming from medical imaging [GBA+20, WLC+21, WWY+20]. Yin et al. considered adapting some of the above methods directly to the setting of coronary angiography [YLW+21].

More generally, to address modeling volumetric data, in visualization numerous authors have begun to consider machine learning methods for time-varying volumes. TSR-TVD is recent work in this space utilizing recurrent generative models for temporal super resolution [HW19]. Han et al. later considered the generalized problem of spatio-temporal super-resolution in STNet [HZCW21]. Both of these methods suffer from long training times and can only produce a fixed temporal scale factor, suffering from the same limitations as many video interpolation methods. STSRNet employs an two-stage framework using optical flow for spatial-temporal super resolution [ASS+21]. Recently, FLINT [GRF24] also used optical flow for temporal super resolution, building upon RIFE [HZH+22].

Following the introduction of implicit neural representation (INRs) to the visualization community for volume compression by Lu et al. [LJLB21], numerous authors considered their use in other applications. CoordNet leveraged INRs for multiple visualization tasks (including spatio-temporal super resolution and visualization synthesis) for time-varying ensembles [HW22]. While this approach produced a generalized framework, to address large training times and model sizes, different authors considered using knowledge distillation for either learning hypernetworks [WBCM23] or compressed models [HZB23] with CoordNet serving as the teacher model. FFEINR [JBY24] considered an INR for joint spatio-temporal super resolution based on VideoINR [CCL+22]. Finally, STSR-INR is a recent neural method for spatio-temporal super resolution, supporting multivariate ensemble data [TW24]. We consider STSR-INR as a representative example of state-of-the-art INRs to compare though, although it differs from our work in that it does not explicitly consider a loss based on topological features. Including such a loss directly into a coordinate-based network (that predicts positional samples of the field in batches, rather than an entire volume) would require a significantly more costly loss function.

4.1.2 Contributions

This chapter makes the following new contributions:

1. An approach combining *(i)* a generative neural model for scalar field interpolation with *(ii)* topological losses based on persistence diagrams, for constraining the topology *and* geometry of the output interpolations.
2. A TTK/PyTorch implementation for reproducibility.

4.2 Approach

This section presents our overall interpolation technique, given an input time-varying sequence of persistence diagrams, along with a sparse temporal sampling of the corresponding scalar fields. While the neural network architecture exploited in our approach is typical of related work [CHW+21], we still document its specifications in Sec. 4.2.2 for completeness and reproducibility purposes.

4.2.1 Overview

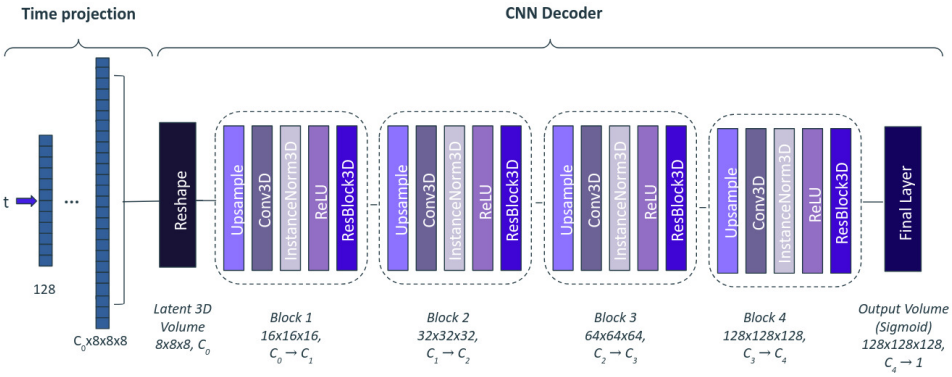

Figure 4.3 – TimeToScalarField *architecture. An input value t is encoded using sinusoidal positional encoding and projected through a fully connected layer into a latent 3D tensor of size* $C_0 \times 8 \times 8 \times 8$, *where* C_0 *denotes the initial number of channels. This tensor is processed by a CNN decoder composed of four sequential blocks, each consisting of:* (i) *trilinear upsampling (by a factor of 2),* (ii) *3D convolution (with kernel size* $3 \times 3 \times 3$*),* (iii) *instance normalization,* (iv) *ReLU activation, and* (v) *a residual block* (ResBlock3D, *[HZRS16]). The spatial resolution is progressively increased while the number of channels is reduced. A final convolution followed by a Sigmoid activation produces the output volume (here, with resolution* 128^3*).*

Our approach relies on a generative neural network architecture, referred to in the remainder as *TimeToScalarField*. It is presented in Sec. 4.2.2

and schematically illustrated in Fig. 4.3. It takes as an input a time parameter value $t \in [0,1]$, and generates a scalar data vector $v_{f(t)} \in \mathbb{R}^{n_v}$ defining a scalar field $f(t)$ on a cubical complex $\mathcal{C}$ (i.e., a regular grid) with dimensions c_x, c_y, c_z assumed to be multiples of powers of two (i.e., $n_v = c_x \times c_y \times c_z$).

The training of this architecture is performed on the N time-steps of the input temporal sequence, specifically with:

- the $n \ll N$ keyframe scalar fields and persistence diagrams, and
- the $(N-n)$ non-keyframe persistence diagrams (for which the scalar fields are *not* given).

Note that the interpolation of distinct input temporal sequences requires the training of distinct networks. The training is achieved by minimizing the overall loss presented in Sec. 4.2.3. In particular, this optimization is carried out by stochastic gradient descent [KB15], relative to the gradient of the training loss, obtained via automatic differentiation [PGM+19]. Note that for the $(N-n)$ non-keyframes, the corresponding scalar data is not available (only the persistence diagrams are available). Therefore, the loss terms involving scalar fields (e.g., *Mean Squared Error*, MSE) are set to zero for non-keyframe time values (see Sec. 4.2.3).

At query time, the time $t \in [0,1]$ for which an interpolation is desired is presented to the input of the trained neural network, which propagates it to return the vector $v_{f(t)}$, defining the output scalar field $f(t) : \mathcal{C} \to \mathbb{R}$.

4.2.2 Architecture

The *TimeToScalarField* architecture (Fig. 4.3) is designed to generate spatially structured scalar fields from time values. It includes *(i)* a time projection and *(ii)* a *convolutional neural network* (CNN) decoder.

Time projection: The input time value $t \in [0,1]$ is first presented to a *positional encoding* (PE) layer [GAG+17, VSP+17] as it is often reported (and confirmed by our experiments) to improve temporal coherence. This layer is *fixed* (i.e., not optimized) and simply maps t to a high-dimensional vector $PE(t) \in \mathbb{R}^T$. Each entry i of this vector is a sinusoidal function of t, with increasing frequencies for increasing values of i [VSP+17]:

$$\begin{cases} \left[PE(t)\right]_{2i} = \sin(2\pi t / 10,000^{2i/T}) \\ \left[PE(t)\right]_{2i+1} = \cos(2\pi t / 10,000^{2i/T}) \end{cases}.$$

This first positional encoding $PE(t)$ is then projected into a higher dimensional latent space $\mathbb{R}^{C_0 \times r_0 \times r_0 \times r_0}$ via sequential fully connected layers with ReLU activations (*"Reshape"* layer, Fig. 4.3). This projection enriches the latent representation and provides a robust foundation for the subsequent decoding. Such a re-projection approach has been widely validated in conditional generative frameworks [IZZE17]. The output of that layer forms the input for the subsequent CNN decoder.

CNN decoder: Our CNN decoder (Fig. 4.3, right) reconstructs the output data progressively through successive upsampling stages. The decoder starts with a coarse resolution ($r_0 \times r_0 \times r_0$ in 3D and $r_0 \times r_0$ in 2D) over C_0 *channels* (i.e., C_0 instances of the optimization, started at distinct random initializations). As recommended in the CNN literature, C_0 is progressively decreased via channel-wise filtering at each stage i into C_i, until reaching 1 (see Sec. 4.3.1 for further discussions). The initial resolution r_0 is multiplied by 2 at each stage i along each dimension until reaching the input resolution (i.e., $c_x \times c_y \times c_z$). Specifically, each processing block i (dashed boxes in Fig. 4.3) consists of:

1. an upsampling phase (with bilinear and trilinear interpolants in 2D and 3D respectively),
2. a convolutional block (with kernel sizes 3×3 and $3 \times 3 \times 3$ in 2D and 3D respectively),
3. instance normalization [UVL16] (to reduce overfitting and improve generalization)
4. a non-linear activation (ReLU) and
5. a residual block [HZRS16] (to stabilize the training and mitigate vanishing gradient issues often encountered in deep CNNs).

4.2.3 Losses

Given a batch of input values t, the output set of predictions provided by the neural network are evaluated with the following overall loss:

$$\mathcal{L} = \mathcal{L}_{MSE} + \alpha\mathcal{L}_{\nabla} + \beta\mathcal{L}_{CV} + \gamma\mathcal{L}_{\mathcal{W}_2}.$$

It is composed of four terms, detailed below.

Data fitting: This term is the traditional *Mean Squared Error* (MSE) which evaluates, only for the n keyframes (it is 0 otherwise), the fitting of each

prediction $v_{f(t_k)}$ to its training keyframe f_k:

$$\mathcal{L}_{MSE} = \frac{1}{n}\sum_{k=1}^{n}\frac{1}{n_v}\sum_{j=1}^{n_v}||v_{f(t_k)}(v_j) - f_k(v_j)||_2^2.$$

Gradient fitting: To improve the geometrical preservation of fine scale details, an additional term is considered, only for the n keyframe timesteps (it is set to 0 otherwise), to evaluate the fitting of the gradient of each network prediction $v_{f(t_k)}$ to that of its training keyframe f_k:

$$\mathcal{L}_{\nabla} = \frac{1}{n}\sum_{k=1}^{n}\frac{1}{n_v}\sum_{j=1}^{n_v}\sum_{i=1}^{n_d}\mathcal{L}_{\nabla_1}\Big(\big(\nabla v_{f(t_k)}(v_j)\big)_i, \big(\nabla f_k(v_j)\big)_i\Big),$$

where n_d is the dimensionality of the dataset (in our experiments, 2 or 3), where ∇ is the vector formed by the partial derivatives of the scalar field at a vertex v_j in the geometrical domain $\mathcal{C}$, and where $\mathcal{L}_{\nabla_1}$ is the so-called *smooth L_1 loss* [Gir15] (which presents the practical interest of the L_1 norm, while still being differentiable in 0):

$$\mathcal{L}_{\nabla_1}(x,y) = \begin{cases} 0.5(x-y)^2, & \text{if } |x-y| < 1, \\ |x-y| - 0.5, & \text{otherwise.} \end{cases}$$

Critical values: For each of its bars, in addition to its birth and death values, the persistence diagram also typically encodes in practice the identifiers v_b and v_d of the vertices respectively implied in the birth and death of the corresponding topological feature (Sec. 2.2). The set of all birth and death vertices form the *critical points* of the underlying piecewise linear scalar field [Ban67, GVT23]. Specifically, the scalar value of a birth (respectively death) vertex v_b (respectively v_d) is given by the birth (respectively death) of its bar in the diagram. This information, which is available for all the N timesteps, can be re-used to enforce the scalar value at the precise location of each critical point, helping preserve the actual location of the topological features in the geometrical domain $\mathcal{C}$ (persistence alone *forgets* the geometrical realization of the data, Fig. 4.2). This can be achieved with the following loss, which evaluates, for each prediction $v_{f(t_k)}$, the fitting between the pointwise value $v_{f(t_k)}(v_j)$ and the corresponding value $f_k(v_j)$ given by the input persistence diagram $\mathcal{D}_k$ for each vertex v_j in the set of critical points CP_k of f_k:

$$\mathcal{L}_{CV} = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{n_{CP_k}}\sum_{j=1}^{n_{C_k}}||v_{f(t_k)}(v_j) - f_j(v_j)||_2^2.$$

Topology correction: To enforce topological preservation, we evaluate the topological fitting between the diagrams $\mathcal{D}(v_{f(t_k)})$ and their target inputs

$\mathcal{D}_k$ with the Wasserstein distance (Sec. 2.3):

$$\mathcal{L}_{\mathcal{W}_2} = \frac{1}{N}\sum_{k=1}^{N} \mathcal{W}_2\big(\mathcal{D}(v_{f(t_k)}), \mathcal{D}_k\big).$$

This loss effectively quantifies and penalizes topological errors. Since the Wasserstein distance is a definable function of persistence [CCG+21], it can be used within the optimization framework from Sec. 2.4 (i.e., $\mathcal{E} = \mathcal{L}_{\mathcal{W}_2}$), with guaranteed convergence [DDKL20, CCG+21], in particular in conjunction with the above losses, which are convex and differentiable.

4.2.4 Computational details

This section provides practical details for the training of the model presented in Sec. 4.2.2 with regard to the overall loss described in Sec. 4.2.3. The training is organized into two phases:

1. *scalar field training* (with the following loss weights, Sec. 4.2.3: $\alpha = 0.1$, $\beta = 1$ and $\gamma = 0$) and
2. *topology correction* (with loss weights: $\alpha = 0$, $\beta = 1$, $\gamma = 1$).

This two-phase strategy is motivated by the computational effort required by the *topology correction* step (which involves the computation of a persistence diagram at each iteration). Then, this strategy first learns quickly a plausible geometry for the missing scalar fields (*scalar field training*) prior to optimizing their topology with more expensive computational efforts (*topology correction*). Also, from a practical standpoint, our initial experiments reported that this two-phase strategy improved the convergence of the training.

For each phase, the entire input (N diagrams, n keyframes) is presented to the network at each epoch, for a number n_1 and n_2 of epochs for the two phases respectively (see Sec. 4.3.1 for further discussions). The purpose of this decomposition is to first generate a good estimation of the scalar fields in phase 1, prior to refining their topology in phase 2 (which is significantly more expensive computationally). Specifically, in phase 2, the last layer of temporal projection as well as the first block of the CNN decoder are *frozen*, in order to maintain through phase 2 the large scale details learned in phase 1. Also, to mitigate overfitting and improve generalization, dropout regularization [WZZ+13] is used, omitting randomly the update of a network weight, with a probability set to 0.1.

Each epoch of topology correction (phase 2) requires the computation of N persistence diargams $\mathcal{D}(v_{f(t_j)})$ as well as the estimation of the

Wasserstein distances to their input targets $\mathcal{D}_j$, which is done in practice in quadratic time (with the number of vertices n_v in the output grid). To accelerate this process, we *prune* each input target diagram $\mathcal{D}_j$ by removing its bars with a persistence (Sec. 2.2) smaller than 1% of its largest bar, which is a typical persistence thresholding in the applications. We prune similarly $\mathcal{D}(v_{f(t_j)})$ and fix the assignment of the pruned bars to the diagonal (which is equivalent to the destruction of these noisy features). This two-stage pruning reduces the size of the diagrams and drastically accelerates the optimal assignment optimization at the basis of the Wasserstein distance computation (Sec. 2.3). Finally, we use shared-memory parallelism to compute each diagram (and its distance to its target) in a distinct task.

The minimization of the overall loss (Sec. 4.2.3) is performed with the Adam solver [KB15] (with weight decay set at 10^{-6}, to mitigate overfitting), with a relatively low initial learning rate to favor a stable optimization (see Sec. 4.3.1 for numerical values).

Finally, this architecture (Sec. 4.2.2) is subject to several meta-parameters which we adjusted empirically, e.g., $T = 128$, $r_0 = 8$ (see Sec. 4.3.1, for dataset specific parameters).

4.3 Results

This section presents experimental results obtained with a PyTorch [PGM+19] implementation of our approach, using TTK [TFL+17, BMBF+19] (for persistence diagram computation and matching). Experiments were performed in a Google Colab environment, with an Nvidia A100-SXM4 GPU (RAM: 40 GB) and an Intel Xeon CPU (2.2 GHz, 6 cores, RAM: 80 GB).

4.3.1 Test datasets

Our experiments include both synthetic and real-life time-varying 2D and 3D scalar fields. For convenience, the considered 2D (respectively 3D) datasets have all been resampled to an initial resolution of 512^2 (respectively 128^3). Moreover, to replicate a setting that is typical of in-situ data production [BWT+11, BNP+21, FPF+23], we selected as *keyframes* 10% of the time steps (i.e., $n = N/10$), uniformly distributed in the time interval $[0, 1]$. The remaining 90% of the time steps were considered as ground-

truth data (not seen by the model during training). Specifically, we considered the following datasets.

Gaussian mixture: This synthetic 2D dataset counts $N = 180$ time steps representing a mixture of six Gaussians, where the Gaussian centers (captured by persistent maxima) evolve through time. Specifically, to evaluate the robustness of our method to value changes as well as feature displacements, this dataset is decomposed into segments where only the weights of a few Gaussians evolve (i.e., making hills appear or disappear in the corresponding terrain), segments where only the center positions for a few Gaussians evolve (i.e., displacing hills in the corresponding terrain) and segments where both phenomena occur.

Mixing vortices: This 2D dataset counts $N = 150$ time steps and represents the vorticity (measured as the orthogonal curl component) of a 2D flow generating a von Karman street (see [TTK20] for downloads). This dataset has the particularity to model a 2D domain with boundaries. Then, in a first segment, the vortices of the street (captured by persistent extrema) follow a typical, common translation motion. However, in a second segment, the vortices hit the boundary and consequently start to mix together in a complicated, turbulent, whirling pattern.

Isabel: This 3D dataset counts $N = 48$ time steps representing the evolution of wind velocity for the Isabel hurricane [WBKS04]. To capture wind regions interacting with the boundary of the domain, we will consider as input scalar field the opposite of the wind velocity. Then, regions associated with high winds will be captured by persistent minima of this opposite velocity. In particular, the eye of the hurricane travels through the domain into several stages (formation, drift, and landfall).

Asteroid impact: This 3D dataset counts $N = 50$ timesteps and represents the impact of an asteroid with the sea at the surface of the Earth [PG18]. The considered scalar field is matter density, which distinguishes well the asteroid from the water and the ambient air in this simulation. In this dataset, the trajectory of the asteroid as well as the topological features resulting from the impact with the sea are captured from a topological point of view by persistent maxima.

Since these datasets exhibit distinct levels of details in terms of geometrical features and temporal variability, several meta-parameters of our approach were empirically adjusted on a per dataset basis (Tab. 4.1). Fig. 4.4 provides curves plotting the corresponding loss, for each dataset.

Table 4.1 – *Meta-parameters adjusted empirically to account for the variability in geometrical/temporal complexity across our datasets.*

Dataset	C_0	C_1	C_2	C_3	C_4	C_5	C_6	Learning rate	n_1	n_2
Gaussian mixture (2D)	256	128	64	64	32	16	8	0.5×10^{-3}	6000	100
Mixing vortices (2D)	256	128	64	64	32	16	8	0.5×10^{-3}	6000	100
Isabel (3D)	128	64	32	32	16	–	–	10^{-3}	3000	100
Asteroid impact (3D)	512	256	128	64	16	–	–	10^{-4}	3000	100

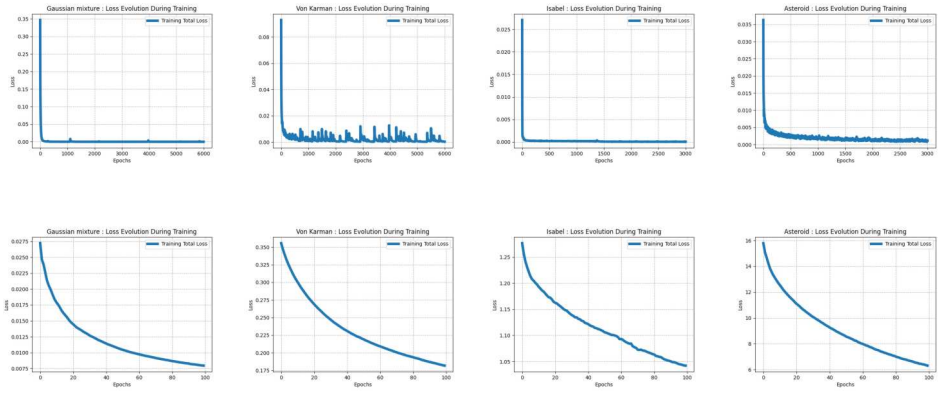

Figure 4.4 – *Loss evolution during* scalar field training *(phase* 1*, top) and* topology correction *(phase* 2*, bottom) for our datasets (from left to right: Gaussian mixture, Mixing vortices, Isabel, Asteroid impact).*

4.3.2 Reference approaches

We compare our technique to two approaches, selected based on their query time, which is comparable to ours (typically, below a second in practice). First, we consider as baseline the traditional pointwise, linear interpolation with regard to time. This scheme requires the evaluation of a linear equation at each vertex of the input grid. Next, among the variety of neural approaches introduced in the visualization literature, we considered the approach STSR-INR [TW24], which is a recent representative of neural methods for spatio-temporal super resolution. In particular, we used the implementation provided by the authors, set up to its default recommended parameters, except for:

- the epoch number (increased to obtain similar training times);
- the model size (increased to obtain sizes also similar to ours).

The purpose of these modifications was to provide the same computational resources to both methods (STSR-INR [TW24] and ours). Finally, note that in both cases (linear interpolation and STSR-INR), these reference approaches have only access to the *keyframe* data.

4.3.3 Quantitative criteria

We evaluate the quantitative quality of the resulting interpolations, for the non-keyframes only, in the light of two fitting terms. First, we evaluate a *data fitting* term with the traditional *Peak Signal-to-Noise Ratio* (PSNR). Second, we evaluate a *topological fitting* term, which measures the topological accuracy of an interpolation with regard to the input target diagram, based on the Wasserstein distance (Sec. 2.3).

For information, we also report indicators related to the computational resources used in our experiments, namely model size (in MB) and training time (in seconds).

4.3.4 Loss influences

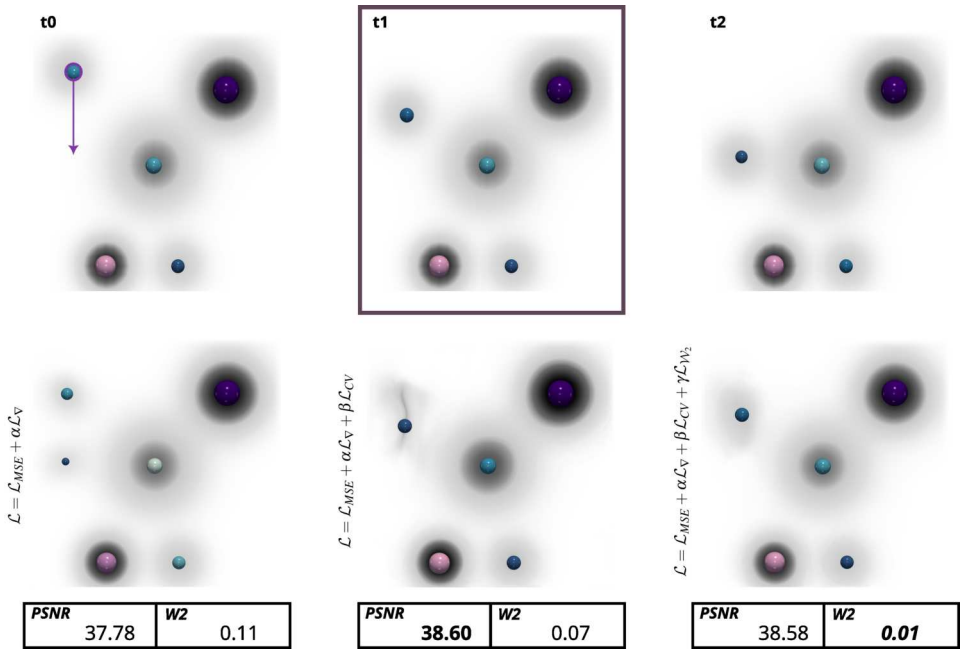

Figure 4.5 – *Influence of our individual loss terms (Sec. 4.2.3) on the interpolation.* Top: *Three time steps of the* Gaussian mixture *dataset (spheres: local maxima, colored and scaled by persistence). From t_0 to t_2, a Gaussian center is moving down, vertically (purple arrow, t_0).* Bottom: *Interpolations obtained for the time step t_1 with different loss blending coefficients (Sec. 4.2.4), from left to right:* (i) $\alpha = 0.1$, $\beta = \gamma = 0$ *(no topological loss),* (ii) $\alpha = 0.1$, $\beta = 1$, $\gamma = 0$ *(critical value enforcement),* (iii) $\alpha = 0.1$, $\beta = 1$, $\gamma = 1$ *(topology correction). Without our topological losses (*bottom, left*), the interpolation exhibits the* superposition artifact *typical of linear interpolation, with the moving Gaussian being replaced by two Gaussians of decreased height (at the start and end points of the displacement). Critical value enforcement (*bottom, center*) addresses this issue, resulting in one maximum in the correct location, while topology correction (*bottom, right*) improves the final geometry and topology.*

We start the practical analysis of our method by investigating the effects of the individual terms of our loss (presented in Sec. 4.2.3). Fig. 4.5 presents interpolation results on our synthetic dataset (*Gaussian mixture*) for various loss blending coefficients. Specifically, this figure focuses on a temporal sequence where a Gaussian center is moving down vertically (purple arrow). When using only the *MSE* and gradient losses (bottom, left), our method results in interpolations with a *superposition artifact* that is typical of linear interpolation: the moving Gaussian has been replaced by two Gaussians of decreased height (see the two maxima), at the start and end points of the displacement. The introduction of our loss based on the critical values reported by the input diagrams greatly contributes to addressing this issue (bottom, center). Specifically, it results in a persistent maximum, at the right value and at the right location. However, the corresponding feature exhibits a ridge-like geometry (dark curve) which does not resemble the original feature (a Gaussian, top-center). The introduction of topology correction (based on the Wasserstein distance to the input diagram, bottom right) further improves topological preservation (in terms of Wasserstein distance), as well as, importantly, the geometry of the prediction: the feature associated to the moving maximum exhibits a shape that is closer visually to the original Gaussian. This illustrates that our topological losses, on top of improving the preservation of the features of interest, also contribute to improving the overall geometry of the prediction.

4.3.5 Comparisons

This section provides a quantitative and qualitative comparison between our method and reference approaches (Sec. 4.3.2).

Table 4.2 – *Quantitative scores (averaged over all non-keyframe timesteps) over our test datasets for the linear interpolation, STSR-INR [TW24] and our method. For each score, the best value is reported in bold. Model sizes and training times are also reported.*

Criterion	Method	Gaussian Mixture	Mixing vortices	Isabel	Asteroid Impact
PSNR (↑)	Linear Interpolation	37.44	26.20	29.56	20.81
	STSR-INR [TW24]	36.18	26.17	29.05	20.06
	Our method	**38.58**	**29.28**	**32.41**	**22.51**
$\mathcal{W}_2$ (↓)	Linear Interpolation	0.11	0.53	0.83	4.56
	STSR-INR [TW24]	0.09	0.57	0.79	6.27
	Our method	**0.01**	**0.29**	**0.66**	**2.83**
Size (MB)	STSR-INR [TW24]	11	11	33	157
	Our method	11	11	33	157
Training (h.)	STSR-INR [TW24]	3.33	3.31	6.76	9.69
	Our method	3.20	2.76	5.70	8.96

Tab. 4.2 provides an overview of the quantitative scores (averaged over all non-keyframe time steps) over our test datasets for the considered methods. This table shows that our method, by design, provides the best topological accuracy (as measured by the Wasserstein distance to the input diagrams). As discussed in Sec. 4.3.4, our topological losses (Sec. 4.2.3), on top of contributing to the preservation of topological features, also contribute to improving the geometry of the prediction. This is illustrated by the PSNR scores of our method, which are superior to those of the reference approaches.

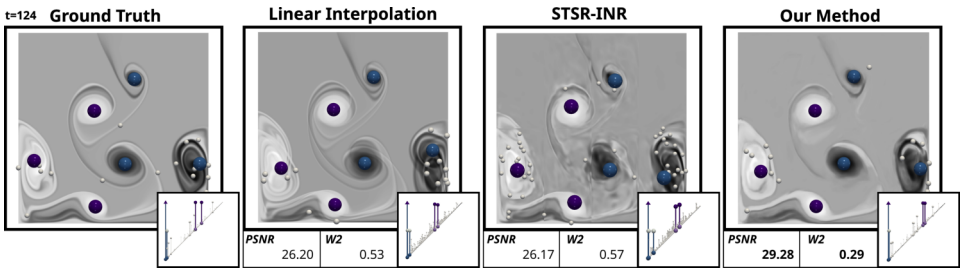

Figure 4.6 – *Comparison of temporal interpolations on the* Mixing vortices *dataset (from left to right: ground-truth, linear interpolation, STSR-INR [TW24] and our method). Persistent extrema are reported as colored spheres (blue: minima, purple: maxima), extrema of intermedia persistence are reported as white spheres. The persistence diagrams are reported in the bottom insets. The linear interpolation exhibits a typical* superposition artifact, *where the two keyframes used for the interpolation are superposed on top of each other. This artifact is addressed by STSR-INR, with an improved PSNR. Our method further improves PSNR, while improving topological accuracy (in particular with fewer noisy bars in the diagram).*

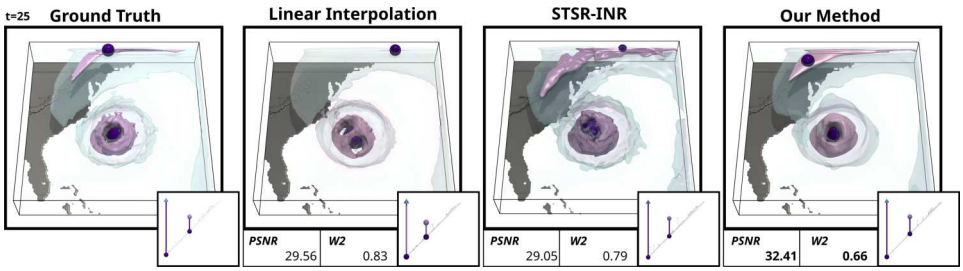

Figure 4.7 – *Comparison of temporal interpolations on the* Isabel *dataset (from left to right: ground-truth, linear interpolation, STSR-INR [TW24] and our method). Persistent minima of the opposite wind velocity are reported as purple spheres. A few isosurfaces are shown to represent the geometry of the data. The persistence diagrams are reported in the bottom insets. The linear interpolation exhibits its typical* superposition artifact *(similarly to Fig. 4.6), where the hurricane eyes from two keyframes are superposed. Moreover, it fails at capturing certain high wind regions (light purple surface, top). Here, STSR-INR produces a reconstruction with a degraded PSNR. Our method provides the best PSNR and the best topological accuracy.*

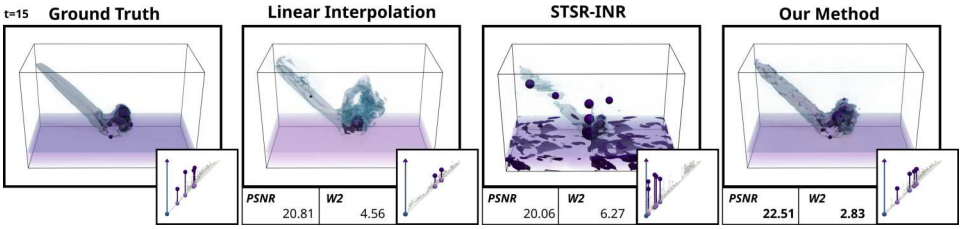

Figure 4.8 – *Comparison of temporal interpolations on the* Asteroid impact *dataset (from left to right: ground-truth, linear interpolation, STSR-INR [TW24] and our method). Persistent maxima are reported as purple spheres. The geometry of the data is represented via volume rendering and isosurfacing. The persistence diagrams are reported in the bottom insets. The linear interpolation exhibits its typical* superposition artifact*, where the two keyframes used for the interpolation are superposed on top of each other (similarly to Fig. 4.6). Here, STSR-INR produces a result with a degraded PSNR. Our method provides the indicators scores (PSNR and topological accuracy), and it provides the result which conforms best visually to the ground truth.*

Qualitative comparisons are provided in Figs. 4.6, 4.7, 4.8. Overall, these figures show that the linear interpolation suffers from a typical *superposition artifact*, where the two keyframes used for the interpolation are superposed on top of each other. This is particularly severe when features of interest are moving within the domain across time (which occurs in all our test datasets). This is partly addressed, sometimes successfully (Fig. 4.6), sometimes less successfully (Fig. 4.8), by the STSR-INR [TW24] approach, which improves PSNR in some case (Fig. 4.6). In all cases, our approach better preserves, by design, the topological features, resulting in a superior topological accuracy. Also, in several instances (Figs. 4.7, 4.8), our method provides the result which clearly conforms best visually (in terms of geometry) to the ground truth (confirming the observations of Sec. 4.3.4).

In principle, the results of the reference approaches (e.g., STSR-INR [TW24]), could be post-processed to improve their topological accuracy. This can be done, by example, by using the approach by Kissi et al. [KPLT24]. However, as shown in Fig. 4.9, such a post-processing step is non-negligible in terms of runtime, which significantly degrades interpolation query response times. In contrast, at query time, our approach provides a result with comparable topological accuracy instantaneously.

4.3.6 Limitations

An obvious limitation of our work, which is intrinsic to neural approaches in general (such as STSR-INR [TW24]), is the computational effort required

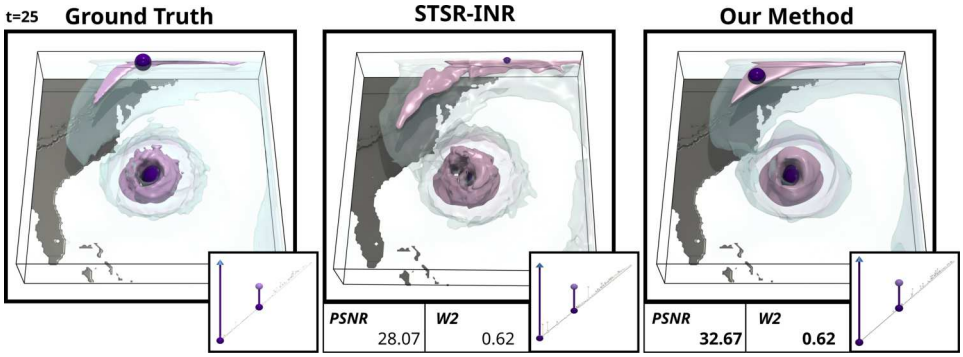

Figure 4.9 – *Comparing STSR-INR with a topological optimization post-process [KPLT24] (center) to the ground truth (left) and our method (right). To obtain a topological accuracy comparable to our method, the predictions provided by STSR-INR need to undergo a non-negligible optimization post-process (56 seconds) while our method is instantaneous.*

for training such models, in the range of hours of computation (Tab. 4.2). However, for applications such as in-situ computing [BAA+16, ABG+15], for which data storage can be a more important concern than computational effort, we believe our data reduction strategy to be still relevant, as assessed by our model sizes (Tab. 4.2).

Another limitation that we observed was the need for larger models for the datasets exhibiting the most geometrical and temporal complexity (Tabs. 4.1, 4.2). While it is understandable that more parameters are required in the model to capture this variability, this model size increase negatively impacts training computation times as well as model storage. Similarly, larger output sizes may require in general more iterations, themselves being more computationally expensive (as each iteration involves the computation of the persistence diagram).

Finally, our approach is currently restricted to regular grids, which is the only data representation supported by our CNN-based decoding. Alternative generative architectures would need to be considered for more generic inputs, e.g., scalar fields defined on tetrahedral meshes.

4.4 Summary

This chapter presented a neural approach for the topology aware interpolation of scalar fields. Our work was motivated by a data model often encountered in in-situ computing [BAA+16, ABG+15], where snapshots of the considered time-varying data are only saved at a low frequency (every *n keyframes*) and where reduced representations, such as topological descriptors, are stored at a higher frequency [BWT+11, BNP+21, FPF+23]

(every $N \gg n$ steps). Specifically, given an input sequence of persistence diagrams and a sparse temporal sampling of the corresponding data, our approach *"inverts"* the non-keyframe diagrams to produce plausible estimations of the missing data. Extensive experiments showed the superiority of our method over reference approaches for preserving the topological features of interest along the interpolation. Interestingly, our experiments also revealed that our topology-aware losses could also significantly contribute to improving the *geometry* of the interpolated data.

We believe our work opens several research avenues, in particular, thanks to its instantaneous query time, for the interactive exploration of ensembles of topological descriptors, as studied in topology-tailored statistical frameworks [TMMH14, LCO18, VBT20, YWM+19, PVDT22, PVT23, PT24]. However, a multidimensional extension of approach (to account for more ensemble parameters than simply time) would need to be investigated.

Conclusion

5

This thesis addressed the central challenge posed by the increasing size, resolution, and structural complexity of scalar datasets, which create significant difficulties in terms of their storage, analysis, and visualization. To mitigate these challenges, topological descriptors such as persistence diagrams offer concise and robust representations that effectively capture the main topological structures of interest within the data. However, although they characterize these structures well, persistence diagrams alone are not sufficient to fully apprehend the geometric complexity of the data, which often requires additional information or reconstruction techniques to be properly understood.

To overcome this fundamental limitation, the thesis introduced two complementary approaches for efficient topological simplification and reconstruction of scalar data.

Firstly, we proposed a specialized numerical optimization framework for topological simplification, significantly improving computational efficiency (up to ×60 faster than state-of-the-art methods) through tailored algorithmic accelerations. This allowed practical and precise simplification of complex three-dimensional datasets, effectively removing artifacts such as spurious loops in filamentary structures or topological defects on surfaces, while preserving essential topological features.

Secondly, to tackle the reconstruction challenge explicitly, we presented a neural interpolation framework that leverages sparse temporal sampling (keyframes) and associated topological descriptors to reconstruct intermediate scalar fields. Incorporating topology-aware loss functions, our approach outperformed existing techniques in both topological and geometric accuracy, demonstrating particular relevance in in-situ

computational contexts where only partial data snapshots are available.

Together, these two contributions provide a coherent methodology for addressing the initial challenge of efficiently managing large-scale, topologically complex scalar data. By integrating topological constraints into optimization and machine learning frameworks, they significantly enhance the capabilities of topological data analysis tools, opening promising new directions for the interactive exploration, visualization, and analysis of large datasets. Future work could extend these methodologies to more general persistence optimization problems, broader classes of multidimensional data, and real-time interactive contexts.

Ultimately, the methods developed in this thesis not only advance computational tools for topological analysis but also offer promising applications in critical domains such as medical imaging, fluid dynamics, and climate modeling, areas in which understanding and preserving topological structures is often crucial. For instance, we illustrate the relevance of our methods in medical imaging through the extraction and simplification of arterial networks (Fig. 3.1), in fluid dynamics with the Mixing vortices dataset (Fig. 4.6), and in climate modeling via the Isabel hurricane dataset and the Asteroid impact dataset (Fig. 4.7 and Fig. 4.8). These examples highlight how our approach can capture and preserve essential topological features across a wide range of real-world scenarios.

5.1 Summary of Contributions

This thesis introduced two novel methodological contributions designed to address the challenges posed by the growing complexity of scalar datasets through efficient topological simplification and accurate reconstruction methods. Both contributions are accompanied by practical, reproducible implementations (C++ and Python), enabling direct adoption by the scientific community:

Topological Simplification Optimization

An efficient solver tailored for topological simplification of scalar fields, featuring optimized procedures for rapid updates of persistence diagrams and pair assignments. These accelerations enable topological simplification to be effectively applied to real-world datasets. The approach was

demonstrated through applications including direct visualization of simplified data, extraction of filament structures with reduced spurious loops, and controlled genus repair of surfaces. A reproducible C++ implementation is provided, and experiments from this work are publicly accessible online via the TTK website.

Neural Topology-Aware Interpolation

A generative neural architecture specifically designed for topology-guided temporal interpolation of scalar fields. This model employs tailored, topology-aware losses informed by sequences of persistence diagrams, improving the preservation and reconstruction of key topological features during interpolation. A reproducible implementation using TTK and PyTorch is also available.

Together, these contributions directly address the central challenge initially posed by the increasing complexity and size of modern scalar datasets. By combining topological simplification techniques with neural interpolation methods that incorporate topological constraints, we provide practical and robust solutions for efficiently managing scalar data through compact yet meaningful representations. Ultimately, this integrated approach bridges the gap between abstract topological descriptors and their underlying data, significantly enhancing both the interpretability and the fidelity of scalar-field reconstruction, and opening new avenues for effective data analysis, visualization, and interpretation in various scientific domains.

5.2 Limitations

The methods developed in this thesis inherit certain intrinsic limitations related to their numerical and neural nature. The topological simplification solver presented in the first part is limited by numerical precision, meaning it cannot completely remove persistence pairs but only minimize their persistence down to a minimal numerical threshold (typically around 10^{-6} in single precision). This issue can leave residual pairs that must subsequently be addressed through combinatorial post-processing methods, which themselves may not always guarantee complete simplification due to the inherent NP-hardness of the underlying problems. Additionally, the solver generates solutions corresponding to local minima of the simplifi-

cation energy, with no guarantee of achieving globally optimal simplifications.

The neural-based interpolation method presented in the second part also carries its own inherent limitations. Most notably, it requires significant computational resources for training, typically spanning several hours. This computational overhead becomes increasingly pronounced for datasets exhibiting greater geometric and temporal complexity, as larger neural models are required, leading to increased training times and greater storage needs. Furthermore, the current neural approach is restricted to data defined on regular grids, limiting its direct applicability to more general representations such as tetrahedral meshes, unless alternative generative architectures are explored.

Overall, while both methods have demonstrated substantial practical value, addressing these numerical precision issues, computational scalability, and data representation constraints constitutes critical directions for future research.

5.3 Perspectives

The outcomes of this thesis open several promising directions for future research. Regarding the topological simplification optimization approach, a relevant next step would be to investigate extensions to other persistence optimization problems and explore further acceleration strategies tailored specifically for these problems. Another interesting direction involves developing parallelization strategies based on divide-and-conquer methods to enhance scalability and performance on larger datasets.

Concerning the neural topology-aware interpolation method, an important future direction is to generalize the proposed approach beyond regular grids, such as scalar fields defined on tetrahedral meshes.

Additionally, extending the method to multidimensional settings (involving parameters beyond just time) could significantly enhance interactive exploration and analysis capabilities for large ensembles of topological data. Fig. 5.1 presents preliminary results in this direction. This figure considers a synthetic 2D dataset (resolution 512×512) consisting of three evolving Gaussians: one fixed in the bottom right corner, one moving vertically and growing with increasing y, and another moving horizontally and growing with increasing x. For this experiment, we used 9 keyframes (highlighted in green) as inputs, with the interpolated results corresponding to points highlighted in red. Specifically, our interpolation scheme

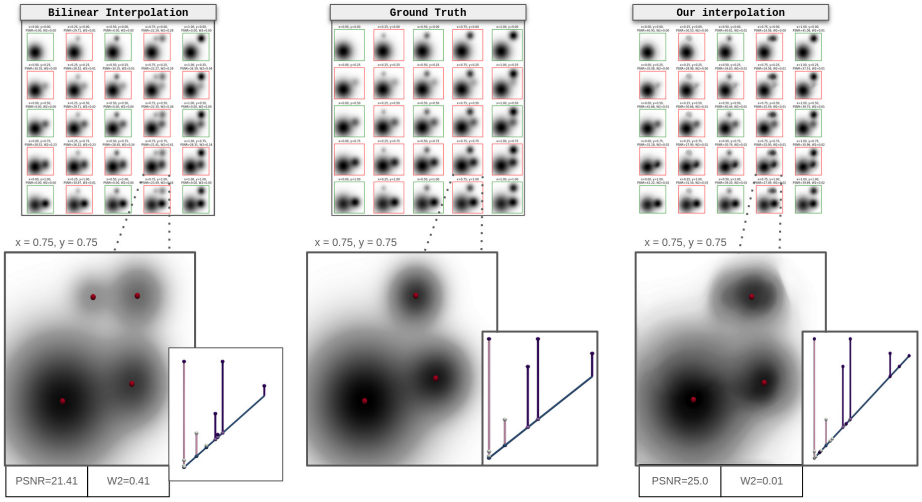

Figure 5.1 – *Comparison between classical bilinear interpolation and our multidimensional neural topology-aware interpolation on a synthetic dataset of three evolving Gaussians. The experiment used 9 keyframes (framed in green) as inputs, with the interpolated results shown in red. The top row presents bilinear interpolation (left), ground truth (center), and our neural method (right). The bottom row shows detailed results at* $(x = 0.75, y = 0.75)$ *along with corresponding persistence diagrams. Bilinear interpolation incorrectly introduces a fourth Gaussian-like structure with low persistence, whereas our approach preserves the correct number of features and achieves higher PSNR and lower Wasserstein (W2) distances.*

was extended to account for two input parameters instead of time (e.g., the above parameters x and y of the Gaussian mixture, using 6000 epochs in the first phase and 300 epochs in the topology correction phase).

This preliminary result shows that our multidimensional model successfully reconstructs the correct topological structures across the domain. In particular, when observing the interpolated data at $(x = 0.75, y = 0.75)$, standard bilinear interpolation introduces an additional fourth Gaussian, a typical artifact of linear interpolation. In contrast, our method accurately maintains the expected three Gaussian structures and yields persistence diagrams closely matching the ground truth. Quantitatively, our model achieves a PSNR of 25.0 compared to 21.41 for bilinear interpolation, and a significantly reduced Wasserstein distance of 0.01 versus 0.41. When averaged over all interpolated data points, our method still clearly outperforms bilinear interpolation, with a mean PSNR of 29.81 versus 27.82 and a mean W2 distance of 0.01 compared to 0.16. Motivated by these promising results on synthetic data, a future direction consists in evaluating our method on real-life datasets (e.g., the Isabel hurricane dataset), moving from controlled $2D$ scenarios to real-world $3D$ data to further assess scalability and topological fidelity.

One notable application of our multidimensional interpolation would be in conjunction with existing statistical frameworks for processing topological representations [SDT23, PVT23, PT24]. Such frameworks enable dimensionality reductions of an ensemble of topological descriptors (e.g., in the plane, for visualization purposes), hence providing *intrinsic* parameters for our multidimensional model. Our approach could nicely complement this pipeline by generating the underlying scalar fields over the (x, y) domain, constrained to align with the target persistence diagrams furnished by the statistical framework under consideration. Thus, our neural model could serve as a scalar reconstruction module within their topological analysis workflow, enabling full end-to-end multidimensional interpolation that respects both geometric and topological constraints.

Finally, addressing the computational cost and reducing neural model sizes while maintaining topological accuracy remains a meaningful challenge to enhance the practical applicability of these methods in real-world scenarios.

Bibliography

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. (Cited page 38.)

[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 pages 3 and 26.)

[AAY06] Pankaj K. Agarwal, Lars Arge, and Ke Yi. I/O-efficient batched union-find and its applications to terrain analysis. In *Symposium on Computational Geometry*, 2006. (Cited pages 27 and 28.)

[ABD+13] Dominique Attali, Ulrich Bauer, Olivier Devillers, Marc Glisse, and André Lieutier. Homological reconstruction and simplification in $\mathbb{R}^3$. In *Symposium on Computational Geometry*, 2013. (Cited pages 27, 28, 31, 44, 45, and 49.)

[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 1, 56, 58, and 73.)

[AGLM09] D. Attali, M. Glisse, F. Lazarus, and D. Morozov. Persistence-Sensitive Simplification of Functions on Surfaces in Linear Time. In *TopoInVis*, 2009. (Cited pages 27 and 28.)

[ASS+21] Yifei An, Han-Wei Shen, Guihua Shan, Guan Li, and Jun Liu. Stsrnet: Deep joint space–time super-resolution for vector field visualization. *IEEE computer graphics and applications*, 41(6):122–132, 2021. (Cited page 60.)

[BAA+16] Andrew C. Bauer, H. Abbasi, J. Ahrens, H. Childs, B. Geveci, S. Klasky, K. Moreland, Patrick O'Leary, V. Vishwanath, B. Whitlock, and E W. Bethel. In-situ methods, infrastructures, and applications on high performance computing platforms. *Computer Graphics Forum*, 2016. (Cited pages 1, 56, 58, and 73.)

[Ban67] T. F. Banchoff. Critical points and curvature for embedded polyhedral surfaces. *The American Mathematical Monthly*, 45(1):245–256, 1967. (Cited pages 3, 26, and 64.)

[Bar94] S. Barannikov. Framed Morse complexes and its invariants. *Adv. Soviet Math.*, 1994. (Cited page 13.)

[BC91] Dimitri P. Bertsekas and David Castañon. Parallel synchronous and asynchronous implementations of the auction algorithm. *Parallel Computing*, 1991. (Cited pages 33 and 38.)

[BDSS18] Alexander Bock, Harish Doraiswamy, Adam Summers, and Cláudio T. Silva. TopoAngler: Interactive Topology-Based Extraction of Fishes. *IEEE TVCG*, 24(1):812–821, 2018. (Cited pages 3 and 26.)

[BEHP04] Peer-Timo Bremer, Herbert Edelsbrunner, Bernd Hamann, and Valerio Pascucci. A topological hierarchy for functions on triangulated surfaces. *IEEE TVCG*, 2004. (Cited page 28.)

[BGL+18] Harsh Bhatia, Attila G. Gyulassy, Vincenzo Lordi, John E. Pask, Valerio Pascucci, and Peer-Timo Bremer. Topoms: Comprehensive topological exploration for molecular and condensed-matter systems. *J. of Computational Chemistry*, 39(16):936–952, 2018. (Cited pages 3, 26, and 57.)

[BGSF08] S. Biasotti, D. Giorgio, M. Spagnuolo, and B. Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 392(1-3):5–22, 2008. (Cited pages 3, 26, and 58.)

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. (Cited page 19.)

[BKR14] Ulrich Bauer, Michael Kerber, and Jan Reininghaus. Distributed computation of persistent homology. In *Algorithm Engin. and Exp.*, 2014. (Cited pages 3, 26, 33, and 58.)

[BKRW17] Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. Phat - persistent homology algorithms toolbox. *J. Symb. Comput.*, 2017. https://bitbucket.org/phat-code/phat/src/master/. (Cited page 33.)

[BLM+19] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3703–3712, 2019. (Cited page 59.)

[BLW12] Ulrich Bauer, Carsten Lange, and Max Wardetzky. Optimal Topological Simplification of Discrete Functions on Surfaces. *Discrete Computational Geometry*, 2012. (Cited pages 27 and 28.)

[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 26, 37, and 66.)

[BNP+21] Nick Brown, Rupert Nash, Piero Poletti, Giorgio Guzzetta, Mattia Manica, Agnese Zardini, Markus Flatken, Jules Vidal, Charles Gueunet, Evgenij Belikov, Julien Tierny, Artur Podobas, Wei Der Chien, Stefano Markidis, and Andreas Gerndt. Utilising urgent computing to tackle the spread of mosquito-borne diseases. In *IEEE/ACM UrgentHPC@SC*, 2021. (Cited pages 1, 56, 58, 66, and 73.)

[BRLP19] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola. TTHRESH: Tensor compression for multidimensional visual data. *IEEE TVCG*, 2019. (Cited pages 1 and 56.)

[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 TVCG*, 17(9):1307–1324, 2011. (Cited pages 1, 3, 26, 56, 57, 58, 66, and 73.)

[Car04] Hamish Carr. *Topological Manipulation of Isosurfaces*. PhD thesis, University of British Columbia, 2004. (Cited page 27.)

[CCG+21] Mathieu Carrière, Frédéric Chazal, Marc Glisse, Yuichi Ike, Hariprasad Kannan, and Yuhei Umeda. Optimizing persistent homology based functions. In *ICML*, 2021. (Cited pages 17, 18, 27, 29, 31, 32, 38, 58, and 65.)

[CCL+22] Zeyuan Chen, Yinbo Chen, Jingwen Liu, Xingqian Xu, Vidit Goel, Zhangyang Wang, Humphrey Shi, and Xiaolong Wang. Videoinr: Learning video implicit neural representation for continuous space-time super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2047–2057, 2022. (Cited pages 59 and 60.)

[CCO17] Mathieu Carrière, Marco Cuturi, and Steve Oudot. Sliced wasserstein kernel for persistence diagrams. In *ICML*, 2017. (Cited page 33.)

[CEM06] David Cohen-Steiner, Herbert Edelsbrunner, and Dmitriy Morozov. Vines and vineyards by updating persistence in linear time. In *Symposium on Computational Geometry*, 2006. (Cited page 35.)

[CHW+21] Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser Nam Lim, and Abhinav Shrivastava. Nerv: Neural representations for videos. *Advances in Neural Information Processing Systems*, 34:21557–21568, 2021. (Cited pages 59 and 61.)

[CJL+18] Erin W. Chambers, Tao Ju, David Letscher, Mao Li, Christopher N. Topp, and Yajie Yan. Some heuristics for the homological simplification problem. In Stephane Durocher and Shahin Kamali, editors, *CCCG*, pages 353–359, 2018. (Cited page 47.)

[CKH+20] Myungsub Choi, Heewon Kim, Bohyung Han, Ning Xu, and Kyoung Mu Lee. Channel attention is all you need for video

frame interpolation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 10663–10671, 2020. (Cited page 59.)

[CSA00] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. In *Symp. on Dis. Alg.*, 2000. (Cited pages 3, 26, and 58.)

[CSvdP04] Hamish A. Carr, Jack Snoeyink, and Michiel van de Panne. Simplifying Flexible Isosurfaces Using Local Geometric Measures. In *IEEE VIS*, 2004. (Cited pages 26 and 30.)

[CWSA16] Hamish Carr, Gunther Weber, Christopher Sewell, and James Ahrens. Parallel peak pruning for scalable SMP contour tree computation. In *IEEE LDAV*, 2016. (Cited pages 3 and 26.)

[DDKL20] Damek Davis, Dmitriy Drusvyatskiy, Sham M. Kakade, and Jason D. Lee. Stochastic Subgradient Method Converges on Tame Functions. *Found. Comput. Math.*, 2020. (Cited pages 18, 29, and 65.)

[DFI+15] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015. (Cited page 59.)

[DOD23] Jiong Dong, Kaoru Ota, and Mianxiong Dong. Video frame interpolation: A comprehensive survey. *ACM Transactions on Multimedia Computing, Communications and Applications*, 19(2s):1–31, 2023. (Cited page 59.)

[EH09] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2009. (Cited pages 1, 3, 13, 14, 26, 56, and 57.)

[ELZ02] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological Persistence and Simplification. *Discrete Computational Geometry*, 28(4):511–533, 2002. (Cited pages 3, 13, 26, 30, 35, and 58.)

[EM90] Herbert Edelsbrunner and Ernst P Mucke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, 1990. (Cited page 13.)

[EMP06] Herbert Edelsbrunner, Dmitriy Morozov, and Valerio Pascucci. Persistence-sensitive simplification functions on 2-manifolds. In *Symposium on Computational Geometry*, 2006. (Cited pages 27 and 28.)

[FL99] P. Frosini and C. Landi. Size theory as a topological tool for computer vision. *Pattern Recognition and Image Analysis*, 1999. (Cited page 13.)

[For98] Robin Forman. A User's Guide to Discrete Morse Theory. *AM*, 1998. (Cited pages 28, 35, 43, and 44.)

[FPF+23] Markus Flatken, Artur Podobas, Riccardo Fellegara, Achim Basermann, Johannes Holke, David Knapp, Max Kontak, Christian Krullikowski, Michael Nolde, Nick Brown, Rupert Nash, Gordon Gibb, Evgenij Belikov, Steven W D Chien, Stefano Markidis, Pierre Guillou, Julien Tierny, Jules Vidal, Charles Gueunet, Johannes Günther, Miroslaw Pawlowski, Piero Poletti, Giorgio Guzzetta, Mattia Manica, Agnese Zardini, Jean-Pierre Chaboureau, Miguel Mendes, Adrián Cardil, Santiago Monedero, Joaquin Ramirez, and Andreas Gerndt. VESTEC: Visual Exploration and Sampling Toolkit for Extreme Computing. *IEEE Access*, 2023. (Cited pages 1, 56, 58, 66, and 73.)

[GAG+17] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *ICML*, 2017. (Cited page 62.)

[GBA+20] Yuyu Guo, Lei Bi, Euijoon Ahn, Dagan Feng, Qian Wang, and Jinman Kim. A spatiotemporal volumetric interpolation network for 4d dynamic medical image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4726–4735, 2020. (Cited page 60.)

[GBG+14] A. Gyulassy, P.T. Bremer, R. Grout, H. Kolla, J. Chen, and V. Pascucci. Stability of dissipation elements: A case study

in combustion. *Computer Graphics Forum*, 33(3):51–60, 2014. (Cited pages 3, 26, and 57.)

[GBHP08] A. Gyulassy, P. T. Bremer, B. Hamann, and V. Pascucci. A practical approach to Morse-Smale complex computation: Scalability and generality. *IEEE TVCG*, 2008. (Cited pages 3, 26, and 58.)

[GBHP09] Attila Gyulassy, Peer-Timo Bremer, Bernd Hamann, and Valerio Pascucci. Practical considerations in morse-smale complex computation. In *TopoInVis*. Springer, 2009. (Cited pages 27 and 49.)

[GBP19] Attila Gyulassy, Peer-Timo Bremer, and Valerio Pascucci. Shared-Memory Parallel Computation of Morse-Smale Complexes with Improved Accuracy. *IEEE TVCG*, 25(1):1183–1192, 2019. (Cited pages 3, 26, and 58.)

[GCW23] Pengfei Gu, Danny Z Chen, and Chaoli Wang. Nervi: Compressive neural representation of visualization images for communicating volume visualization results. *Computers & Graphics*, 116:216–227, 2023. (Cited page 60.)

[GDN+07] Attila Gyulassy, Mark A. Duchaineau, Vijay Natarajan, Valerio Pascucci, Eduardo Bringa, Andrew Higginbotham, and Bernd Hamann. Topologically Clean Distance Fields. *IEEE TVCG*, 13(6):1432–1439, 2007. (Cited page 44.)

[GFJT17] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Task-based Augmented Merge Trees with Fibonacci Heaps,. In *IEEE LDAV*, 2017. (Cited pages 3 and 26.)

[GFJT19a] Charles Gueunet, Pierre Fortin, Julien Jomier, and Julien Tierny. Task-Based Augmented Contour Trees with Fibonacci Heaps. *IEEE TPDS*, 30(8):1889–1905, 2019. (Cited pages 3, 26, and 58.)

[GFJT19b] Charles Gueunet, Pierre Fortin, Julien Jomier, and Julien Tierny. Task-based Augmented Reeb Graphs with Dynamic ST-Trees. In *EGPGV*, 2019. (Cited pages 3, 26, and 58.)

[GGSG20] Rickard Brüel Gabrielsson, Vignesh Ganapathi-Subramanian, Primoz Skraba, and Leonidas J. Guibas. Topology-Aware Sur-

face Reconstruction for Point Clouds. *Computer Graphics Forum*, 2020. (Cited pages 29 and 58.)

[Gir15] Ross B. Girshick. Fast R-CNN. In *ICCV*, 2015. (Cited page 64.)

[GJR+14] David Günther, Alec Jacobson, Jan Reininghaus, Hans-Peter Seidel, Olga Sorkine-Hornung, and Tino Weinkauf. Fast and Memory-Efficienty Topological Denoising of 2D and 3D Scalar Fields. *IEEE TVCG*, 2014. (Cited page 28.)

[GKL+16] A. Gyulassy, A. Knoll, K.C. Lau, B. Wang, P.T. Bremer, M.E. Papka, L. A. Curtiss, and V. Pascucci. Interstitial and Interlayer Ion Diffusion Geometry Extraction in Graphitic Nanosphere Battery Materials. *IEEE TVCG*, 22(1):916–925, 2016. (Cited pages 3 and 26.)

[GND+07] Attila Gyulassy, Vijay Natarajan, Mark Duchaineau, Valerio Pascucci, Eduardo Bringa, Andrew Higginbotham, and Bernd Hamann. Topologically Clean Distance Fields. *IEEE TVCG*, 2007. (Cited page 57.)

[GNP+05] Attila Gyulassy, Vijay Natarajan, Valerio Pascucci, Peer-Timo Bremer, and Bernd Hamann. Topology-based simplification for feature extraction from 3d scalar fields. In *VIS*, pages 535–542. IEEE, 2005. (Cited page 44.)

[GRF24] Hamid Gadirov, Jos BTM Roerdink, and Steffen Frey. Flint: Learning-based flow estimation and temporal interpolation for scientific ensemble visualization. *arXiv preprint arXiv:2409.19178*, 2024. (Cited page 60.)

[GRSW13] David Günther, Jan Reininghaus, Hans-Peter Seidel, and Tino Weinkauf. Notes on the simplification of the morse-smale complex. In *TopoInVis*. Springer, 2013. (Cited pages 27, 44, and 49.)

[GVT23] Pierre Guillou, Jules Vidal, and Julien Tierny. Discrete Morse Sandwich: Fast Computation of Persistence Diagrams for Scalar Data – An Algorithm and A Benchmark. *IEEE TVCG*, 2023. (Cited pages 3, 13, 14, 25, 26, 33, 35, 36, 38, 43, 48, 49, 58, and 64.)

[Gyu08] Attila Gyulassy. *Combinatorial construction of Morse-Smale complexes for data analysis and visualization*. PhD thesis, UC Davis, 2008. (Cited pages 26 and 27.)

[GZ07] Yotam I. Gingold and Denis Zorin. Controlled-topology filtering. *Comput. Aided Des.*, 2007. (Cited page 28.)

[H. 42] H. Freudenthal. Simplizialzerlegungen von beschrankter Flachheit. *Annals of Mathematics*, 43:580–582, 1942. (Cited page 13.)

[HBMK22] Ping Hu, Saeed Boorboor, Joseph Marino, and Arie E. Kaufman. Geometry-aware planar embedding of treelike structures. *IEEE TVCG*, 2022. (Cited pages 43 and 44.)

[HLH+16] C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. De Floriani, G. Scheuermann, H. Hagen, and C. Garth. A survey of topology-based methods in visualization. *Computer Graphics Forum*, 35(3):643–667, 2016. (Cited pages 1, 3, 26, 56, and 57.)

[H.W60] H.W. Kuhn. Some combinatorial lemmas in topology. *IBM Journal of Research and Development*, 45:518–524, 1960. (Cited page 13.)

[HW19] Jun Han and Chaoli Wang. Tsr-tvd: Temporal super-resolution for time-varying data analysis and visualization. *IEEE transactions on visualization and computer graphics*, 26(1):205–215, 2019. (Cited page 60.)

[HW22] Jun Han and Chaoli Wang. Coordnet: Data generation and visualization generation for time-varying volumes via a coordinate-based neural network. *IEEE Transactions on Visualization and Computer Graphics*, 29(12):4951–4963, 2022. (Cited page 60.)

[HZB23] Jun Han, Hao Zheng, and Chongke Bi. Kd-inr: Time-varying volumetric data compression via knowledge distillation-based implicit neural representation. *IEEE Transactions on Visualization and Computer Graphics*, 30(10):6826–6838, 2023. (Cited page 60.)

[HZCW21] Jun Han, Hao Zheng, Danny Z Chen, and Chaoli Wang. Stnet: An end-to-end generative framework for synthesizing

spatiotemporal super-resolution volumes. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):270–280, 2021. (Cited page 60.)

[HZH+22] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Real-time intermediate flow estimation for video frame interpolation. In *European Conference on Computer Vision*, pages 624–642. Springer, 2022. (Cited pages 59 and 60.)

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE CVPR*, 2016. (Cited pages 20, 61, and 63.)

[IFF15] Federico Iuricich, Ulderico Fugacci, and Leila De Floriani. Topologically-consistent simplification of discrete morse complex. *Comput. Graph.*, 51:157–166, 2015. (Cited page 44.)

[IMS+17] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017. (Cited page 59.)

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 448–456. PMLR, 2015. (Cited page 19.)

[Iur21] Federico Iuricich. Persistence cycles for visual exploration of persistent homology. *IEEE TVCG*, 2021. https://github.com/IuricichF/PersistenceCycles. (Cited pages 25, 43, 48, and 49.)

[IZZE17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *IEEE CVPR*, 2017. (Cited page 63.)

[JBY24] Chenyue Jiao, Chongke Bi, and Lu Yang. Ffeinr: flow feature-enhanced implicit neural representation for spatiotemporal super-resolution. *Journal of Visualization*, 27(2):273–289, 2024. (Cited page 60.)

[JSJ+18] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High

quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9000–9008, 2018. (Cited page 59.)

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. (Cited pages 18, 29, 32, 33, 38, 42, 62, and 66.)

[Kla20] Pavol Klacansky. Open Scientific Visualization Data Sets. https://klacansky.com/open-scivis-datasets/, 2020. (Cited page 38.)

[KMN17] Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams. *ACM J. of Experimental Algorithmics*, 22, 2017. (Cited pages 33 and 38.)

[KPCT23] Tarun Kalluri, Deepak Pathak, Manmohan Chandraker, and Du Tran. Flavr: Flow-agnostic video representations for fast frame interpolation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2071–2082, 2023. (Cited page 59.)

[KPLT24] Mohamed Kissi, Mathieu Pont, Joshua Aaron Levine, and Julien Tierny. A Practical Solver for Scalar Data Topological Simplification. *IEEE TVCG*, 2024. (Cited pages 24, 58, 72, and 73.)

[KRHH11] J. Kasten, J. Reininghaus, I. Hotz, and H.C. Hege. Two-dimensional time-dependent vortex regions based on the acceleration magnitude. *IEEE TVCG*, 17(12):2080–2087, 2011. (Cited pages 3, 26, and 57.)

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. (Cited page 18.)

[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 2, 56, 58, and 74.)

[LGMT20] Jonas Lukasczyk, Christoph Garth, Ross Maciejewski, and Julien Tierny. Localized topological simplification of scalar data. *IEEE TVCG*, 2020. (Cited pages 27 and 28.)

[LGW24] Yunfei Lu, Pengfei Gu, and Chaoli Wang. Fcnr: Fast compressive neural representation of visualization images. In *2024 IEEE Visualization and Visual Analytics (VIS)*, pages 31–35. IEEE, 2024. (Cited page 60.)

[Lin14] P. Lindstrom. Fixed-Rate Compressed Floating-Point Arrays. *IEEE TVCG*, 2014. (Cited pages 1 and 56.)

[LJLB21] Yuzhe Lu, Kairong Jiang, Joshua A Levine, and Matthew Berger. Compressive neural representations of volumetric scalar fields. In *Computer Graphics Forum*, volume 40, pages 135–146. Wiley Online Library, 2021. (Cited page 60.)

[LK81] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, volume 2, pages 674–679, 1981. (Cited page 59.)

[LKC+20] Hyeongmin Lee, Taeoh Kim, Tae-young Chung, Daehyun Pak, Yuseok Ban, and Sangyoun Lee. Adacof: Adaptive collaboration of flows for video frame interpolation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5316–5325, 2020. (Cited page 59.)

[LLW+25] Yuxiao Li, Xin Liang, Bei Wang, Yongfeng Qiu, Lin Yan, and Hanqi Guo. Msz: An efficient parallel algorithm for correcting morse-smale segmentations in error-bounded lossy compressors. *IEEE TVCG*, 2025. (Cited page 58.)

[LN24] Yuan Luo and Bradley J. Nelson. Accelerating iterated persistent homology computations with warm starts. *Computational Geometry*, 2024. (Cited page 35.)

[LWW+24] Jonas Lukasczyk, Michael Will, Florian Wetzels, Gunther H. Weber, and Christoph Garth. ExTreeM: Scalable Augmented Merge Tree Computation via Extremum Graphs. *IEEE TVCG*, 2024. (Cited pages 3, 26, and 58.)

[LXS+20] Yihao Liu, Liangbin Xie, Li Siyao, Wenxiu Sun, Yu Qiao, and Chao Dong. Enhanced quadratic video interpolation. In *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 41–56. Springer, 2020. (Cited page 59.)

[MBGY14] Clément Maria, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The gudhi library: Simplicial complexes and persistent homology. In *Mathematical Software*, 2014. https://github.com/GUDHI/. (Cited page 38.)

[MK16] Joseph Marino and Arie E. Kaufman. Planar visualization of treelike structures. *IEEE TVCG*, 2016. (Cited pages 43 and 44.)

[MLT+23] Robin G. C. Maack, Jonas Lukasczyk, Julien Tierny, Hans Hagen, Ross Maciejewski, and Christoph Garth. Parallel computation of piecewise linear morse-smale segmentations. *IEEE TVCG*, 2023. (Cited pages 3 and 26.)

[Mun57] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957. (Cited page 33.)

[MWR+16] Daniel Maljovec, Bei Wang, Paul Rosen, Andrea Alfonsi, Giovanni Pastore, Cristian Rabiti, and Valerio Pascucci. Topology-inspired partition-based sensitivity analysis and visualization of nuclear simulations. In *IEEE PacificViz*, 2016. (Cited pages 3 and 26.)

[MWZ+15] Simone Meyer, Oliver Wang, Henning Zimmer, Max Grosse, and Alexander Sorkine-Hornung. Phase-based frame interpolation for video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1410–1418, 2015. (Cited page 59.)

[NGH04] Xinlai Ni, Michael Garland, and John C. Hart. Fair morse functions for extracting the topological structure of a surface mesh. *ACM Transactions on Graphics*, 2004. (Cited page 28.)

[NKSM24] Arnur Nigmetov, Aditi S. Krishnapriyan, Nicole Sanderson, and Dmitriy Morozov. Topological regularization via persistence-sensitive optimization. *Computational Geometry*, 2024. (Cited page 28.)

[NL20] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5437–5446, 2020. (Cited page 59.)

[NM22] Arnur Nigmetov and Dmitriy Morozov. Topological optimization with big steps. *CoRR*, abs/2203.16748, 2022. (Cited pages 27, 29, 38, and 58.)

[NML17] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE international conference on computer vision*, pages 261–270, 2017. (Cited page 59.)

[NVBB+22] Florent Nauleau, Fabien Vivodtzev, Thibault Bridel-Bertomeu, Heloise Beaugendre, and Julien Tierny. Topological Analysis of Ensembles of Hydrodynamic Turbulent Flows – An Experimental Study. In *IEEE Symposium on Large Data Analysis and Visualization*, 2022. (Cited pages 3, 26, and 57.)

[OGT19] Malgorzata Olejniczak, André Severo Pereira Gomes, and Julien Tierny. A Topological Data Analysis Perspective on Non-Covalent Interactions in Relativistic Calculations. *International Journal of Quantum Chemistry*, 120(8):e26133, 2019. (Cited pages 3 and 26.)

[OT23] Malgorzata Olejniczak and Julien Tierny. Topological Data Analysis of Vortices in the Magnetically-Induced Current Density in LiH Molecule. *Physical Chemistry Chemical Physics*, 2023. (Cited pages 3, 26, and 57.)

[PF09] Giuseppe Patanè and Bianca Falcidieno. Computing smooth approximations of scalar functions with constraints. *Comput. Graph.*, 2009. (Cited page 28.)

[PG18] John Patchett and Galen Ross Gisler. The IEEE SciVis Contest. http://sciviscontest.ieeevis.org/2018/, 2018. (Cited page 67.)

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019. https://pytorch.org/cppdocs/. (Cited pages 38, 62, and 66.)

[PKLK20] Junheum Park, Keunsoo Ko, Chul Lee, and Chang-Su Kim. Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 109–125. Springer, 2020. (Cited page 59.)

[PLK21] Junheum Park, Chul Lee, and Chang-Su Kim. Asymmetric bilateral motion estimation for video frame interpolation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14539–14548, 2021. (Cited page 59.)

[PSBM07] V Pascucci, G Scorzelli, P T Bremer, and A Mascarenhas. Robust on-line computation of Reeb graphs: simplicity and speed. *ACM Transactions on Graphics*, 26(3):58, 2007. (Cited pages 3, 26, 27, and 58.)

[PSO18] Adrien Poulenard, Primoz Skraba, and Maks Ovsjanikov. Topological function optimization for continuous shape matching. *Computer Graphics Forum*, 2018. (Cited pages 29, 33, and 58.)

[PT24] Mathieu Pont and Julien Tierny. Wasserstein Auto-Encoders of Merge Trees (and Persistence Diagrams). *IEEE TVCG*, 2024. (Cited pages 2, 56, 58, 74, and 82.)

[PVDT22] Mathieu Pont, Jules Vidal, Julie Delon, and Julien Tierny. Wasserstein Distances, Geodesics and Barycenters of Merge Trees. *IEEE TVCG*, 28(1):291–301, 2022. https://github.com/MatPont/WassersteinMergeTreesData. (Cited pages 2, 56, 58, and 74.)

[PVT23] Mathieu Pont, Jules Vidal, and Julien Tierny. Principal Geodesic Analysis of Merge Trees (and Persistence Diagrams). *IEEE TVCG*, 2023. (Cited pages 2, 56, 58, 74, and 82.)

[RKT+22] Fitsum Reda, Janne Kontkanen, Eric Tabellion, Deqing Sun, Caroline Pantofaru, and Brian Curless. Film: Frame interpolation for large motion. In *European Conference on Computer Vision*, pages 250–266. Springer, 2022. (Cited page 59.)

[Rob99] Vanessa Robins. Toward computing homology from finite approximations. *Topology Proceedings*, 1999. (Cited page 13.)

[RWS11] Vanessa Robins, Peter John Wood, and Adrian P. Sheppard. Theory and Algorithms for Constructing Discrete Morse Complexes from Grayscale Digital Images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(8):1646–1658, 2011. (Cited pages 3, 14, 26, 35, 36, and 58.)

[SDT23] Keanu Sisouk, Julie Delon, and Julien Tierny. Wasserstein Dictionaries of Persistence Diagrams. *CoRR*, 2023. (Cited page 82.)

[SN12] Nithin Shivashankar and Vijay Natarajan. Parallel Computation of 3D Morse-Smale Complexes. *Computer Graphics Forum*, 31(3):965–974, 2012. (Cited pages 3, 26, and 58.)

[Soi04] Pierre Soille. Optimal Removal of Spurious Pits in Digital Elevation Models. *Water Resources Research*, 2004. (Cited pages 27 and 28.)

[Sou11] T. Sousbie. The Persistent Cosmic Web and its Filamentary Structure: Theory and Implementations. *Royal Astronomical Society*, 414:384–403, 2011. (Cited pages 3, 26, 45, 46, and 57.)

[SPCT18] Maxime Soler, Mélanie Plainchault, Bruno Conche, and Julien Tierny. Topologically controlled lossy compression. In *IEEE PacificViz*, 2018. (Cited page 58.)

[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 LDAV*, 2019. (Cited pages 3, 26, and 57.)

[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 TVCG*, 22(6):1745–1759, 2016. (Cited pages 3, 26, 45, 46, and 57.)

[SWB21] Elchanan Solomon, Alexander Wagner, and Paul Bendich. A fast and robust method for global topological functional optimization. In *AISTATS*, 2021. (Cited pages 27, 29, and 58.)

[SYLK18] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and

cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018. (Cited page 59.)

[TD20] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020. (Cited page 59.)

[TFL+17] Julien Tierny, Guillaume Favelier, Joshua A. Levine, Charles Gueunet, and Michael Michaux. The Topology ToolKit. *IEEE TVCG*, 24(1):832–842, 2017. https://topology-tool-kit.github.io/. (Cited pages 24, 26, 37, and 66.)

[TMMH14] Katharine Turner, Yuriy Mileyko, Sayan Mukherjee, and John Harer. Fréchet Means for Distributions of Persistence Diagrams. *Discrete Computational Geometry*, 52(1):44–70, 2014. (Cited pages 2, 56, 58, and 74.)

[TP12] Julien Tierny and Valerio Pascucci. Generalized topological simplification of scalar fields on surfaces. *IEEE TVCG*, 2012. (Cited pages 27 and 28.)

[TTK20] TTK Contributors. TTK Data. https://github.com/topology-tool-kit/ttk-data/tree/dev, 2020. (Cited pages 38, 49, and 67.)

[TTK22] TTK Contributors. TTK Online Example Database. https://topology-tool-kit.github.io/examples/, 2022. (Cited page 26.)

[TW24] Kaiyuan Tang and Chaoli Wang. Stsr-inr: Spatiotemporal super-resolution for multivariate time-varying volumetric data via implicit neural representation. *Computers & Graphics*, 119:103874, 2024. (Cited pages 60, 68, 70, 71, and 72.)

[UVL16] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. (Cited pages 19 and 63.)

[VBT20] Jules Vidal, Joseph Budin, and Julien Tierny. Progressive Wasserstein Barycenters of Persistence Diagrams. *IEEE TVCG*, 26(1):151–161, 2020. (Cited pages 2, 33, 56, 58, and 74.)

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. (Cited page 62.)

[WBCM23] Qi Wu, David Bauer, Yuyang Chen, and Kwan-Liu Ma. Hyperinr: A fast and predictive hypernetwork for implicit neural representations via knowledge distillation. *arXiv preprint arXiv:2304.04188*, 2023. (Cited page 60.)

[WBKS04] Wei Wang, Cindy Bruyere, Bill Kuo, and Tim Scheitlin. The IEEE SciVis Contest. http://sciviscontest.ieeevis.org/2004/, 2004. (Cited page 67.)

[WGS10] Tino Weinkauf, Yotam I. Gingold, and Olga Sorkine. Topology-based Smoothing of 2D Scalar Fields with C^1-Continuity. *Computer Graphics Forum*, 2010. (Cited page 28.)

[WLC+21] Zejin Wang, Jing Liu, Xi Chen, Guoqing Li, and Hua Han. Sparse self-attention aggregation networks for neural sequence slice interpolation. *BioData Mining*, 14:1–19, 2021. (Cited page 60.)

[WWY+20] Zhaotao Wu, Jia Wei, Wenguang Yuan, Jiabing Wang, and Tolga Tasdizen. Inter-slice image augmentation based on frame interpolation for boosting medical image segmentation accuracy. In *ECAI 2020*, pages 1954–1961. IOS Press, 2020. (Cited page 60.)

[WZZ+13] Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013. (Cited page 65.)

[YLW+21] Xiao-lei Yin, Dong-xue Liang, Lu Wang, Jing Qiu, Zhi-yun Yang, Jian-zeng Dong, and Zhao-yuan Ma. Analysis of coronary angiography video interpolation methods to reduce x-ray exposure frequency based on deep learning. *Cardiovascular Innovations and Applications*, 6(1):17, 2021. (Cited page 60.)

[YWM+19] Lin Yan, Yusu Wang, Elizabeth Munch, Ellen Gasparovic, and Bei Wang. A structural average of labeled merge trees for uncertainty visualization. *IEEE TVCG*, 26(1):832–842, 2019. (Cited pages 2, 56, 58, and 74.)

[ZCLJ20] Dan Zeng, Erin W. Chambers, David Letscher, and Tao Ju. To cut or to fill: a global optimization approach to topological simplification. *ACM Transactions on Graphics*, 39(6):201:1–201:18, 2020. (Cited page 47.)

[ZCLJ22] Dan Zeng, Erin W. Chambers, David Letscher, and Tao Ju. Topological simplification of nested shapes. *Computer Graphics Forum*, 41(5):161–173, 2022. (Cited page 47.)

[Zom10] Afra J. Zomorodian. Topology for computing. In Mikhail J. Atallah and Marina Blanton, editors, *Algorithms and Theory of Computation Handbook (Second Edition)*, chapter 3, pages 82–112. CRC Press, 2010. (Cited pages 3 and 26.)

Persistence optimization for data visualization

This thesis addresses the challenge of managing the growing complexity of scalar data in both space and time, by leveraging topological methods for data reduction and reconstruction. As datasets become increasingly large and detailed, topological descriptors such as persistence diagrams offer compact and robust summaries that support efficient storage and analysis. However, these abstract representations often lack the necessary information for detailed visualization or interpretation, highlighting the need for methods capable of reconstructing scalar data from them. To this end, the thesis introduces two complementary contributions. First, it proposes an optimization framework for topological simplification that preserves meaningful features while removing noise, extending prior approaches by handling complex structures such as saddle pairs in three-dimensional data. Second, it presents a neural interpolation scheme that reconstructs intermediate scalar fields from sparse keyframes, guided by topological constraints. By incorporating topology-aware loss functions, the model improves both geometric and topological accuracy, enabling the faithful reconstruction of time-varying scalar data. These contributions are validated on synthetic and real-world datasets, including applications in medical imaging and meteorology, and are supported by open-source implementations to ensure reproducibility. Together, they provide practical tools for simplifying and reconstructing complex scalar data through a topological lens.

Optimisation de la persistance pour la visualisation de données

Cette thèse aborde le défi de la gestion de la complexité croissante des champs scalaires dans l'espace et le temps, en s'appuyant sur des méthodes topologiques pour la réduction et la reconstruction des données. À mesure que les jeux de données deviennent plus volumineux et plus détaillés, les descripteurs topologiques, tels que les diagrammes de persistance, offrent des résumés compacts et robustes qui facilitent le stockage et l'analyse. Toutefois, ces représentations abstraites ne contiennent souvent pas suffisamment d'informations pour permettre une visualisation ou une interprétation détaillée, ce qui met en évidence la nécessité de méthodes capables de reconstruire les données scalaires à partir de ces descripteurs. Pour répondre à ce besoin, la thèse propose deux contributions complémentaires. Premièrement, elle introduit un cadre d'optimisation pour la simplification topologique, permettant de préserver les structures significatives tout en éliminant le bruit, et qui étend les approches existantes en prenant en charge des structures complexes comme les paires selles dans des données tridimensionnelles. Deuxièmement, elle présente un schéma d'interpolation neuronal permettant de reconstruire des champs scalaires intermédiaires à partir de keyframes espacés, en s'appuyant sur des contraintes topologiques. En intégrant des fonctions de perte sensibles à la topologie, le modèle améliore à la fois la fidélité géométrique et topologique des reconstructions, permettant une interpolation fidèle de données scalaires évoluant dans le temps. Ces contributions sont validées sur des jeux de données synthétiques et réels, incluant des applications en imagerie médicale et en météorologie, et sont accompagnées d'implémentations open-source assurant la reproductibilité. Ensemble, elles fournissent des outils pratiques pour la simplification et la reconstruction de données scalaires complexes à travers une approche guidée par la topologie.