Homology-Preserving Dimensionality Reduction via Adaptive Mapper and Landmark Isomap

Shakiba Khourashahi, Ilia Jahanshahi, Bei Wang, Lin Yan

arXiv:2606.04464v1 [cs.CG] 3 Jun 2026

***Abstract*—As data becomes increasingly central across engineering and scientific disciplines, effective visualization is essential for interpreting complex, high-dimensional structures. Dimensionality reduction techniques project high-dimensional data into lower dimensions while aiming to preserve structural properties such as pairwise distances and local neighborhoods. In this paper, we focus on improving *homological preservation*, that is, the retention of topological features such as connected components and loops, which is critical for maintaining global shape and continuity. We first introduce AdaMapper, a Mapper-based algorithm that leverages persistence diagrams to guide both skeleton construction and landmark selection. AdaMapper incorporates an adaptive refinement strategy that automatically increases cover resolution in regions exhibiting topological loops. We then propose AdaHIsomap, which extends landmark Isomap by incorporating homology-informed landmark selection and augmenting it with random anchor points to better balance distance and homology preservation. We evaluate both methods on a diverse set of datasets, including high-dimensional point clouds, scientific simulations, networks, and image data, and benchmark their performance against state-of-the-art approaches.**

***Index Terms*—Dimensionality reduction, Mapper, topology-based visualization, high-dimensional data visualization.**

I. Introduction

High-dimensional datasets are increasingly common across domains such as economics [1], biology [2], physics [3], and chemistry [4]. Visualization provides a critical means to uncover and interpret patterns in such data, with dimensionality reduction (DR) being one of the most widely used approaches.

DR techniques can be broadly grouped into linear and nonlinear methods. Linear methods such as principal component analysis (PCA) [5], multidimensional scaling (MDS) [6], and Fisher's linear discriminant (FLD) [7] are valued for their simplicity and employ global linear projections. Nonlinear techniques, often associated with manifold learning, capture more complex structures through either metric-based graph constructions (e.g., Isomap [8], locally linear embedding (LLE) [9], Laplacian eigenmaps [10]) or nonmetric optimizations that preserve dissimilarity rankings [11]. Beyond this distinction, methods are also categorized by the structural properties they preserve. While distance preservation ensures geometric fidelity (e.g., PCA [5], MDS [6]), topology preservation emphasizes neighborhood relations (e.g., Isomap [8], LLE [9]). In this paper, we focus on a specific aspect of topological preservation—*homology preservation*—which seeks to preserve 0-dimensional (0D, connected components) and 1-dimensional (1D, loop) features.

Homology-preserving DR is particularly valuable for datasets exhibiting nontrivial topological structure. Persistent homology has been applied across a wide range of domains to analyze high-dimensional point clouds, including biology [12] and astronomy [13]. For instance, it has been used to identify statistically significant cosmic voids by modeling galaxies as point clouds [14]. In image analysis, Carlsson et al. [15] and Xia et al. [16] showed that subspaces of natural image patches can exhibit circular structures or even be topologically equivalent to a Klein bottle, depending on patch size. In scientific simulations, topological descriptors such as merge trees and Morse complexes have been used to uncover structural patterns, including clusters, outliers, and periodic behavior. For example, Yan et al. [17] characterized periodicity in a vortex street dataset by analyzing merging and splitting behavior via merge trees, while Lan et al. [18] employed 1-dimensional topological skeletons derived from Morse complexes to study algorithmic uncertainties in atmospheric river detection. A recent survey [19] provides a broader overview of the integration of homological features into scientific visualization. In complex network science, persistent homology also serves as a fundamental tool for abstraction and summarization, with applications spanning biological and social networks; see [20] for a comprehensive survey.

Compared with other topology-preserving methods, t-SNE and UMAP have limited control over global structure. The KL (Kullback–Leibler) objective in t-SNE ignores large-distance relationships—points that are far apart in the original space can be placed close together without significant penalty [21]. UMAP [22], in contrast, optimizes edge-level similarities of a fuzzy k-nearest neighbor (kNN) graph; however, it does not explicitly preserve end-to-end geodesic distances, and long-range relationships are only indirectly constrained. As a result, both methods may introduce topological artifacts, such as merging or tearing, depending on hyperparameter choices.

In contrast, HIsomap, the manifold-landmarking approach of Yan et al. [23], promotes homology preservation by combining landmark Isomap (L-Isomap) with homology-informed landmark selection. Specifically, it selects landmarks from nodes in the Mapper graph, thereby reducing computational complexity while improving the quality and stability of the resulting embedding.

Despite these advantages, HIsomap has two key limitations: (1) sensitivity to parameter choices (its performance depends on five parameters), which can hinder its practical adoption in scientific data analysis, and (2) a globally fixed Mapper

Shakiba Khourashahi is with Iowa State University. E-mail: shakiba@iastate.edu
Ilia Jahanshahi is with Iowa State University. E-mail: ilia@iastate.edu
Bei Wang is with University of Utah. E-mail: beiwang@sci.utah.edu
Lin Yan is with Iowa State University. E-mail: linyan@iastate.edu

resolution that assumes uniform topological complexity along the filter function, often leading to oversampling in non-critical regions and undersampling in regions containing loops. To address these limitations, we propose an adaptive landmarking strategy that extends manifold-landmarking-based DR to more robustly preserve 0D and 1D homological features. Our goal is to provide a general framework for more accurate and insightful data visualization. Our contributions are as follows:

- AdaMapper: A Mapper-based skeletonization method that derives locally adaptive cover resolutions directly from the 1D persistence diagram, refining loop-critical regions while coarsening non-critical ones. This approach overcomes the limitations of globally fixed Mapper resolutions without extensive manual tuning.
- AdaHIsomap: Building on AdaMapper, we propose AdaHIsomap, which integrates persistence-guided adaptive landmark selection with stochastic anchor points to balance 0D and 1D homology preservation. The result is a topology-aware pipeline that leverages homological information to guide intermediate representations for DR.
- A systematic evaluation across diverse datasets—including high-dimensional point clouds, scientific simulations, networks, and images—demonstrating the behavior of the proposed pipeline under default settings and its competitive performance relative to state-of-the-art DR methods.

Our algorithms and datasets is released open source under the MIT license on GitHub (https://github.com/VisTALELab/AdaHIsomap.git).

II. Related Work

We review related work in two areas most relevant to our contributions: topology-aware skeletonization methods based on Mapper and structure-preserving DR techniques.

Mapper-based data skeletonization. The Mapper construction [24] captures the topological structure of a high-dimensional point cloud and summarizes it as a 1D skeleton graph. It has been widely adopted in topological data analysis and visualization, with applications spanning finance [25] and medical imaging [26], [27].

A central challenge of Mapper is its sensitivity to the choice of filter function, cover parameters, and clustering parameters. These user-specified parameters can significantly influence the resulting graph, complicating both interpretation and reproducibility. Motivated by its broad applicability, numerous variants have been proposed to improve robustness and extend Mapper to new domains [28]–[32].

Soft Mapper [28] addresses sensitivity to the filter function by introducing an optimization framework that selects filters automatically based on topological criteria. Other approaches focus on cover parameter selection. For example, multi-pass AIC/BIC [29], inspired by the X-means algorithm [33] (an extension of K-means [34]), constructs adaptive covers using information-theoretic model selection. F-Mapper [30] employs fuzzy C-means clustering [35] to allow overlapping clusters. More recently, Alvarado et al. proposed G-Mapper [31], which leverages G-means clustering with Gaussian mixture models (GMMs) and iterative splitting to eliminate the need for a priori interval counts, adapting cover elements to the underlying data distribution. Scoccola et al. [32] further advanced this direction by introducing a general cover-learning framework that adapts automatically to data-driven topological features.

These methods substantially improve the usability and stability of Mapper by automating filter or cover selection. However, they do not explicitly leverage persistent homology to control cover resolution locally along the filter function.

Compared with prior Mapper variants, AdaMapper (detailed in Sec. IV) derives key cover parameters directly from properties of 1D persistent homology, reducing reliance on user expertise while still allowing optional user control. Specifically, it introduces a locally adaptive cover that increases resolution in regions exhibiting 1D loops while avoiding unnecessary refinement elsewhere. As a result, AdaMapper produces more stable and informative Mapper skeletons and is more robust across datasets with heterogeneous topological structure.

Structure-preserving dimensionality reduction. Structure-preserving DR aims to recover intrinsic shapes in data, such as manifolds, spirals, or curved surfaces. UMAP [22] constructs a fuzzy kNN graph and learns a low-dimensional embedding that preserves local structure while approximating global relationships. t-SNE [21] is a probabilistic DR method that models local pairwise similarities and emphasizes neighborhood preservation, producing embeddings that are stable and visually interpretable. Another notable approach is Isomap [8], which captures manifold geometry using geodesic distances, that is, the shortest paths along the underlying manifold. This is particularly effective for data lying on nonlinear manifolds, where Euclidean distances can underestimate true separations between distant points and overestimate distances between points that are close along the manifold but far in ambient space. Landmark Isomap (L-Isomap) [36] addresses the computational cost of Isomap by selecting a subset of points as landmarks to approximate the distance matrix, thereby reducing complexity while largely preserving geometric fidelity.

More recently, DR techniques have been developed with an explicit focus on preserving topological features. TopoMap [37] preserves the 0D persistence diagram of the Vietoris–Rips filtration during embedding by leveraging the Euclidean minimum spanning tree and iteratively merging connected components to maintain topological structure. Building on this approach, TopoMap++ [38] improves computational efficiency and introduces an interactive framework for exploring the hierarchical organization of topological components. In parallel, a growing body of work incorporates topology-aware loss functions into machine learning models for homology-preserving DR. For example, TopoAE [39] introduces a topological loss based on discrepancies in 0D homology between the original and embedded spaces. This framework was recently extended to preserve 1D persistent homology by Clémot et al. [40], who proposed TopoAE++, a cycle-aware topological autoencoder.

The work most closely related to ours is that of Yan et al. [23], which is designed to preserve both 0D and 1D homological features. Referred to as HIsomap in this paper, the method builds on the landmark selection strategy of L-Isomap,

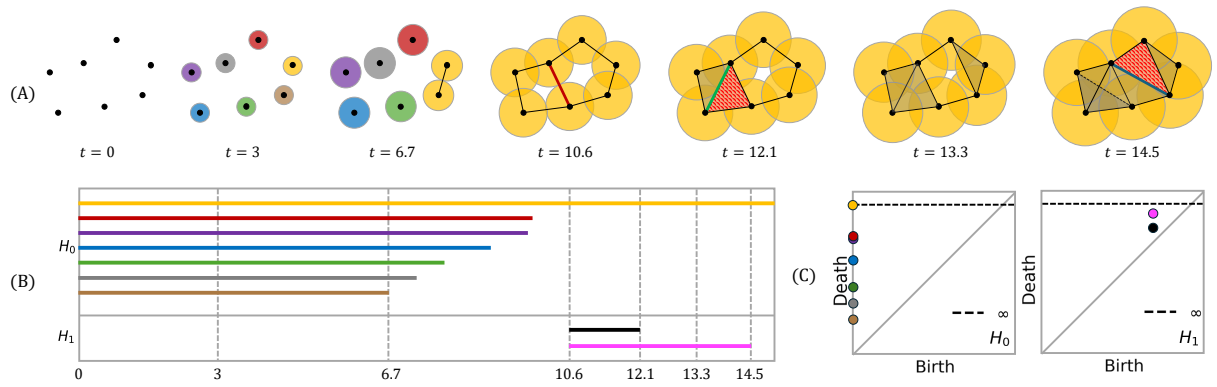

Fig. 1. Computing persistent homology of a point cloud. (A) Simplicial complexes constructed on seven points at increasing values of t. (B) Corresponding persistence barcodes. (C) Persistence diagrams for 0D (left) and 1D (right) features.

using a data skeleton derived from the Mapper graph [41] to guide the selection of representative landmarks on the manifold.

In this work, we adopt the landmark Isomap paradigm but reformulate the landmark selection strategy and associated pipeline components. Specifically, we use AdaMapper to derive landmarks from a persistence-guided, adaptively constructed skeleton, and we introduce random anchor points to balance distance preservation with 0D and 1D homology preservation in regions not associated with prominent loops. The resulting DR pipeline is evaluated against the structure-preserving methods discussed above in Sec. VI.

III. Technical Background

We review persistent homology, Mapper, and HIsomap, which form the foundations of our proposed methods. We then describe how these techniques are customized to develop AdaMapper (Sec. IV) and AdaHIsomap (Sec. V).

A. Persistent Homology

Homology provides a formal framework for distinguishing topological spaces based on their structural features, particularly the presence of "holes" across various dimensions. Mathematically, homology associates a topological space with a sequence of abelian groups, known as homology groups, which encode the number and type of topological features present in the space [42]. These groups capture essential information about the space's connectivity. Given a topological space $\mathbb{X}$, its homology groups $\mathrm{H}_0(\mathbb{X})$, $\mathrm{H}_1(\mathbb{X})$ and $\mathrm{H}_2(\mathbb{X})$ capture structural features of increasing dimension: $\mathrm{H}_0(\mathbb{X})$ represents connected components, $\mathrm{H}_1(\mathbb{X})$ represents loops or tunnels, and $\mathrm{H}_2(\mathbb{X})$ represents voids. By systematically quantifying these features, homology enables the comparison of topological spaces through their algebraic invariants. This characterization is foundational to the persistent homology framework, which extends classical homology to study how topological features evolve across different scales.

To compute homology from discrete data such as point clouds, we approximate the underlying space with a simplicial complex, a combinatorial structure built from vertices, edges, triangles, and higher-dimensional simplices that generalizes the notion of meshes; see the black skeleton in Fig. 1(A) for an example. A simplicial complex provides a finite, structured representation of the space, enabling efficient algebraic operations for computing homology groups.

As illustrated in Fig. 1, we can now construct a filtration of nested simplicial complexes to study homology across scales. Given a distance metric $\mathcal{D}_X$ (e.g., the Euclidean metric), we construct a sequence of nested topological spaces by forming unions of balls of radius t centered at each point in X, where t varies over a non-negative scale. Persistent homology then records the birth and death of topological features within this filtration, capturing their evolution across scales.

In Fig. 1(A), at $t = 0$, each point is *born* (appears) as its own (connected) component. As t increases, we focus on the important events when components merge with one another to form larger components or tunnels. We begin by tracking the birth and death times of each component or tunnel as well as its lifetime in the filtration. At $t = 6.7$, the brown component merges into the orange component and *dies* (disappears); therefore, the brown component has a lifetime (i.e., *persistence*) of 6.7. At $t = 7.3$, the gray component merges into the purple component and dies; therefore, it has a persistence of 7.3. So on and so forth. At $t = 10.6$, the collection of components forms two tunnels; see two loops beside the red edge in Fig. 1(A). This edge is the birth simplex responsible for the creation of the loops.

The left tunnel is filled (disappears/dies) at $t = 12.1$ with the appearance of the red triangle with a green edge, whereas the right tunnel is filled (disappears/dies) at $t = 14.5$ with the appearance of the red triangle with a blue edge. These simplices, responsible for the death of loops, are called the *death triangles* in Sec. IV-A. The orange component born at time 0 never dies, therefore, it has a persistence of ∞. We record and visualize the appearance (birth), the disappearance (death), and the persistence of homological features in the filtration via persistence diagrams [43] (Fig. 1(C)), or equivalently, persistence *barcodes* [44] (Fig. 1(B)). A point $\rho = (b, d)$ in the persistent diagram of X records a homological feature that is born at time b and dies at time d. 0D and 1D persistence diagrams, denoted as $\mathrm{PD}_0(X)$ and $\mathrm{PD}_1(X)$, capture the births and deaths of components and tunnels. Equivalently in the barcode of Fig. 1(B), such a feature is summarized by a horizontal bar that begins at b and ends at d.

From a computational perspective, a filtration can be rep-

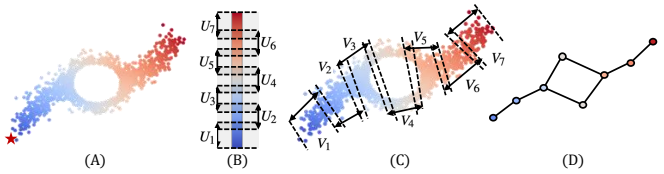

Fig. 2. Illustration of the Mapper pipeline. (A) Point cloud X, colored by the DTB function $f(X)$, with the base point marked by a red star. (B) Cover $\mathcal{U}$ of $f(X)$ using open intervals with $m = 7$ and overlap $p = 0.2$. (C) Induced cover $\mathcal{V}$ of X, formed by clustering points in preimages $f^{-1}(a_i, b_i)$ for each i. (D) Mapper construction, given by the nerve of $\mathcal{V}$.

resented as a sequence of simplicial complexes, providing a compact and algorithmically tractable encoding (see [45] for details). Moreover, birth and death simplices offer spatial clues to the location of the corresponding topological structures [46].

In this paper, persistent homology is used to extract structural information, rather than to recover explicit cycle representatives. It is used in Sec. IV to guide the construction of a Mapper that subsequently supports the preservation of 0D and 1D features in lower-dimensional space in Sec. V.

B. Mapper

Our method adopts a topological approach to data summarization, using the Mapper algorithm [24] to guide the selection of landmarks for dimensionality reduction. To support this, we briefly review the definitions and computational steps underlying Mapper, which yields a discrete representation of the data's structural features.

Filter function. Given a topological space $\mathbb{X}$, a continuous function $f : \mathbb{X} \to \mathbb{R}$, referred to as the *filter function*, plays a critical role in determining how structural features of the data are revealed. In practice, the filter function is chosen to encode geometric or topological information relevant to the analysis task [47]. Common examples include height functions, barycentric distance, curvature, and geodesic distances. In this work, we use the geodesic distance from a fixed source point as the filter function, referred to as the distance-to-basepoint (DTB) function. Prior studies have shown that such functions are effective in revealing 1D homological structures, including prominent loops and circular patterns [47], [48]. While DTB is used as the default in our experiments, our framework is not restricted to this choice; see a discussion in Sec. IV-D.

Mapper graph. To handle discrete point cloud data, we utilize the Mapper algorithm [24], which builds a graph-like summary called a skeleton by analyzing the behavior of the filter function over overlapping regions of the domain. Mapper partitions the image of the filter function into overlapping intervals and examines the pre-images of these intervals to identify structural patterns in the data [49].

As illustrated in Fig. 2, given a point cloud X and a filter function f, we construct a cover $\mathcal{U}$ of $f(X)$ using overlapping intervals controlled by the number of intervals m and overlap ratio p. For example, Fig. 2(B) shows the DTB function partitioned into $m = 7$ intervals with a 0.2 overlap, forming $\mathcal{U} = [U_1, \ldots, U_7]$. Each interval U_i induces a pre-image $V_i = f^{-1}(U_i)$, and the collection $\mathcal{V} = \{V_i\}$ forms a cover of X. Each V_i is then clustered to identify components. In our implementation, we use DBSCAN [50], a density-based clustering method that can identify arbitrarily shaped clusters without requiring the number of clusters to be specified in advance. Other clustering strategies could be used without altering the overall Mapper pipeline. The resulting clusters form the nodes of the Mapper graph, with edges added between nodes that share data points, as illustrated in Fig. 2(C).

The resulting structure, denoted $\mathcal{M}$, is referred to throughout this paper as the *homological skeleton*. A depiction of the full Mapper output is shown in Fig. 2(D), illustrating how the data's topological structure is encoded in a compact, interpretable graph with branches and loops.

C. HIsomap

In the context of non-linear DR, methods such as Isomap, L-Isomap, and the more recent HIsomap [23] play a significant role in the preservation of the intrinsic geometry of high-dimensional data. Give a point cloud X, the implementation of HIsomap involves four steps:

Step 1: kNN graph construction. Build the kNN graph G on X.

Step 2: Filter function. Compute the DTB filter function $f : X \to \mathbb{R}$, which encodes structural properties relevant to DR. The base point is chosen from the extreme points of X.

Step 3: Mapper and landmark selection. Compute Mapper graph $\mathcal{M}$ with user-specified resolution parameters. Nodes of $\mathcal{M}$ correspond to clusters of points in X; the cluster centroids are selected as the landmarks denoted as $X_L \subseteq \mathbb{X}$.

Step 4: Apply L-Isomap. Replace the randomly generated landmarks in L-Isomap with X_L, and apply the rest of the L-Isomap algorithm.

For clarity, we distinguish different versions of L-Isomap according to their landmark selection schemes: the variant with randomly selected landmarks is referred to as *random L-Isomap*, while the homology-preserving approach from [23] is denoted as HIsomap. In this paper, HIsomap serves as a conceptual and algorithmic baseline for the topology-aware DR pipeline developed in subsequent sections.

IV. AdaMapper: A Mapper-based algorithm for homological skeleton extraction

The core idea of AdaMapper is to use persistent homology in the original space to guide Mapper construction in a homology-informed manner. In this section, we present the technical details.

Overview. An overview of AdaMapper is shown in Fig. 3. As a preliminary step, we construct the kNN graph on the input data. This graph is used both to compute the DTB filter function and by AdaHIsomap, so we build it once upfront. The main procedure then proceeds as follows. First, we compute the persistence diagram of the input data to characterize its topological structure. Second, we compute the filter function and partition its range into regular and critical segments, guided by the significant 1D features (loops) identified in the persistence diagram (Sec. IV-A). Third, we derive parameters for the Mapper construction separately for the regular and critical segments, based on the persistence

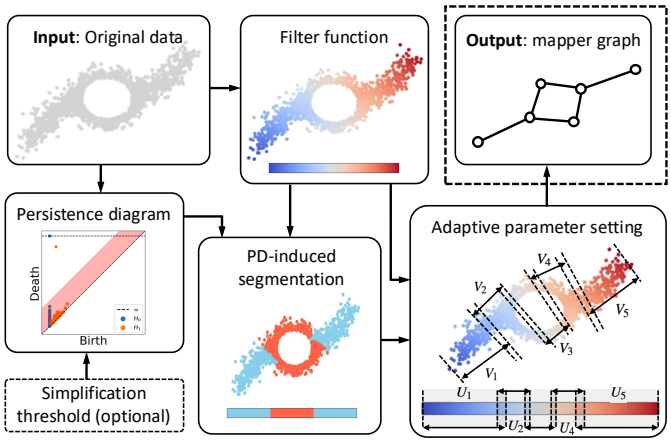

Fig. 3. AdaMapper pipeline for homology-informed Mapper construction.

properties of loops (Sec. IV-B). We define "significant" loops according to their persistence, using a default threshold of 0.35 (i.e., 35% of the maximum persistence). Users may also adjust this threshold interactively, depending on their interpretation of the persistence diagram (Sec. IV-C).

Notations. Let X denote the original data and $f : X \to \mathbb{R}$ its filter function. We use the 1D persistence diagram $\mathrm{PD}_1(X)$ to separate f into the critical segment and the regular segment, denoted $\mathcal{C}$ and $\mathcal{R}$, respectively. With PD-induced segmentation, we calculate the cover $\mathcal{U}$ of $f(X)$ and then derive the cover $\mathcal{V}$ of X. The Mapper graph $\mathcal{M}$ is the 1D skeleton of the nerve of $\mathcal{V}$. The input to AdaMapper is X and the output is $\mathcal{M}$.

With its default settings, AdaMapper requires no user-specified parameters. However, users may optionally incorporate prior knowledge by specifying parameters such as the persistence threshold T, the overlap percentage between adjacent intervals p, and the clustering algorithm settings. If $\mathrm{PD}_1(X)$ does not exhibit any significant loops, the user is prompted to specify the number of intervals m, in which case a standard Mapper is applied.

A. PD-Induced Segmentation

The first step of AdaMapper is to compute the persistence diagram of the original data. This persistence diagram, $\mathrm{PD}_1(X)$, captures the underlying topological structure of the data. The key idea behind AdaMapper is to leverage the homological information encoded in $\mathrm{PD}_1(X)$ to identify a critical segment $\mathcal{C}$ along f, where finer-grained intervals are required to preserve the selected loops in the Mapper construction.

Let $\rho_i \in \mathrm{PD}_1(X)$ be a loop whose persistence exceeds the threshold T (with $T = 0.35$). For each such loop, we determine its corresponding range on f, denoted by $r_i = [u_i, v_i]$. The *critical segment* $\mathcal{C}$ of f is then defined as the union of all such ranges, $\mathcal{C} = r_1 \cup r_2 \cup \cdots \cup r_M$, where M is the number of loops whose persistence is greater than the simplification threshold. The remaining portion of f is referred to as the *regular segment*, denoted by $\mathcal{R}$. Thus, $f = \mathcal{C} \cup \mathcal{R}$. We refer to this procedure as the *PD-induced segmentation* of f. In the remainder of this subsection, we describe how to determine r_i given X, ρ_i, and f.

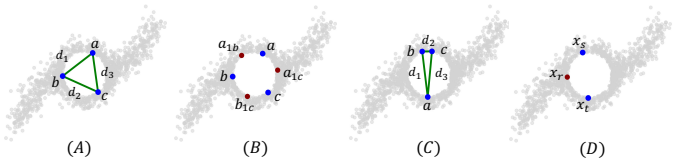

Fig. 4. (A) Balanced death triangle. (B) Centroids of the shortest paths between vertex pairs (a, b), (b, c), and (a, c), denoted by a_{1b}, b_{1c}, and a_{1c}, respectively. (C) Unbalanced death triangle. (D) Vertices x_s and x_t are identified from the longest edge of the death triangle, and the centroid x_r is computed along their shortest path.

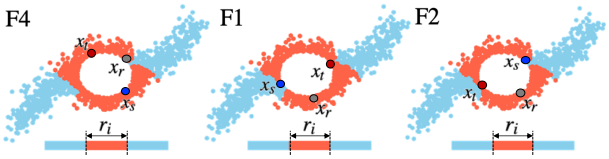

Fig. 5. Derivation of r_i from x_s, x_t, and x_r under cases F4, F1, and F2.

1) Death-Triangle-Based Localization: For each loop ρ_i, we identify its associated death triangle, which corresponds to the 2-simplex responsible for the disappearance of the loop in the filtration (Sec. III-A). This death triangle provides the basis for determining the loop-critical interval r_i.

Step 1: Identify triangle shape. Given a death triangle (calculated by Ripserer.jl library [51], [52]), let its vertices be a, b, and c. We calculate the Euclidean distances between each pair of these vertices in the original point cloud: d_1 for (a, b), d_2 for (b, c), and d_3 for (a, c). To assess the uniformity of the triangle, we compute the normalized entropy $H = -\sum_{i=1}^{3} p_i \frac{\log(p_i)}{\log(3)}$, where $p_i = \frac{d_i}{d_1+d_2+d_3}$.

A higher entropy value indicates more uniformly distributed edge lengths. Based on empirical observation, if $0.975 \leq H \leq 1$, the death triangle is classified as *balanced* (Fig. 4(A)); otherwise, it is classified as *unbalanced* (Fig. 4(C)).

Step 2a: Derive r_i for balanced death triangle. Using the kNN graph G (already constructed for computing the filter function), we compute shortest paths between each pair of vertices (a, b), (b, c), and (a, c), and denote the centroids of these paths as a_{1b}, b_{1c}, and a_{1c} (see Fig. 4(B)). The interval r_i is then defined by the minimum and maximum filter values among the death-triangle vertices and the three centroids. $r_i = [\min f(S),\ \max f(S)]$, where $S = \{a, b, c, a_{1b}, b_{1c}, a_{1c}\}$.

Step 2b: Derive r_i for unbalanced death triangle. If the death triangle edges are not balanced (Fig. 4(C)), we identify the longest edge of the triangle and denote its two vertices as x_s and x_t. Using these vertices and the kNN graph G, we compute the shortest path between x_s and x_t. The centroid of this path is then chosen as x_r (Fig. 4(D)).

We can now derive r_i considering the following four cases.

F1: If $f(x_s) \leq f(x_r) \leq f(x_t)$, then $r_i = [f(x_s),\ f(x_t)]$.
F2: If $f(x_t) \leq f(x_r) \leq f(x_s)$, then $r_i = [f(x_t),\ f(x_s)]$.
F3: If $f(x_r) \leq \min\{f(x_s), f(x_t)\}$, then $r_i = [f(x_r),\ f(x_s) + (f(x_t) - f(x_r))]$.
F4: If $f(x_r) \geq \max\{f(x_s), f(x_t)\}$, then $r_i = [f(x_s) - (f(x_r) - f(x_t)),\ f(x_r)]$.

Fig. 5 illustrates the derivation of r_i under three representative cases. From left to right, the panels correspond to cases F4, F1, and F2. In each case, the interval $r_i = [u_i, v_i]$ derived

from x_s, x_t, and x_r is highlighted in red along the filter function f, together with the points whose filter values fall within this interval. Since the dataset contains a single loop, the critical segment is $\mathcal{C} = r_i$, and the complement defines the regular segment $\mathcal{R}$, shown in blue. Not all intervals r_i in Fig. 5 fully cover the points surrounding the feature ρ_i, and this issue is exacerbated when ρ_i has an irregular shape. However, a suboptimal choice of r_i does not necessarily prevent the construction of a cover $\mathcal{V}$ of X that captures ρ_i; see Sec. IV-B2 for further discussion.

After computing all r_i (for $i \in [1, M]$), the function f is partitioned into critical and non-critical regions, from which we derive adaptive parameters to construct the Mapper cover, as detailed in Sec. IV-B.

2) Design Rationale and Tradeoffs: The goal of the death-triangle strategy is not to recover a geometrically minimal or complete set of points enclosing a loop, but rather to provide a reliable localization of loop-critical regions suitable for AdaMapper construction. This design reflects the role of PD-induced segmentation in AdaMapper, where loop localization is used to control Mapper resolution rather than to extract explicit cycle representatives.

Several existing approaches aim to identify a subset of X that geometrically surrounds a loop represented in $\mathrm{PD}_1(X)$, including representative cycles via involuted homology [52], reconstructed shortest cycles [51], and linear-programming-based optimal cycle representatives [53], among others [54]. While effective for explicit loop reconstruction, these methods can become unstable when multiple loops are spatially proximate or interacting, causing the associated regions to expand uncontrollably or mix contributions from multiple features.

By contrast, the death-triangle strategy prioritizes reliable localization over geometric tightness. By relying on the simplices responsible for the disappearance of loops in the filtration, it yields localizations that are consistent with persistence pairings and robust to interactions among nearby features. As a result, the derived interval r_i may not fully enclose the geometric extent of a loop, particularly for irregular or geometrically complex features.

Importantly, this conservative localization is intentional. In the context of AdaMapper construction, recovering a compact and stable loop-critical region is often more effective than including all points surrounding a loop. Overly aggressive localization that includes substantially more points than necessary can be counterproductive for AdaMapper parameterization, a phenomenon analyzed in detail in Sec. IV-B2. Nevertheless, although r_i may be imperfect, it provides sufficient information to guide the downstream construction and preserve the presence of the corresponding feature ρ_i.

B. AdaMapper Parameterization

The derivation of internal parameters for AdaMapper starts from the observation that a Mapper requires a cover with at least three elements ($\{V_i\}_{i=1}^m$, $m \geq 3$) to identify a loop using the point cloud around it. Using data points only within the critical segment, Fig. 6 demonstrates the Mappers constructed with different sizes of cover. The Mapper captures the loop with $m = 3$, while it fails to summarize the loop with $m = 2$; cf. Fig. 6(right) and Fig. 6(left).

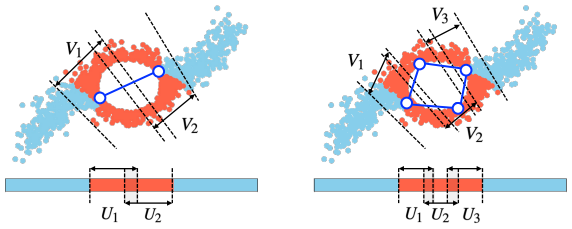

Fig. 6. Critical segment for $m = 2$ (left) and $m = 3$ (right).

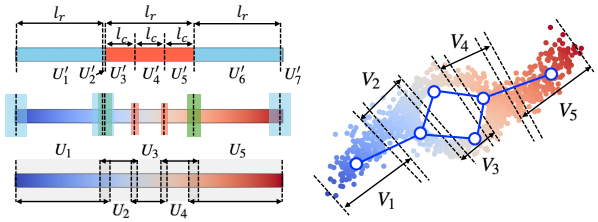

Fig. 7. Adaptive parameter setting for AdaMapper. Left: three-step construction of open intervals $\mathcal{U}$. Right: the corresponding cover $\mathcal{V}$ of X and the resulting Mapper.

1) Parameter Derivation for AdaMapper Construction: Motivated by this observation, we derive the cover $\mathcal{U}$ of $f(X)$ and then the cover $\mathcal{V}$ of X in the following three steps.

Step 1: Initialize the length of open interval. Among the loops selected by the simplification threshold, denoted by $\rho_1, \ldots, \rho_M$, let the lengths of their corresponding ranges be $|r_i| = |v_i - u_i|$ for $i = 1, \ldots, M$. We define the shortest range by $|r_{\min}| := \min\{|r_1|, \ldots, |r_M|\}$. The initial length of open intervals within the critical segment $\mathcal{C}$ is set to $l_c = |r_{\min}|/3$, while the initial length of open intervals within the regular segment $\mathcal{R}$ is set to $l_r = |r_{\min}|$.

Based on these settings, we construct an initial cover $\mathcal{U}'$ of $f(X)$, where $\mathcal{U}' = \{U_i'\}_{i=1}^{m'} = \{[a_i', b_i']\}_{i=1}^{m'}$, with adjacent intervals satisfying $b_i' = a_{i+1}'$. Furthermore, $|a_i' - b_i'| = l_c$ if $U_i' \in \mathcal{C}$, and $|a_i' - b_i'| = l_r$ if $U_i' \in \mathcal{R}$. Note that the initial number of intervals m' may differ from the final number of intervals m after the following two refinement steps.

Using the PD-induced segmentation shown in Fig. 6, we observe that this dataset contains only one significant loop, and $|r_{\min}|$ corresponds to the length of the red region. Based on this observation, the initialization of l_c and l_r is illustrated at the top of Fig. 7(left). We further derive the initial cover $\mathcal{U}'$ of $f(X)$ with $m' = 7$. Specifically, we set $l_c = |r_{\min}|/3$ according to the structure observed in Fig. 6. In contrast, we choose $l_r = |r_{\min}|$, since regions without a prominent loop are expected to exhibit fewer structural changes.

The latter choice is also a natural implementation of persistence simplification on Mapper, which will be discussed in Sec. IV-C.

Step 2: Adaptive expand l_c and l_r with p. The classic Mapper algorithm uses a parameter p to specify the percentage of overlap between each pair of adjacent intervals, thereby enforcing a uniform overlap across the entire cover. In AdaMapper, we instead adaptively expand the interval lengths l_c and l_r to generate overlaps of varying sizes, using a default overlap ratio of $p = 0.2$.

These expansions may occur in three different settings: between two intervals in the critical segment (C1), between two intervals in the regular segment (C2), and between one interval in the critical segment and one in the regular segment (C3). We design separate expansion strategies for each case.

Given $U'_i = [a'_i, b'_i]$ and $U'_{i+1} = [a'_{i+1}, b'_{i+1}]$ with $b'_i = a'_{i+1}$,

C1: if $U'_i \in \mathcal{C}$ and $U'_{i+1} \in \mathcal{C}$, we expand U'_i from $[a'_i, b'_i]$ to $[a'_i, b'_i + p/2 \times l_c)$ and U'_{i+1} from $[a'_{i+1}, b'_{i+1}]$ to $(a'_{i+1} - p/2 \times l_c, b'_{i+1}]$; see the red rectangles in the middle of Fig. 7(left) for an illustration.

C2: if $U'_i \in \mathcal{R}$ and $U'_{i+1} \in \mathcal{R}$, we expand U'_i from $[a'_i, b'_i]$ to $[a'_i, b'_i + p/2 \times l_r)$ and U'_{i+1} from $[a'_{i+1}, b'_{i+1}]$ to $(a'_{i+1} - p/2 \times l_r, b'_{i+1}]$; see the blue rectangles in the middle of Fig. 7(left) for an example.

C3: if $U'_i \in \mathcal{C}$ and $U'_{i+1} \in \mathcal{R}$ or $U'_i \in \mathcal{R}$ and $U'_{i+1} \in \mathcal{C}$, we expand U'_i from $[a'_i, b'_i]$ to $[a'_i, b'_i + p \times l_c)$ and U'_{i+1} from $[a'_{i+1}, b'_{i+1}]$ to $(a'_{i+1} - p \times l_c, b'_{i+1}]$; see the green rectangles the middle of Fig. 7(left) for an example.

The high-level idea behind this adaptive expansion strategy is to extend the initial intervals by a proportion determined by their segment types, namely $\mathcal{C}$ or $\mathcal{R}$. For the C3 case, which lies between C1 and C2, we apply a larger expansion based on l_c, using p instead of $p/2$. Note that after Step 2, the elements of the cover $\mathcal{U}'$ of $f(X)$ become open intervals.

Step 3: Finalize the cover $\mathcal{U}$ of $f(X)$. Upon closer inspection of $\mathcal{U}'$, we observe that some intervals $U' \in \mathcal{U}'$ are completely contained within their adjacent intervals. For example, U'_2 is contained in U'_1, and U'_6 is contained in U'_7. In this step, we remove such redundant intervals to obtain the final cover $\mathcal{U}$ of $f(X)$, as illustrated at the bottom left of Fig. 7. After this refinement, the number of intervals becomes $m = 5$.

Using the cover $\mathcal{U}$ of $f(X)$, we then derive the adaptive cover $\mathcal{V}$ of X. Based on $\mathcal{V}$, we compute the Mapper graph of X using a user-defined clustering algorithm; see the right panel of Fig. 7. Although the choice of clustering algorithm is not central to our method, we additionally provide an automatic parameter selection strategy for DBSCAN based on PD-induced segmentation in Appendix A.

2) Robustness to Imperfect Loop Localization: As noted in Sec. IV-A, the interval r_i derived from the death-triangle strategy may not perfectly localize all points surrounding the feature ρ_i. We now show that such imperfect localization is nevertheless sufficient, and in some cases even preferable, for AdaMapper. As illustrated in Fig. 6, Mapper requires at least three cover elements within a loop-critical region to capture a 1D loop. Given an imperfect interval $r_i = [u_i, v_i]$, the point set $\{x \mid f(x) \in [u_i, v_i]\}$ typically forms a subset of the points surrounding ρ_i, ensuring that AdaMapper allocates at least three intervals within this region.

In contrast, cocycle-based strategies [55] that include substantially more points may produce excessively large loop-critical regions, particularly when multiple loops interact. When mapped onto the filter function, these enlarged regions can reduce the effective number of intervals assigned to each loop, thereby violating the condition $m \geq 3$ and preventing successful loop capture despite the broader geometric coverage. Therefore, although the death-triangle strategy may yield only an approximate localization of ρ_i, its conservative characterization preserves sufficient resolution in the most relevant regions, enabling AdaMapper to reliably capture ρ_i; see Fig. 8 for examples.

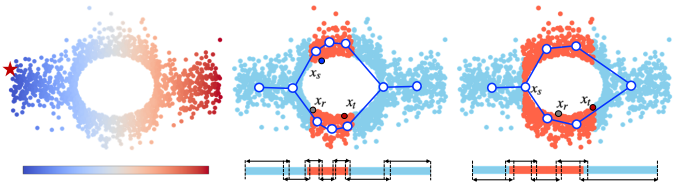

Fig. 8. Capturing loops with an imperfect critical segment. Left: point cloud X colored by the DTB function. Middle and right: segmentations and corresponding Mapper constructions using manually chosen x_s points.

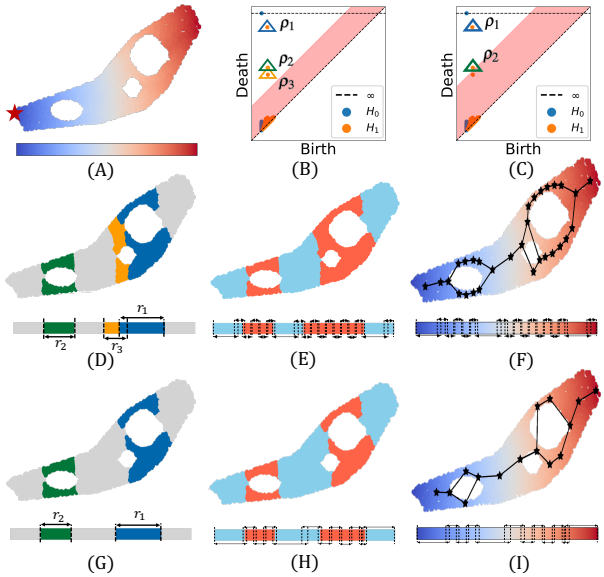

Fig. 9. A toy example illustrating AdaMapper. (A) Point cloud X, colored by the DTB function, with the base point marked by a red star. (B–C) Persistence diagrams with default and user-specified simplification thresholds, T and T'. (D–F) and (G–I) Loop-induced ranges, persistence-guided segmentations, and corresponding Mapper constructions for thresholds T and T'.

C. A Toy Example for AdaMapper

We now consider a slightly more complex example than the point clouds presented previously to illustrate the pipeline of AdaMapper and demonstrate its ability to capture loops under a user-specified simplification threshold.

Given the point cloud X in Fig. 9(A), its persistence diagram $\mathrm{PD}_1(X)$ identifies three loops, ρ_1, ρ_2, and ρ_3, under the default simplification threshold T; see Fig. 9(B). Using the DTB function $f(X)$ together with the methodology introduced in Sec. IV-A, we derive the corresponding ranges r_1, r_2, and r_3 along $f(X)$; see Fig. 9(D). Notably, r_1 and r_3 overlap. The critical segment $\mathcal{C}$ is therefore defined as $r_1 \cup r_2 \cup r_3$, while the regular segment $\mathcal{R}$ consists of the remaining portions of $f(X)$; see Fig. 9(E). Following the strategy described in Sec. IV-B, we set the initial interval length within $\mathcal{C}$ to $l_c = |r_3|/3$. The resulting cover $\mathcal{U}$ of $f(X)$ is illustrated at the bottom of Fig. 9(E-F). As shown in Fig. 9(F), AdaMapper successfully captures all three loops using the default parameter setting.

We now manually choose a simplification threshold T' to filter out the loop ρ_3; see Fig. 9(C). Under this threshold, the selected loop ranges reduce to r_1 and r_2, yielding a critical

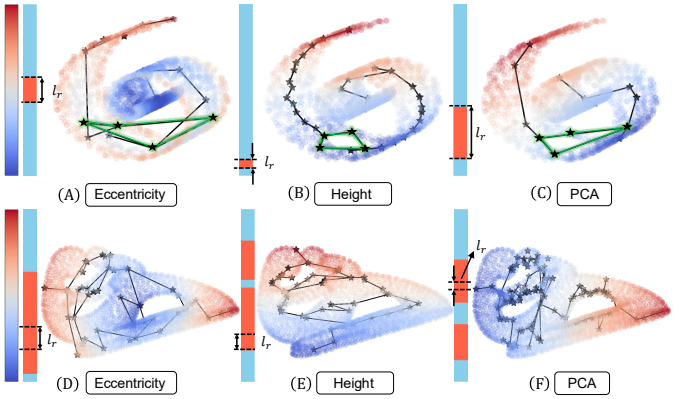

Fig. 10. Effect of different filter functions $f(X)$ on AdaMapper constructions. (A–F) Point clouds colored by filter values (blue: low, red: high); adjacent color bars indicate regular (blue) and critical (orange) ranges. Black graphs show the AdaMapper skeletons. (A–C) Swiss Hole and (D–F) Fertility datasets using eccentricity, height, and PCA filters.

segment defined by $\mathcal{C} = r_1 \cup r_2$; see Fig. 9(G-H). In this setting, the construction of the cover $\mathcal{U}$ of $f(X)$ begins with the initialization $l_c = |r_2|/3$, which ultimately produces a Mapper graph containing two loops; see Fig. 9(H-I).

D. Discussion on AdaMapper

AdaMapper is designed to guide Mapper construction using persistent homology as a mechanism for adaptive resolution control, rather than to guarantee recovery of all topological features present in the data. Consequently, its behavior reflects both the strengths and inherent limitations of persistence-guided Mapper constructions.

Scope and limitations. Not every loop identified in a persistence diagram can necessarily be represented by AdaMapper. For example, a torus contains two independent loops, yet the loop associated with the inner tube may fail to appear in the resulting Mapper, even when it is detected through PD-induced segmentation. This limitation stems from the interaction between the filter function and the Mapper cover: some topological features cannot be faithfully represented under a given scalar function, regardless of how finely the cover is refined. Such behavior is a well-known property of Mapper-based summaries and is not specific to AdaMapper.

Persistence versus Mapper resolution. The length of an interval r_i derived from PD-induced segmentation is not necessarily proportional to the persistence of the corresponding feature ρ_i. Persistence measures the stability of a feature across scales in the original space, whereas r_i is defined along the filter function. Consequently, persistence-based simplification does not directly correspond to interval-based simplification in Mapper. In particular, the smallest interval r_i may correspond to a feature with relatively high persistence, potentially leading to over-refinement or unintended simplification. To address this discrepancy, AdaMapper does not assume a one-to-one relationship between persistence and cover size. Instead, it uses persistence to localize loop-critical regions while explicitly controlling resolution along the filter function.

Filter function dependence and sensitivity. Like all Mapper-based methods, AdaMapper depends on the choice of filter function. In this work, we use the distance-to-basepoint (DTB) function as the default, although the framework itself is not restricted to this choice. Fig. 10 presents AdaMapper constructions using eccentricity, height, and PCA as filter functions. For both the Swiss Hole and Fertility datasets, the same loop-critical features are consistently identified across all filters through PD-induced segmentation.

However, the resulting interval sizes l_c and l_r vary substantially across filters, leading to different cover resolutions and Mapper complexities. This effect is particularly evident in Fig. 10(B) for Swiss Hole and in Fig. 10(E-F) for Fertility, where different filter choices produce noticeably different graph densities, even though all resulting Mapper graphs remain meaningful and interpretable. Importantly, when the filter function varies smoothly along the underlying manifold, AdaMapper consistently captures the loop-critical regions while making the influence of filter choice on Mapper complexity explicit and transparent.

V. AdaHIsomap: Adaptive Homology-Preserving Dimensionality Reduction

We now introduce AdaHIsomap, an adaptive homology-preserving DR technique that combines AdaMapper-derived homological skeletons with landmark Isomap. By using homology-informed skeletons to guide landmark selection, AdaHIsomap aims to better preserve both loop structures and global connectivity in low-dimensional embeddings.

A. Adaptive Homology-Preserving Landmark Selection

AdaHIsomap follows the same overall pipeline as HIsomap (Sec. III-C), with the key difference occurring in Step 3: landmark construction. Specifically, AdaHIsomap modifies HIsomap in two principled ways: (1) We replace the classical Mapper with AdaMapper, which constructs a homology-informed skeleton using PD-induced adaptive covers. (2) We augment Mapper-based landmarks with stochastic anchor points sampled from the regular segment $\mathcal{R}$ of the filter function.

Together, these modifications aim to preserve both 1D (loop) and 0D (connectivity) topological features during DR, while substantially reducing the need for manual parameter tuning.

Mapper-based landmarks for 0D and 1D homology preservation. In the classic Mapper algorithm, each node represents an abstract cover element of the point cloud, corresponding to a cluster of points in X. In our framework, following the HIsomap approach, we select the centroid of each cluster of $\mathcal{M}$ (constructed with AdaMapper) as its representative, and these representatives collectively serve as the subset of the landmarks for AdaHIsomap.

Stochastic anchor points for enhancing 0D homology preservation. A closer examination of the cover $\mathcal{U}$ of $f(X)$ used in constructing $\mathcal{M}$, such as those shown in Figs. 7 and 9, reveals that substantially more clusters—and therefore more landmarks—are selected within the critical segment $\mathcal{C}$ than within the regular segment $\mathcal{R}$. This imbalance in landmark distribution can distort the preservation of 0D homological

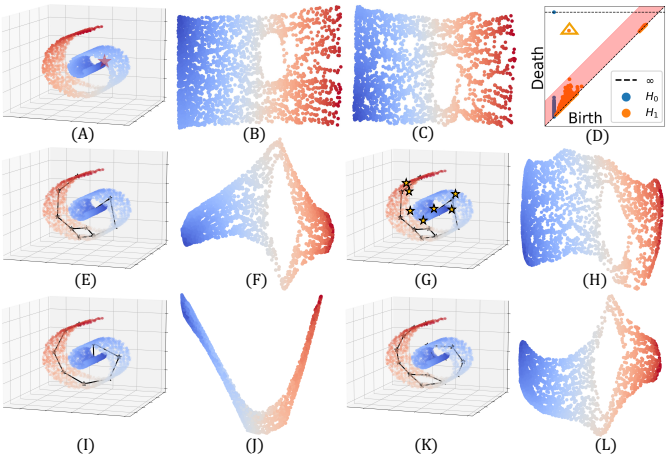

Fig. 11. 2D embeddings of the Swiss Hole dataset. (A) Point cloud X, colored by the DTB function $f(X)$. (B) Isomap embedding. (C) Random L-Isomap embedding with $|X_L| = 18$. (D) Persistence diagram $\mathrm{PD}(X)$. (E–F) L-Isomap using nodes of the Mapper computed by AdaMapper as landmarks. (G–H) AdaHIsomap embeddings with $|X_L| = 18$. (I–J) and (K–L) L-Isomap embeddings using nodes of the standard Mapper as landmarks (HIsomap), with $m = 10$ and $m = 15$.

features. Importantly, such distortions cannot be resolved simply by decreasing the interval lengths in the regular segment. We illustrate this phenomenon with an example in Sec. V-B.

To improve the preservation of 0D features, we augment AdaHIsomap with stochastic anchor points. Specifically, we randomly select one data point from each open interval in the regular segment $\mathcal{R}$ and include it as an additional landmark. These stochastic anchors rebalance the landmark distribution and improve the preservation of global connectivity.

Without these additional anchors, most landmarks are concentrated near loop-critical regions, while the regular segment $\mathcal{R}$ remains sparsely represented. Consequently, regions away from dominant loops may be insufficiently sampled during DR, leading to distortions in preserving 0D homological features; see Fig. 11(F). By selecting one stochastic anchor point from each interval in $\mathcal{R}$, we maintain a more balanced representation across the filter range and improve the preservation of global connectivity without introducing a large number of additional landmarks; see Fig. 11(H).

B. A Toy Example

We illustrate the AdaHIsomap pipeline using the Swiss Hole dataset (Fig. 11), a point cloud sampled from a Swiss roll containing a subtle interior loop. For comparison, Fig. 11(B-C) show embeddings generated by Isomap and random L-Isomap, respectively, both using $|X_L| = 18$ landmarks.

We first compute a DTB filter function with respect to an extremal base point; see Fig. 11(A). We then compute the persistence diagram, which captures the homological structure of the dataset; see Fig. 11(D). Based on the detected loop, AdaMapper constructs a 1D homological skeleton (the black skeleton in Fig. 11(E)), whose nodes are used as a subset of landmarks for AdaHIsomap.

If only the skeleton nodes are used as landmarks, the resulting 2D embedding is shown in Fig. 11(F). To further improve the preservation of 0D homology, AdaHIsomap introduces stochastic anchor points, highlighted as orange stars in Fig. 11(G). The resulting embedding is shown in Fig. 11(H). Comparing Fig. 11(F) and (H) with (B) demonstrates the effectiveness of AdaMapper-induced landmarks in preserving 1D homology, while also partially preserving 0D homological structure during DR. Furthermore, comparing Fig. 11(F) with (H) shows that stochastic anchor points in the regular segment are essential for mitigating distortions of 0D features located away from the dominant loop.

We next investigate why stochastic anchor points are necessary, rather than simply increasing the resolution of AdaMapper within the regular segment. To this end, we first examine the behavior of the classical Mapper under different parameter settings. Our results show that the number of intervals m must exceed 15 for the classical Mapper to successfully capture the prominent hole in the Swiss Hole dataset; compare Fig. 11(I) and (K). When fewer intervals are used, the Mapper captures insufficient homological information, leading to greater distortion in the resulting embedding; compare Fig. 11(J) and (L).

In contrast, AdaHIsomap captures the loop structure using only $m = 10$, where the parameter value is determined automatically by AdaMapper without manual tuning. Fig. 11(L) may also be interpreted as the result of a 2D embedding constructed using fine-grained cover elements concentrated within the regular segment. Comparing Fig. 11(L) with (H) further demonstrates that stochastic anchor points sampled from the regular segment reduce distortions for data points located away from one-dimensional loops.

Compared with Isomap, AdaHIsomap provides clear advantages in preserving significant 1D loops under simplification, as further demonstrated in Sec. VI. Nevertheless, AdaHIsomap retains certain limitations, which are discussed in Sec. V-D.

C. Computational Considerations

Compared with HIsomap, AdaHIsomap introduces modest additional computational overhead. The dominant computational costs—including kNN graph construction, geodesic distance computation, and multidimensional scaling—remain unchanged from the original Isomap-based pipeline. The additional overhead in AdaHIsomap arises primarily from constructing a homology-informed skeleton, while persistence diagrams are used only to identify prominent 1D features rather than to analyze the full filtration of the dataset. In practice, persistence diagram computation can be further accelerated through subsampling, and all experiments in Sec. VI were conducted at data scales where this overhead remains manageable. Consequently, AdaHIsomap remains computationally practical for the datasets considered in this work, while providing additional topological guidance for landmark selection.

Beyond algorithmic runtime, practical computational cost is also influenced by parameter-tuning requirements. In Sec. VI, baseline methods such as Isomap, UMAP, and t-SNE were evaluated across multiple parameter configurations for each dataset, whereas AdaHIsomap, TopoAE++, and TopoMap were used largely with their default settings (see Appendix B,

Table II). Therefore, overall computational effort cannot be assessed solely based on runtime comparisons.

D. Limitation Analysis of AdaHIsomap

AdaHIsomap inherits several limitations from Isomap. The most significant is its inability to effectively analyze datasets containing multiple manifolds. For this reason, all datasets considered in Sec. VI consist of a single manifold. Nevertheless, compared with Isomap, AdaHIsomap exhibits improved robustness to noise, since noisy points are less likely to be selected as landmarks.

Another limitation stems from the use of classical MDS in Isomap to project the geodesic distance matrix. This projection is fundamentally a linear transformation based on pairwise distances. Consequently, when loops encoded in the distance matrix are oriented approximately perpendicular to one another, some loops may not be preserved in the resulting Isomap embedding. The same limitation naturally carries over to AdaHIsomap; see, for example, Fig. 14.

In addition, the performance of AdaHIsomap depends strongly on the quality of the AdaMapper construction, making it sensitive to limitations inherited from that component. Unlike AdaMapper, AdaHIsomap may not fully reflect the intended simplification effect, since landmarks can still be placed near low-persistence loops.

Finally, AdaHIsomap is intended as a homology-informed extension of landmark Isomap rather than a comprehensive solution to all topology-preserving dimensionality reduction problems. Future work could incorporate more advanced embedding techniques or alternative distance models to further address these limitations.

VI. Experimental Results

We first describe the experimental setup, baseline methods, and parameter settings used to evaluate our approach.

Evaluation rationale. Before presenting the dataset-specific results, we first clarify the evaluation rationale underlying our experimental design. As illustrated in Fig. 12, fixed-cover Mapper constructions—and consequently HIsomap—are highly sensitive to manually specified cover resolutions. Even reasonable parameter choices can produce substantially different Mapper skeletons and downstream embeddings, making direct comparisons based on "best" parameters inherently ill-posed. Importantly, Fig. 12 is not intended as a performance comparison, but rather as a conceptual demonstration of this parameter sensitivity and its implications for evaluation. In contrast, AdaMapper automatically derives cover resolutions from persistent homology, yielding a more stable and reproducible structural summary. This observation motivates our evaluation protocol and explains why we focus on comparing AdaHIsomap with methods that do not rely on manual Mapper parameter tuning.

Experimental setup. We evaluate our method on 11 datasets to demonstrate its behavior under default settings across diverse data types. All results were obtained on a MacBook Pro with 16 GB of RAM, an Apple M3 chip with 8 CPU cores and 10 GPU cores. For all datasets except the

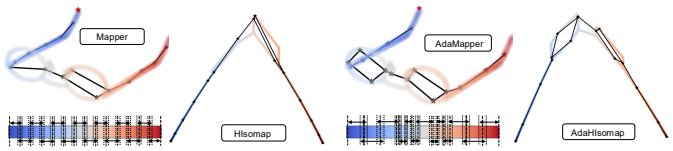

Fig. 12. Comparison of fixed-cover and adaptive-cover Mapper constructions and their impact on downstream dimensionality reduction.

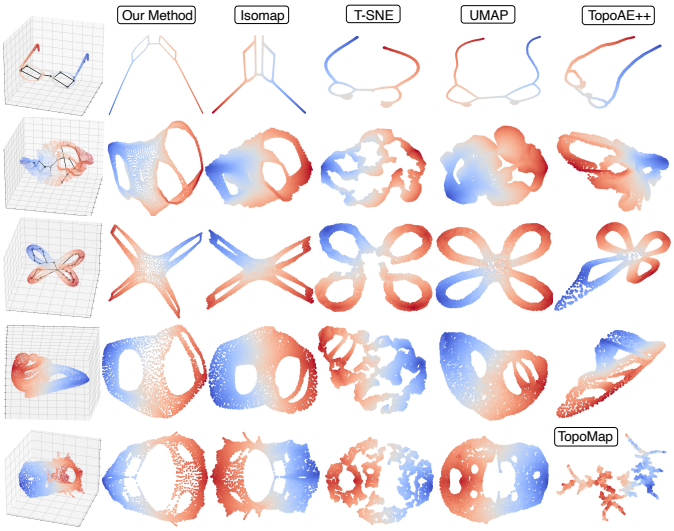

Fig. 13. Dimensionality reduction results for the Glasses, 4elt, Octa, Fertility, and Bcsstk31 datasets. From left to right: original point clouds, AdaHIsomap, Isomap, t-SNE, UMAP, and TopoAE++/TopoMap embeddings. Points are colored by the filter function.

Coauthor dataset, we use the default settings of AdaHIsomap. Since the Coauthor dataset contains only link information, we adapt our framework accordingly. For comparison, we include widely used DR methods introduced in Sec. II, selecting the best-performing configurations through parameter sampling. For TopoAE++, we use the default settings reported in [40]. TopoMap, which is primarily designed to preserve 0D homological features, performs poorly relative to the other methods. Its embeddings are therefore presented in Appendix C, except in cases where TopoAE++ results are unavailable (Bcsstk31 due to its size and Coauthor due to its data type).

We omit explicit HIsomap results. Since HIsomap requires manual parameter specification, running it with the cover parameters automatically derived by AdaMapper produces nearly identical outcomes, while alternative parameter choices can yield substantially different results.

A full list of parameter settings is given in Appendix B, Table II. In the remainder of this section, we use N to denote dataset size and Dim to denote dimensionality.

Quantitative evaluation. Table I summarizes the quantitative evaluation results across all datasets. For each metric, the best result is highlighted in **bold**, and the second-best result is underlined. The evaluation is based on three metrics: one measuring geometric fidelity (RMSE) and two measuring topological preservation (PDW^0 and PDW^1). Detailed definitions of these metrics are provided in Sec. VI-E.

TABLE I
QUANTITATIVE EVALUATION RESULTS FOR 11 DATASETS ACROSS FIVE METHODS. WD1 DENOTES $PDW^1(X,Y)$, AND WD0 DENOTES $PDW^0(X,Y)$.

Dataset	Metric	AdaHIsomap	Isomap	T-SNE	UMAP	TopoMap	TopoAE++
Glasses [56]–[61]	RMSE	**4.4721**	10.0819	5.5015	5.5312	9.5470	5.8106
	WD1	2.3330	**2.2930**	2.6935	2.3318	2.6638	2.6500
	WD0	15.4884	16.1414	14.2706	15.5181	**13.1878**	13.5380
Fertility [62]	RMSE	5.4661	**2.9248**	5.1855	3.1769	9.4070	7.8328
	WD1	6.8148	7.0455	7.9490	7.2081	7.3557	**6.7560**
	WD0	30.7790	30.4625	31.4957	**28.7413**	44.7255	35.4177
Octa	RMSE	5.3117	6.2988	5.1494	**4.4168**	13.3433	11.9366
	WD1	8.8498	9.1891	9.7950	9.3686	**8.5730**	9.1632
	WD0	45.9230	45.7363	39.4204	**39.3655**	53.7894	47.6841
Vortex Street [63], [64]	RMSE	**135.2299**	138.3632	136.0178	136.3662	137.2030	137.2617
	WD1	15.5263	15.4714	**15.4537**	16.0610	15.9942	15.9470
	WD0	255.3856	**253.9320**	255.5551	258.8871	256.0890	258.4001
Mice [65]	RMSE	**58.1075**	59.7248	58.3452	58.5394	59.4077	60.6419
	WD1	36.1554	35.9450	36.2990	36.0612	**35.5404**	36.2929
	WD0	407.4759	405.7253	409.0985	408.1793	**404.3524**	410.8558
4elt [66]	RMSE	6.7877	**4.3194**	7.5606	5.9708	24.4609	9.5671
	WD1	**8.2003**	10.1485	10.4036	9.8665	8.2998	8.9554
	WD0	55.3615	52.3128	54.9673	**42.5516**	68.9178	53.5470
Bcsstk31 [67]	RMSE	6.3713	**3.8359**	5.9974	4.9440	12.0154	-
	WD1	8.8243	8.7969	**8.1602**	9.6296	8.4452	-
	WD0	50.0520	50.1318	51.5510	**50.0258**	61.1355	-
Cartoon [68]	RMSE	23.5719	23.9283	**23.3440**	23.7147	23.6332	23.7677
	WD1	2.3970	2.3273	2.3658	**2.3163**	2.3263	2.3617
	WD0	76.0661	**75.7054**	76.2669	76.6828	76.7583	76.6847
Face [69]	RMSE	**177.9032**	179.4126	178.7576	178.8818	178.4508	179.2001
	WD1	8.3507	8.2292	**8.0623**	8.1025	8.1389	8.3904
	WD0	246.0745	**245.6767**	246.1492	246.9120	248.0108	247.0201
Coauthor [70]	RMSE	-	-	-	-	-	-
	WD1	13.1683	12.8986	14.4222	**11.9474**	13.7099	-
	WD0	108.1479	94.6187	117.7190	65.2374	**0.0116**	-

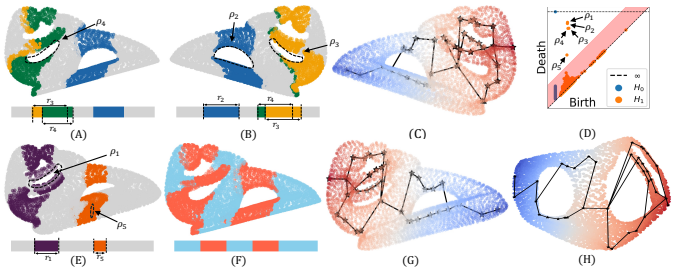

Fig. 14. Mapper analysis of the Fertility dataset. (A–B) Three critical ranges captured by AdaMapper and preserved by AdaHIsomap. (E) Two critical ranges captured by AdaMapper but not preserved by AdaHIsomap. (D) Persistence diagram. (F) Persistence-guided segmentation. (C, G) Mapper embedded in 3D. (H) Mapper in the AdaHIsomap embedding.

A. Point Cloud Data

We evaluate our method on point cloud datasets with nontrivial homology, covering 3D and high-dimensional cases.

1) 3D Datasets: The Glasses dataset is a 3D object with $N = 2000$ points [56]. As shown in Fig. 13(1st row), AdaHIsomap and TopoAE++ outperform the other methods. The remaining methods capture two loops but distort the shape: Isomap and UMAP introduce slight distortion, while t-SNE produces more severe distortion.

The 4elt dataset is a 3D embedding with $N = 7807$ points [66]. In Fig. 13(2nd row), both AdaHIsomap and Isomap preserve three major loops and outperform t-SNE, UMAP, and TopoAE++. Quantitatively, AdaHIsomap achieves the best PDW^1 score, while Isomap and t-SNE perform better on PDW^0 (see Table I).

The Octa dataset has $N = 2994$ points sampled from a mesh of octahedral handles. As shown in Fig. 13(3rd row), AdaHIsomap successfully preserves all four loops. Other methods also perform well, except t-SNE, which struggles

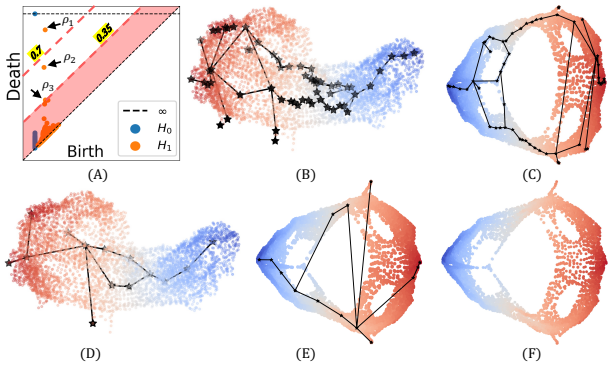

Fig. 15. Mapper analysis of the Bcsstk31 dataset with default and user-specified simplification thresholds $T = 0.35$ and $T' = 0.7$. (A) Persistence diagram with thresholds T and T'. (B–C) Mapper with T in 3D and 2D. (D–E) Mapper with T' in 3D and 2D. (F) AdaHIsomap embedding with T'. All visualizations (B–F) are colored by the DTB function.

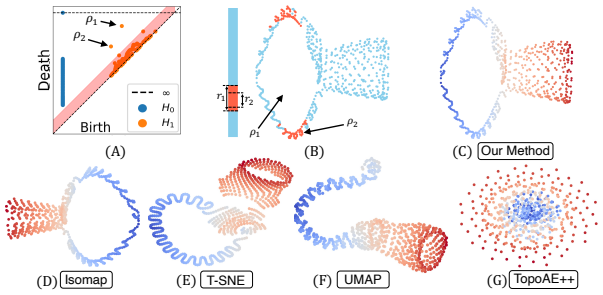

Fig. 16. Dimensionality reduction of the Mice dataset. (A) Persistence diagram. (B) Persistence-guided segmentation. (C–G) Embeddings produced by AdaHIsomap, Isomap, t-SNE, UMAP, and TopoAE++, respectively, colored by the DTB function.

with central 0D features.

The Fertility dataset is a 3D object consisting of $N = 3739$ points [62]. We first analyze this dataset using AdaMapper. As shown in Fig. 14(D), the persistence diagram reveals five significant loops. Specifically, ρ_1 is formed by the two arms (Fig. 14(E)); ρ_2 corresponds to the loop between the mother's legs and the ground (Fig. 14(B)); ρ_3 is enclosed by the left arm and leg (Fig. 14(B)); ρ_4 is enclosed by the right arm and leg (Fig. 14(A)); and ρ_5 corresponds to a tunnel within the 3D object (Fig. 14(E)). These results demonstrate that AdaMapper successfully derives the corresponding ranges for all five significant 1D features, leading to an accurate critical segmentation for these loops; see Fig. 14(F). The primary limitations are the inability of both AdaMapper and AdaHIsomap to capture ρ_5, as well as the failure of AdaHIsomap to preserve ρ_1.

As shown in Fig. 13(4th row), UMAP performs best, preserving four loops, while TopoAE++ also preserves four loops but with greater distortion of the natural shape. Although the AdaMapper skeleton captures four loops (Fig. 14(H)), AdaHIsomap preserves only three in its embedding. Both UMAP and TopoAE++ outperform AdaHIsomap in preserving ρ_1. Because ρ_1 is perpendicular to ρ_3 and ρ_4, AdaHIsomap prioritizes preserving ρ_3 and ρ_4 at the expense of ρ_1, reflecting the limitations of AdaHIsomap discussed in Sec. V-D. However, based on the evaluation results in Table I, TopoAE++ and AdaHIsomap outperforms all other methods on PDW^1,

while UMAP achieves the best performance on PDW^0. It is worth noting that all DR methods fail to preserve ρ_5, which corresponds to a tunnel.

The Bcsstk31 dataset represents a 3D embedding constructed from a stiffness matrix with $N = 5660$ points [67]. As shown in Fig. 13(5th row), Isomap and AdaHIsomap perform better at preserving 1D homological features compared to t-SNE, UMAP, and TopoMap. In particular, AdaHIsomap emphasizes 1D features by enlarging loops, though at the cost of introducing distortions to 0D features. As illustrated in Fig. 15(A), the dataset contains three loops above the default simplification threshold. By setting a manual threshold of $T' = 0.7$, two 1D features are filtered out. The Mapper generated by AdaMapper is successfully simplified under this homology-based resolution control (Fig. 15(D)). Applying AdaHIsomap to the simplified Mapper yields the embeddings shown in Fig. 15(E) and (F). Compared with Fig. 15(C), the AdaHIsomap embedding with threshold T' preserves fewer 1D features. However, the effect of simplification is less pronounced for AdaHIsomap than for AdaMapper, since more than two loops remain visible in Fig. 15(F). This occurs because several landmarks are located near low-persistence loops, which promote the preservation of local structures around nearby points.

2) High-Dimensional Dataset: The Mice dataset contains $N = 674$ points in $Dim = 300$ [65]. In Fig. 16, we compare the DR results together with the persistence diagram computed from the original high-dimensional space. As shown in Fig. 16(A), the dataset contains two significant topological features, ρ_1 and ρ_2. These features likely correspond to periodic patterns in the mice temperature profiles, with ρ_1 associated with circadian rhythms and ρ_2 associated with ultradian rhythms. As shown in Fig. 16(C-G), AdaHIsomap, Isomap, and UMAP each preserve only one of the two features, whereas t-SNE successfully preserves both and achieves the best overall performance. TopoAE++ fails on this dataset. Interestingly, Fig. 16(B) shows that AdaHIsomap attempts to preserve ρ_2 by incorporating $|r_2|$ into the Mapper construction, since $|r_2| < |r_1|$. However, as discussed in Sec. IV-D, AdaMapper cannot guarantee the preservation of all features detected in the persistence diagram. Furthermore, Fig. 16(B) reveals that the critical range r_1 associated with ρ_1 in the DTB function is imperfect. Nevertheless, as discussed in Sec. IV-B2, even an imperfect PD-induced segmentation can still yield a meaningful Mapper construction.

B. Scientific Data

The VortexStreet dataset is a classic 2D von Kármán vortex street simulation [63], [64] with $N = 495$ and $Dim = 8192$, where each simulation is treated as a data point. We focus on the region of vortex shedding behind the cylinder and use velocity magnitude as the scalar field. Since vortex shedding becomes intense after 1005 time steps, we analyze the last 495 time instances (steps 1005–1500 of the original 1500). This subset emphasizes the periodic flow structures induced by vortex shedding and serves as an effective test case for evaluating the ability of DRs to capture recurring topological features. The persistence diagram (Fig. 17, top middle) reveals a prominent 1D homological feature.

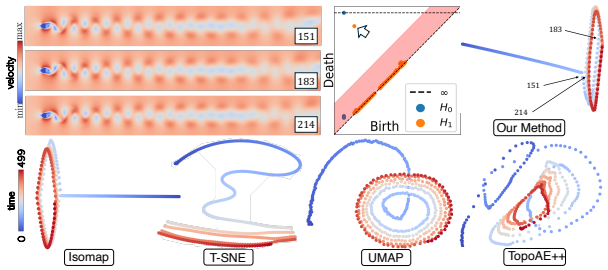

Fig. 17. Dimensionality reduction for the VortexStreet dataset. Top left: selected instances. Top middle and right: persistence diagrams in the original space and the AdaHIsomap embedding, respectively. Bottom: embeddings produced by Isomap, t-SNE, UMAP, and TopoAE++.

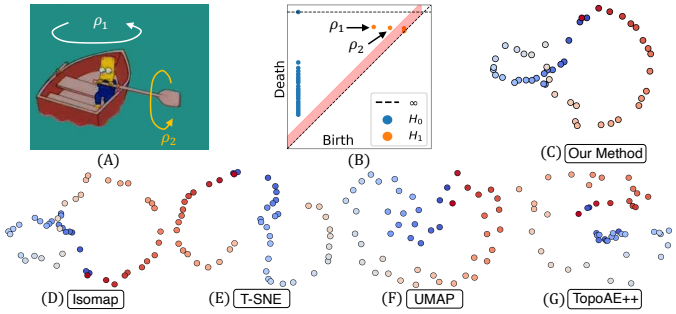

Fig. 18. Dimensionality reduction of the Cartoon dataset. (A) Example instance. (B) Persistence diagram. (C–G) Embeddings produced by AdaHIsomap, Isomap, t-SNE, UMAP, and TopoAE++, respectively, all colored by time.

The 2D embeddings of four DR algorithms are shown in Fig. 17(bottom), where each point represents an instance and is colored by time. All DR algorithms detect the periodicity from steps 151 to 495, although TopoAE++ exhibits greater distortion of the natural shape. The results indicate a repeating pattern of alternating vortices with a time interval roughly equal to 63. For example, the instance at time step 151 is very similar to the instance 214. Among the embeddings, Isomap and AdaHIsomap provide the clearest separation between periodic and non-periodic regions. Furthermore, as shown in Table I, AdaHIsomap achieves the best RMSE compared to the other methods.

C. Imaging Data

The Cartoon dataset records a cartoon character rowing a boat and consists of $N = 48$ frames with $Dim = 4900$ [68]. This dataset is of interest because it contains two nontrivial 1D loops, ρ_1 and ρ_2 (Fig. 18(B)). The dominant feature, ρ_1, corresponds to the circular motion of the character with the boat, while ρ_2 arises from the circular motion of the paddle (Fig. 18(A)). As shown in Fig. 18(C–G), most DR methods are able to preserve both ρ_1 and ρ_2. However, only AdaHIsomap preserves these two features using its default parameter settings. For the other methods, we explored a wide range of parameter combinations to obtain embeddings that reveal both loop structures.

The Face3DModel dataset is constructed by capturing a face from different viewing angles and consists of $N = 387$

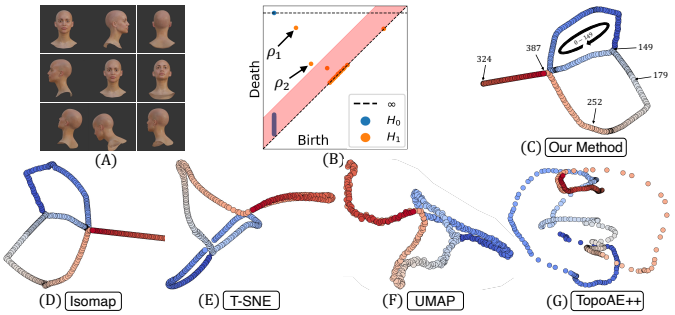

Fig. 19. Dimensionality reduction of the Face dataset. (A) Nine sample instances. (B) Persistence diagram. (C–G) Embeddings produced by AdaHIsomap, Isomap, t-SNE, UMAP, and TopoAE++, respectively, all colored by time.

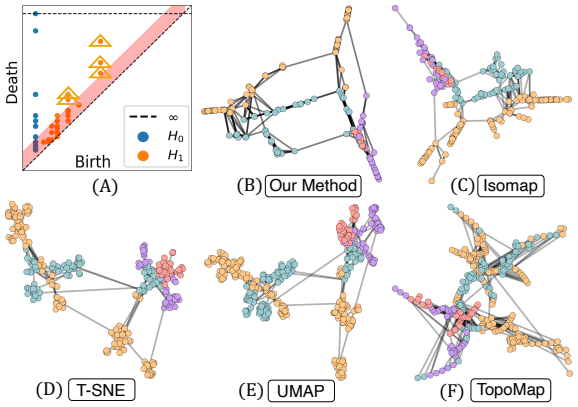

Fig. 20. Dimensionality reduction of the Coauthor dataset. (A) Persistence diagram. (B–F) Embeddings produced by AdaHIsomap, Isomap, t-SNE, UMAP, and TopoMap, respectively, colored by cluster labels.

frames with $Dim = 4900$ [69]. The camera first moves horizontally around the subject from time steps 0–149, then moves vertically downward from 149–179, shifts to the right from 179–252, moves upward from 252–324, and finally moves downward again from 324–387. As shown in Fig. 19, the persistence diagram identifies two significant loops, ρ_1 and ρ_2. Both AdaHIsomap and Isomap perform well in preserving ρ_1, which corresponds to the horizontal rotation from time steps 0–149. The second feature, ρ_2, does not represent a full rotation; it spans time steps 149–278. However, because the trajectory at the end of this movement approaches the starting point of ρ_1, a second loop ρ_2 emerges.

In addition, during time steps 0–149, t-SNE and UMAP exhibit similar embedding patterns, while AdaHIsomap and Isomap also produce consistent structures. In contrast, TopoAE++ shows substantially greater discrepancies relative to the other methods. Considering the underlying camera motion during this period, the embeddings produced by all methods except TopoAE++ appear reasonable.

D. Network Data

We apply our methods to the Coauthor network dataset, which contains $N = 379$ nodes [70]. Unlike the previous datasets, the Coauthor dataset consists solely of an edge list, where each edge is represented as an ordered pair of nodes (source, destination). Accordingly, we make several minor modifications to the AdaMapper and AdaHIsomap pipelines. Specifically, we use the shortest-path distance matrix derived from the graph for base-point selection, compute persistence diagrams using the same shortest-path distance matrix, and configure DBSCAN with a precomputed distance metric. The persistence diagram shown in Fig. 20(A) reveals the presence of nontrivial 1D loops in the Coauthor dataset. Since no prior structural information is available for this dataset, we cluster the data points based on their connections to the four highest-degree nodes. As shown in Fig. 20(B-F), AdaHIsomap provides the clearest visualization of the loopy structures in the resulting 2D embeddings. Examining the clustering distributions further suggests that the embeddings produced by AdaHIsomap and Isomap are more coherent than those of the other methods, as points within the same cluster remain relatively compact. In contrast, the green, red, and yellow clusters are fragmented into multiple disconnected components in the t-SNE, UMAP, and TopoMap embeddings.

Overall, our experiments do not claim that AdaHIsomap universally outperforms all existing DR methods. Rather, they demonstrate that our approach provides a principled mechanism for preserving homological structure in a manner that is both complementary to and competitive with existing approaches.

E. Evaluation Metrics

We employ three evaluation metrics: one measuring geometric fidelity and two measuring the preservation of 0D and 1D topological features.

1) Geometric Evaluation: To assess geometric preservation, we use the root mean squared error (RMSE), referred to as *metric distortion*, which measures discrepancies between pairwise distances in the original space and the embedded space. Lower values indicate better geometric preservation. Let $X = \{x_i\}_{i=1}^n$ denote the original data points, and let $Y = \{y_i\}_{i=1}^n$ denote their corresponding 2D embedded points, where $y_i \in \mathbb{R}^2$. The metric is defined as:

$$\mathcal{D}(X,Y) = \sqrt{\frac{1}{n}\sum_{1\le i<j\le n}\left(\|x_i - x_j\|_2 - \|y_i - y_j\|_2\right)^2}.$$

2) Topological Evaluation: To evaluate topological faithfulness, we use the 2-*Wasserstein distance* between the persistence diagrams of the original and embedded spaces, defined as

$$\mathrm{PDW}^j(X,Y) = \left[\inf_{\eta:\mathrm{PD}_j(X)\to\mathrm{PD}_j(Y)}\sum_{x\in\mathrm{PD}_j(X)}||x-\eta(x)||_2^2\right]^{\frac{1}{2}}$$

We denote by PDW^0 and PDW^1 the distances between the 0D and 1D persistence diagrams, respectively.

VII. Conclusion and Discussion

In this paper, we demonstrate that the preservation of 0D and 1D homological features during dimensionality reduction can be achieved through a homological skeleton extracted

using an adaptive Mapper-based framework. Our work also opens several directions for future research.

First, the computational efficiency of both AdaMapper and AdaHIsomap could be further improved by accelerating the PD-induced segmentation process. Currently, this segmentation relies only on partial information extracted from the persistence diagram. Consequently, the development of algorithms that provide only the persistence information necessary for segmentation—rather than computing the full persistence diagram—could significantly reduce computational cost and benefit both methods.

Second, due to the limitations of the kNN graph construction, neither AdaMapper nor AdaHIsomap can effectively handle datasets containing multiple manifolds. For example, our approach cannot be directly applied to a dataset containing two disconnected faces, such as the example shown in Fig. 19. One possible direction is to first identify the underlying manifolds and then construct separate kNN graphs for each component. The main challenge would then be how to construct a skeleton that captures the global structure of the entire dataset while remaining faithfully representable in low-dimensional space.

Finally, a key remaining challenge is the development of a theory-driven framework for parameter selection in AdaMapper. Although we provide an automated strategy that performs well empirically, most parameter choices are still guided by heuristic observations and practical experience. Establishing rigorous mathematical foundations for parameter selection would provide stronger theoretical justification and improve the robustness and interpretability of the framework. Recent research offers promising evidence toward this direction [29], [31].

Overall, this work demonstrates that persistent homology can serve not only as a post hoc analysis tool, but also as an active mechanism for guiding algorithmic structure in dimensionality reduction. We believe this perspective opens new opportunities for topology-informed pipeline design in the analysis and visualization of high-dimensional data.

Acknowledgment

This work was supported in part by NSF grant OAC-2313124 and DOE grant DE-SC0021015.

References

[1] J. H. Stock and M. W. Watson, "Macroeconomic forecasting using diffusion indexes," *Journal of Business & Economic Statistics*, vol. 20, no. 2, pp. 147–162, 2002.

[2] C. Cunningham and B. Sun, "Representation of high-dimensional cell morphology and morphodynamics in 2D latent space," *Physical Biology*, vol. 22, no. 3, 2025.

[3] D. Maljovec, B. Wang, V. Pascucci, P.-T. Bremer, and D. Mandelli, "Analyzing dynamic probabilistic risk assessment data through topology-based clustering," in *International Topical Meeting on Probabilistic Safety Assessment and Analysis (PSA)*, 2013.

[4] M. Joswiak, Y. Peng, I. Castillo, and L. H. Chiang, "Dimensionality reduction for visualizing industrial chemical process data," *Control Engineering Practice*, vol. 93, p. 104189, 2019.

[5] K. Pearson, "LIII. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[6] W. S. Torgerson, "Multidimensional scaling: I. Theory and method," *Psychometrika*, vol. 17, no. 4, pp. 401–419, 1952.

[7] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[8] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[9] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[10] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[11] S. Liu, D. Maljovec, B. Wang, P.-T. Bremer, and V. Pascucci, "Visualizing high-dimensional data: Advances in the past decade," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 3, pp. 1249–1268, 2017.

[12] L. Bou Dagher, D. Madern, P. Malbos, and C. Brochier-Armanet, "Persistent homology reveals strong phylogenetic signal in 3d protein structures," *PNAS nexus*, vol. 3, no. 4, p. 158, 2024.

[13] P. S. Guerrero, Y. Katsukawa, S. Toriumi, and D. O. Suárez, "Persistent homology analysis for solar magnetograms," *The Astrophysical Journal*, vol. 964, no. 1, p. 32, 2024.

[14] S. B. Green, A. Mintz, X. Xu, and J. Cisewski-Kehe, "Topology of our cosmology with persistent homology," *Chance*, vol. 32, no. 3, pp. 6–13, 2019.

[15] G. Carlsson, T. Ishkhanov, V. De Silva, and A. Zomorodian, "On the local behavior of spaces of natural images," *International journal of computer vision*, vol. 76, no. 1, pp. 1–12, 2008.

[16] S. Xia, "A topological analysis of high-contrast patches in natural images," *Journal of Nonlinear Sciences and Applications*, vol. 9, pp. 126–138, 2016.

[17] L. Yan, T. B. Masood, F. Rasheed, I. Hotz, and B. Wang, "Geometry aware merge tree comparisons for time-varying data with interleaving distances," *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[18] F. Lan, B. Gamelin, L. Yan, J. Wang, B. Wang, and H. Guo, "Topological characterization and uncertainty visualization of atmospheric rivers," *Computer Graphics Forum*, vol. 43, no. 3, p. e15084, 2024.

[19] L. Yan, T. B. Masood, R. Sridharamurthy, F. Rasheed, V. Natarajan, I. Hotz, and B. Wang, "Scalar field comparison with topological descriptors: Properties and applications for scientific visualization," *Computer Graphics Forum*, vol. 40, no. 3, pp. 599–633, 2021.

[20] M. E. Aktas, E. Akbas, and A. E. Fatmaoui, "Persistence homology of networks: methods and applications," *Applied Network Science*, vol. 4, no. 1, pp. 1–28, 2019.

[21] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[22] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform Manifold Approximation and Projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[23] L. Yan, Y. Zhao, P. Rosen, C. Scheidegger, and B. Wang, "Homology-preserving dimensionality reduction via manifold landmarking and tearing," Symposium on Visualization in Data Science at IEEE Visualization, 2018.

[24] G. Singh, F. Mémoli, and G. Carlsson, "Topological methods for the analysis of high dimensional data sets and 3D object recognition," *Eurographics Symposium on Point-Based Graphics*, vol. 22, 2007.

[25] M. M. B. Shiraj, M. Rahman, M. Al-Imran, M. Liza, M. M. Murshed, and N. Akhter, "Anomaly detection in financial time series data via mapper algorithm and dbscan clustering," *World Journal of Advanced Engineering Technology and Sciences*, vol. 13, pp. 70–084, 09 2024.

[26] D. Haşegan, C. Geniesse, S. Chowdhury, and M. Saggar, "Deconstructing the Mapper algorithm to extract richer topological and temporal features from functional neuroimaging data," *Network Neuroscience*, vol. 8, no. 4, pp. 1355–1382, 12 2024.

[27] C. F. Loughrey, P. Fitzpatrick, N. Orr, and A. Jurek-Loughrey, "The topology of data: opportunities for cancer research," *Bioinformatics*, vol. 37, no. 19, pp. 3091–3098, 07 2021.

[28] Z. Oulhaj, M. Carrière, and B. Michel, "Differentiable mapper for topological optimization of data representation," in *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[29] N. Chalapathi, Y. Zhou, and B. Wang, "Adaptive covers for mapper graphs using information criteria," in *2021 IEEE International Conference on Big Data*, 2021, pp. 3789–3800.

[30] Q.-T. Bui, B. Vo, H.-A. N. Do, N. Q. V. Hung, and V. Snasel, "F-mapper: A fuzzy mapper clustering algorithm," *Knowledge-Based Systems*, vol. 189, p. 105107, 2020.

[31] E. Alvarado, R. Belton, E. Fischer, K.-J. Lee, S. Palande, S. Percival, and E. Purvine, "G-Mapper: Learning a cover in the mapper construction," *SIAM Journal on Mathematics of Data Science*, vol. 7, no. 2, pp. 572–596, 2025.

[32] L. Scoccola, U. Lim, and H. A. Harrington, "Cover learning for large-scale topology representation," in *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

[33] D. Pelleg, A. W. Moore *et al.*, "X-means: Extending k-means with efficient estimation of the number of clusters." in *Proceedings of the Seventeenth International Conference on Machine Learning*, vol. 1. Stanford, CA, 2000, pp. 727–734.

[34] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[35] J. C. Bezdek, *Models for Pattern Recognition*. Boston, MA: Springer US, 1981, pp. 1–13.

[36] V. Silva and J. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," in *Advances in Neural Information Processing Systems*, S. Becker, S. Thrun, and K. Obermayer, Eds., vol. 15. MIT Press, 2002.

[37] H. Doraiswamy, J. Tierny, P. J. Silva, L. G. Nonato, and C. Silva, "TopoMap: A 0-dimensional homology preserving projection of high-dimensional data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 561–571, 2020.

[38] V. Guardieiro, F. I. De Oliveira, H. Doraiswamy, L. G. Nonato, and C. Silva, "TopoMap++: A faster and more space efficient technique to compute projections with topological guarantees," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 31, pp. 229–239, 2024.

[39] M. Moor, M. Horn, B. Rieck, and K. Borgwardt, "Topological Autoencoders," in *Proceedings of the 37th International Conference on Machine Learning*, H. D. III and A. Singh, Eds., vol. 119, 2020, pp. 7045–7054.

[40] M. Clémot, J. Digne, and J. Tierny, "Topological autoencoders++: Fast and accurate cycle-aware dimensionality reduction," *IEEE Transactions on Visualization and Computer Graphics*, 2025.

[41] Y. Shinagawa, T. Kunii, and Y. Kergosien, "Surface coding based on morse theory," *IEEE Computer Graphics and Applications*, vol. 11, no. 5, pp. 66–78, 1991.

[42] R. Ghrist, "Three examples of applied & computational homology," *Nieuw Archief voor Wiskunde (The Amsterdam Archive, Special issue on the occasion of the fifth European Congress of Mathematics)*, pp. 122–125, 2008.

[43] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer, "Stability of persistence diagrams," *Discrete and Computational Geometry*, vol. 37, no. 1, pp. 103–120, 2007.

[44] R. Ghrist, "Barcodes: The persistent topology of data," *Bullentin of the American Mathematical Society*, vol. 45, pp. 61–75, 2008.

[45] H. Edelsbrunner and J. Harer, *Computational Topology: An Introduction*. Providence, RI, USA: American Mathematical Society, 2010.

[46] B. A. Okediji, "Persistent homology and persistent cohomology: A review," *Earthline Journal of Mathematical Sciences*, vol. 14, no. 2, pp. 349–378, 2024.

[47] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno, "Reeb graphs for shape analysis and applications," *Theoretical Computer Science*, vol. 392, pp. 5–22, 2008.

[48] X. Ge, I. Safa, M. Belkin, and Y. Wang, "Data skeletonization via Reeb graphs," *Advances in Neural Information Processing Systems*, 2011.

[49] N. Saul and H. J. van Veen, "Mlwave/kepler-mapper: 186f (version 1.0.1). zenodo." November 2017.

[50] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, 1996.

[51] M. Čufar, "Ripserer.jl: flexible and efficient persistent homology computation in Julia," *Journal of Open Source Software*, vol. 5, no. 54, p. 2614, 2020.

[52] M. Čufar and Ž. Virk, "Fast computation of persistent homology representatives with involuted persistent homology," *Foundations of Data Science*, vol. 5, no. 4, pp. 466–479, 2023.

[53] L. Li, C. Thompson, G. Henselman-Petrusek, C. Giusti, and L. Ziegelmeier, "Minimal cycle representatives in persistent homology using linear programming: An empirical study with user's guide," *Frontiers in artificial intelligence*, vol. 4, p. 681117, 2021.

[54] N. Saul, "Extracting representative cycles from persistence diagrams," https://github.com/sauln/cyclonysus, 2018.

[55] V. De Silva, D. Morozov, and M. Vejdemo-Johansson, "Persistent cohomology and circular coordinates," *Discrete & Computational Geometry*, vol. 45, no. 4, pp. 737–759, 2011.

[56] R. López-Sastre, A. García-Fuertes, C. Redondo-Cabrera, F. Acevedo-Rodríguez, and S. Maldonado-Bascón, "Evaluating 3D spatial pyramids for classifying 3D shapes," *Computers & Graphics*, vol. 37, no. 5, pp. 473–483, 2013.

[57] C. Redondo-Cabrera, R. J. López-Sastre, J. Acevedo-Rodriguez, and S. Maldonado-Bascón, "SURFing the point clouds: Selective 3D spatial pyramids for category-level object recognition," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3458–3465.

[58] B. Li, A. Godil, A. Tatsuma, T. Yanagimachi, S. Zhang, M. Aono, X. Bai, T. Furuya, L. Li, R. López-Sastre, H. Johan, R. Ohbuchi, and C. Redondo-Cabrera, "SHREC'12 Track: Generic 3D Shape Retrieval," in *Eurographics Workshop on 3D Object Retrieval*, M. Spagnuolo, M. Bronstein, A. Bronstein, and A. Ferreira, Eds. The Eurographics Association, 2012.

[59] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The princeton shape benchmark," in *Proceedings Shape Modeling Applications, 2004.*, 2004, pp. 167–178.

[60] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, *Numerical geometry of non-rigid shapes*. Springer Science & Business Media, 2008.

[61] R. W. Sumner and J. Popović, "Deformation transfer for triangle meshes," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 399–405, 2004.

[62] B. Falcidieno, "Bringing the semantics into digital shapes: the AIM@SHAPE approach," in *Eurographics Italian Chapter Conference*, R. D. Amicis and G. Conti, Eds. The Eurographics Association, 2007.

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

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

[65] B. L. Smarr, I. Zucker, and L. J. Kriegsfeld, "Detection of successful and unsuccessful pregnancies in mice within hours of pairing through frequency analysis of high temporal resolution core body temperature data," *PLoS One*, 2016.

[66] E. R. Gansner, Y. Koren, and S. C. North, "Topological fisheye views for visualizing large graphs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 4, pp. 457–468, 2005.

[67] I. Duff, R. Grimes, and J. Lewis, "Sparse matrix problems," *ACM Transactions on Mathematical Software*, vol. 14, no. 1, pp. 1–14, 1989.

[68] giphy.com, "Sad The Simpsons GIF," https://giphy.com/gifs/sad-the-simpsons-lonely-dACqNmvAfY12M, 2013.

[69] 3dscanstore.com, "Free 3D Head Model," https://www.3dscanstore.com/blog/Free-3D-Head-Model, 2023.

[70] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical Review E*, vol. 74, p. 036104, 2006.

[71] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux, "The Topology ToolKit," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 1, pp. 832–842, 2017.

[72] J. Ahrens, B. Geveci, and C. Law, "ParaView: An end-user tool for large data visualization," *Visualization Handbook*, 2005.

[73] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

APPENDIX A
PARAMETER SELECTION FOR THE DBSCAN ALGORITHM BASED ON PD-INDUCED SEGMENTATION

Although the choice of clustering algorithm is not central to our method, we adopt DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [50] because of its ability to detect clusters of arbitrary shape and identify noise in spatial datasets. In its original formulation, successful DBSCAN clustering requires suitable choices of the parameters ϵ (the neighborhood radius) and $minPts$ (the minimum number of points needed to form a dense region), as well as at least one representative point from each cluster. However, in practice, there is no straightforward way to automatically determine appropriate parameter values for all clusters [50].

However, our experiments indicate that, once a suitable value of ϵ is chosen, the clustering results are relatively insensitive to the choice of $minPts$. To simplify parameter tuning, we therefore fix $minPts = 1$ for all datasets. We then propose an automatic strategy for estimating ϵ for each point cloud, while still allowing for optional manual adjustment. The procedure is described as follows.

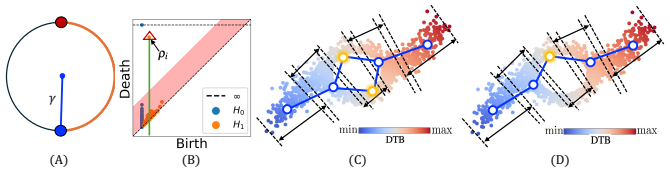

Fig. 21. Main idea for selecting ϵ. (A) Intuition behind the choice of ϵ. (B) Persistence diagram $\mathrm{Dgm}(X)$. (C) Point cloud X colored by the DTB function $f(X)$, showing the DBSCAN clustering result in the loop region for $\epsilon \approx \rho_i[1]/1.6$, with the identified cluster highlighted in orange. (D) Point cloud colored by the DTB function, showing the DBSCAN clustering result in the loop region for $\epsilon > \rho_i[1]/1.6$, with the identified cluster highlighted in orange.

Automatic estimation of ϵ. We automatically estimate ϵ for each point cloud, while still allowing users to override the value if desired. Let T denote the simplification threshold. Among all 1D homological features with persistence greater than T, we select the feature with the smallest persistence and denote its death time by $\rho_i[1]$. We then define $\epsilon = \frac{\rho_i[1]}{1.6}$. The scaling factor 1.6 is motivated by both geometric intuition and empirical observations. Consider an ideal circular loop of radius γ covered by three cubes. To preserve the loop structure during segmentation, the middle cube should contain at least two distinct clusters (see Fig. 21(A)), which requires ϵ to be sufficiently small for density-based clustering to separate points within that region.

We estimate γ using the death time $\rho_i[1]$ of the selected loop feature. Experiments on noisy synthetic loops suggest that the relation $\gamma \approx \frac{\rho_i[1]}{1.6}$ provides a reliable balance between preserving the loop structure and avoiding excessive over-segmentation. In particular, this choice of ϵ typically yields at least two distinct clusters in the central loop region (see Fig. 21(C)) without introducing an excessive number of additional clusters.

APPENDIX B
DIMENSIONALITY REDUCTION PARAMETERS

We present the parameter configurations used for the experimental and evaluation results on 11 datasets. Unless otherwise specified, AdaHIsomap uses the following default settings: simplification threshold $T = 0.35$, overlap percentage $p = 0.2$, minimum number of points $minPts = 1$, and number of nearest neighbors NN. For datasets with $N \geq 1000$, we set $NN = 10$; for datasets with $N < 1000$, we set $NN = 8$; see Table II.

TABLE II
ISOMAP, T-SNE AND UMAP PARAMETER SETTINGS. NN DENOTES THE NUMBER OF NEIGHBORS, n DENOTES THE NUMBER OF COMPONENTS.

	Parameter	Swiss Hole	Glasses	Fertility	Octa	Bcsstk31	4elt	Mice	Face	Vortex Street	Coauthor	Cartoon
Isomap	n	2	2	2	2	2	2	2	2	2	2	2
	NN	20	15	15	5	10	15	8	8	6	17	9
t-SNE	n	2	2	2	2	2	2	2	2	2	2	2
	$perplexity$	50	50	50	40	30	30	40	40	5	30	30
	max_iter	1000	5000	1000	1000	1000	5000	5000	1000	5000	5000	1000
UMAP	n	2	2	2	2	2	2	2	2	2	2	2
	NN	35	35	10	30	30	15	25	30	15	40	20
	min_dist	0.5	0.1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.3

We omit TopoMap and TopoAE++ from the parameter table because, for all datasets, we used the existing TTK (Topology ToolKit) [71] implementations with default parameters through ParaView 5.13 [72]. Specifically, for TopoMap, we used the Kruskal MST algorithm with Angular Samples = 2, while for TopoAE++, we used 1000 training epochs with hidden layer dimensions 128 and 32.

For Isomap and t-SNE, we used the open-source implementations available in scikit-learn [73], and for UMAP we used the `umap-learn` implementation [22], both available in Python. Finally, for AdaMapper, we used our Python implementation.

APPENDIX C
ADDITIONAL EXPERIMENTAL RESULTS

The TopoMap embeddings are shown in Fig. 22, and the quantitative analysis is summarized in Table II.

As illustrated in Fig. 22, TopoMap effectively preserves 0D homological features but is less successful at preserving 1D features. Table II further shows that, across the 11 datasets, the relative performance of the methods varies depending on the evaluation metric. In terms of RMSE, AdaHIsomap ranks among the top two performers and often outperforms Isomap and TopoMap on datasets such as Glasses, Mice, Face, and Vortex Street, while UMAP performs particularly well on datasets such as Octa.

For WD1, which measures preservation of 1D homology, AdaHIsomap achieves the best result on 4elt, while TopoAE++ performs best on Fertility. t-SNE and UMAP also achieve top performance on several individual datasets, such as Face and Vortex Street. For WD0, which evaluates preservation of 0D homology, TopoMap and UMAP dominate across multiple datasets. In particular, TopoMap performs strongly on

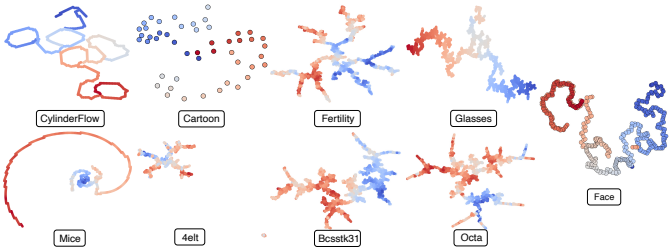

Fig. 22. TopoMap embeddings for all remaining datasets in Sec. VI.

Mice, Glasses, and Coauthor, while UMAP achieves the best results on Octa, Bcsstk31, and 4elt.

Overall, AdaHIsomap demonstrates robust and well-balanced performance across all evaluation metrics.