

# Skeletonisation Algorithms for Unorganised Point Clouds with Theoretical Guarantees

P. Smith<sup>a</sup>, V. Kurlin<sup>a,\*</sup>

<sup>a</sup>*Department of Computer Science, University of Liverpool, Liverpool L69 3BX, UK*

## Abstract

Real datasets often come in the form of an unorganised cloud of points. An overall shape of such data is hard to visualise when data points are given by coordinates in a high-dimensional Euclidean space. More general data can be represented by an abstract graph with weights or distances on links between data points. The skeletonisation problem for a point cloud is to find a graph or a skeleton that correctly represents the shape of a cloud. This paper compares three algorithms that solve the data skeletonisation problem for a general cloud with topological and geometric guarantees. First, the Mapper algorithm outputs a network of interlinked clusters. Second, the  $\alpha$ -Reeb algorithm discretises the classical Reeb graph and can be applied to discrete clouds at different scales  $\alpha$ . The third algorithm HoPeS produces a Homologically Persistent Skeleton without any extra parameters. HoPeS represents the 1-dimensional shape of a cloud at any scale by the Optimality Theorem. The Reconstruction Theorem gives conditions on a noisy sample of a graph when HoPeS provides a reconstructed graph with a correct homotopy type and within a small offset of the sample. The experiments on synthetic and real data show that HoPeS outperforms the other algorithms on several topological and geometric measures.

*Keywords:* topological data analysis, computational geometry, skeletonisation

*2010 MSC:* 97P99

\*Corresponding author

Email address: [vkurlin@liverpool.ac.uk](mailto:vkurlin@liverpool.ac.uk), <http://kurlin.org> (V. Kurlin)

## 1. Introduction: Motivations, Problem Statement and Contributions

Data often comes in the form of an unorganised point cloud, namely a finite set of points equipped with a pairwise distance function. It can be difficult to extract meaningful information from these clouds without automatic tools. One needs to correctly represent any unorganised cloud by a hierarchical skeleton or a graphical network in order to speed up higher level processing such as pattern detection or classification. This problem of data skeletonisation is an important step in understanding and interpreting the data. Indeed, an algorithm with theoretical guarantees will allow the visualisation of topological structures and the extraction of invariants that are implicitly present in the point cloud.

The data skeletonisation problem is, when given only a noisy point cloud  $C$  sampled from a graph  $G$  in a metric space, to produce a reconstructed graph  $G'$  that is geometrically and topologically similar to  $G$ . The geometric and topological similarity will be rigorously stated at the end of section 2, which first introduces necessary concepts. Briefly, topological similarity requires that the reconstructed graph  $G'$  can be continuously deformed to the original graph  $G$ . Geometric similarity will mean that  $G$  and  $G'$  are close to each other with respect to a distance. For example,  $G'$  should be in a small offset of  $G$  and vice versa.

Below we list the contributions of this paper.

- The paper reviews three algorithms (Mapper,  $\alpha$ -Reeb, HoPeS), which solve the skeletonisation problem with theoretical guarantees. Sections 4, 5, 6 describe these algorithms and their input parameters.
- Optimality and Reconstruction Theorems for a Homologically Persistent Skeleton (HoPeS) are proved for the first time in sections 7 and 8.
- All three algorithms are extensively compared in section 10 by their run-time, as well as by their topological and geometric errors on the same datasets whose generation is explained in section 9.

## 2. Basic Definitions and Results from Geometry and Topology

**Definition 2.1** (point cloud). We define a *point cloud* to be a finite metric space. Namely, a finite collection of points with a pairwise distance  $d(p_1, p_2)$  between any two points  $p_1, p_2$  that satisfies the metric axioms:

- Positiveness:  $d(p_1, p_2) \geq 0$  with equality iff  $p_1 = p_2$ .
- Symmetry:  $d(p_1, p_2) = d(p_2, p_1)$
- The triangle inequality:  $d(p_1, p_2) + d(p_2, p_3) \geq d(p_1, p_3)$ .

**Definition 2.2** (metric graph and neighbourhood graph). We define a *graph* to be a set of vertices with edges being unordered pairs of vertices.

A *metric graph* is a graph that has a length assigned to each edge. Therefore, if there exists a path between two vertices, the distance between these vertices will be the minimum total length of any path from one vertex to the other.

A *neighbourhood graph* of a point cloud  $C$  with threshold  $\epsilon$ , denoted as  $N(C, \epsilon)$ , is an example of a metric graph. It is defined to be a graph whose vertex set is all points in  $C$ , with edges existing between pairs of vertices if the pairwise distance between the two vertices is less than the specified threshold  $\epsilon$ . The length of each edge is equal to the pairwise distance of the two vertices it connects.

**Definition 2.3** (graph homeomorphism). Two graphs  $G_1, G_2 \neq S^1$  are said to be *homeomorphic*,  $G_1 \simeq G_2$ , if, ignoring all vertices of degree 2, there exists a bijection  $\psi$  between the vertices of  $G_1$  and  $G_2$  that respects edges. By respecting edges, we mean that if there are  $m$  edges between  $v_1$  and  $v_2$  in  $G_1$ , then there will be  $m$  edges between  $\psi(v_1)$  and  $\psi(v_2)$  in  $G_2$ .

**Definition 2.4** (simplicial complex). A simplicial complex  $Q$  with a vertex set  $V$  is a collection of finite subsets  $\{v_0, \dots, v_k\} \subset V$  (called *simplices*) such that

- any subset (called a *face*) of a simplex is also a simplex and is included in  $Q$ ;
- any non-empty intersection of simplices is their common face included in  $Q$ .

Any simplex on  $k + 1$  vertices has this geometric realisation:

$$\Delta^k = \{(t_0, t_1, \dots, t_k) \in \mathbb{R}^k \mid t_0 + t_1 + \dots + t_k = 1, t_i \geq 0\} \subset \mathbb{R}^k.$$

We define the *geometric realisation* of any simplicial complex  $Q$  by gluing realisations of all its simplices along their common faces. Hence  $Q$  inherits the Euclidean topology. The dimension of a simplex spanned by  $k+1$  vertices is  $k$ . The dimension of a complex  $Q$  is the maximum dimension of its simplices.

**Definition 2.5** (Čech complex). Let  $C$  be any set in a metric space  $M$ . The *ambient Čech complex*  $\check{C}h(C, M; \alpha)$  has a simplex on  $v_0, \dots, v_k \in C$  if the full intersection of  $k+1$  closed balls with radius  $\alpha$  and centres  $v_0, \dots, v_k$  has a point from the ambient space  $M$ , see [1, section 4.2.3].

**Lemma 2.6.** [2, Corollary 4G.3] Let  $C$  be a subspace of a metric space  $M$ . Then any  $\alpha$ -offset  $C^\alpha \subset M$  is homotopy equivalent to the ambient Čech complex  $\check{C}h(C, M; \alpha)$ .

**Definition 2.7** (Vietoris-Rips complex). Again, let  $C$  be any set in a metric space  $M$ . The *ambient Vietoris-Rips complex*  $VR(C, M, \alpha)$  has a simplex on  $v_0, \dots, v_k \in C$  if each pairwise intersection of the  $k+1$  closed balls with radius  $\alpha$  and centres  $v_0, \dots, v_k$  has a point from the ambient space  $M$ .

**Definition 2.8** (level set). For a space  $X$  and a function  $f : X \rightarrow \mathbb{R}$ , we define the level set of  $f$  corresponding to a value  $c \in \mathbb{R}$ ,  $L_c(f)$ , to be the set of points  $L_c(f) = \{x \in X \mid f(x) = c\}$ .

**Definition 2.9** (Reeb graphs). Let  $(X, f_X)$  be a simplicial complex  $X$  with a continuous function  $f_X : X \rightarrow \mathbb{R}$ . We define, for points  $x, y \in X$ , the equivalence relation  $\sim$  such that  $x \sim y$  if and only if  $f(x) = f(y)$  and  $x$  and  $y$  are in the same connected component of the level set of  $f(x)$ .

We define the Reeb graph  $G$  for  $(X, f_X)$  to be the space formed from  $X$  by mapping all points that are equivalent under the equivalence relation  $\sim$  to a single point. Namely, if two points  $x, y \in X$  are such that  $f(x) = f(y)$  and  $x$  and  $y$  are in the same connected component of the level set of  $d(x)$ , then  $x$  and  $y$  will be mapped to the same point in  $G$ .

HoPeS, one of the skeletonisation algorithms introduced later in the paper, uses homology, and so we introduce this concept here. A 1-dimensional cycle in a

complex  $Q$  is a sequence of edges  $e_1, \dots, e_k$  such that  $e_i$  and  $e_{i+1}$  have a common endpoint, where  $e_{k+1} = e_1$ . Below we define the first homology group  $H_1(Q)$  of a simplicial complex  $Q$  only with coefficients in the group  $\mathbb{Z}_2 = \mathbb{Z}/2\mathbb{Z} = \{0, 1\}$ .

**Definition 2.10** (homology  $H_1(Q)$  of a complex). Cycles of a complex  $Q$  can be algebraically written as finite linear combinations of edges (with coefficients 0 or 1) and generate the vector space  $C_1$  of cycles. The boundaries of all triangles in  $Q$  are cycles of 3 edges and generate the space  $B_1 \subset C_1$ . The quotient space  $C_1/B_1$  is the *homology* group  $H_1(Q)$ . The operation is the addition of cycles, the empty cycle is the zero. We define the  $k$ th Betti number of a complex  $Q$  to be the rank of the  $k$ th homology group. For instance, the first Betti number of a complex  $Q$  is the rank of the first homology group, and is equivalent to the number of linearly independent 1-dimensional holes in  $Q$ .

The sphere  $S^2$  has trivial first homology group,  $H_1(S^2) = 0$ , while the torus  $T$  has  $H_1(T) = \mathbb{Z}_2 \oplus \mathbb{Z}_2$ . Even if a simplicial complex  $Q$  is infinite, say the Vietoris-Rips complex of an infinite set,  $H_1(Q)$  is well-defined in terms of finite linear combinations of edges.

### 3. Review of Related Past Work on Data Skeletonisation Algorithms

**Delaunay-based skeletons.** R. Singh et al. [3] approximated a skeleton of a shape by a subgraph of a Delaunay triangulation using 3 thresholds:  $K$  for the minimum number of edges in a cycle and  $\delta_{min}, \delta_{max}$  for inserting/merging Voronoi regions. Similar parameters are used in [4].

**Skeletonization via Reeb graphs.** Starting from a noisy sample  $C$  of an unknown graph  $G$  with a scale parameter, X. Ge et al. [5] considered the Reeb graph of the Vietoris-Rips complex on a cloud  $C$  at a given scale  $\alpha$ . The Reeb graph is not intrinsically embedded into any space even if  $C \subset \mathbb{R}^2$ . The reconstruction in [5, Theorem 3.1] outputs a graph with a correct homotopy type, while all our derived skeletons  $\text{HoPeS}_{k,l}(C)$  also give close geometric approximations in the  $2\epsilon$ -offset of an unknown graph  $G$ .

**Metric graph reconstruction.** M. Aanjaneya et al. [6] studied a related problem approximating a metric on a large input graph  $Y$  by a metric on a small output graph  $\hat{X}$ . If  $Y$  is a good  $\epsilon$ -approximation to an unknown graph  $X$ , then [6, Theorem 2] guarantees the existence of a homeomorphism  $X \rightarrow \hat{X}$  that distorts the metrics on  $X$  and  $\hat{X}$  with a multiplicative factor  $1 + c\epsilon$  for  $c > \frac{30}{b}$ , where  $b > 14.5\epsilon$  is the length of a shortest edge of  $X$ . According to [5], the algorithm may not run on all inputs, but only for carefully chosen parameters. The skeletons  $\text{HoPeS}_{k,l}(C)$  are well-defined for any cloud  $C$  and  $k, l \geq 1$ .

**Stability of 1-dimensional Mapper.** Mathieu Carrière et al. [7, Theorem 5.2] found an explicit connection between Mapper and the Reeb graph through an intermediate object called the MultiNerve Mapper. This enabled them to produce stability guarantees for the persistence diagram of the Mapper output, similar to that of the general stability theorem of extended persistence diagrams, bounding the amount of perturbation of the persistence diagram of the output graph by an expression related to the amount of perturbation of the filter function  $f$ .

**Filamentary structures using Reeb-type graphs.** F. Chazal et al. [8] defined the  $\alpha$ -Reeb graph  $G$  of a metric space  $X$  at a user-defined scale  $\alpha$ . If  $X$  is  $\epsilon$ -close to an unknown graph with edges of minimum length  $8\epsilon$ , then the output  $G$  is  $34(\beta(G) + 1)\epsilon$ -close to the input  $X$ , where  $\beta(G)$  is the first Betti number of  $G$ , see [8, Theorem 4.9]. The algorithm has the very fast time  $O(n \log n)$  for  $n$  points in  $X$  and needs the scale  $\alpha$ .

**Graph Induced Complex GIC.** T. Dey et al. [9] built GIC depending on a scale  $\alpha$  and a user-defined graph that spans a cloud  $C$ . If  $C$  is an  $\epsilon$ -sample of a good manifold, then GIC has the same homology  $H_1$  as the Vietoris-Rips complex on  $C$  at scales  $\alpha \geq 4\epsilon$ . Theorem 8.9 describes graphs  $G$  that can be geometrically and topologically approximated from any  $\epsilon$ -sample  $C$  without extra input parameters.

**Skeleton for  $\alpha$ -offsets in  $\mathbb{R}^2$ .** This work extends [10, 11, 12] from locating holes in 2D clouds to a full skeleton. The Gap Search method in Section 6.2

vastly improves [13, Theorem 7], which was stated for one subskeleton under stronger assumptions on a graph  $G \subset \mathbb{R}^2$ .

**The key advantage** of a Homologically Persistent Skeleton  $\text{HoPeS}(C)$  is its universal scale-independent structure. In comparison with the persistence diagram of isolated dots (homology classes),  $\text{HoPeS}(C)$  shows all persistent cycles directly on a cloud  $C$ . In comparison with all algorithms that require a scale  $\alpha$ , the skeleton  $\text{HoPeS}(C)$  contains a hierarchy of derived skeletons  $\text{HoPeS}_{k,l}(C)$  independent of  $\alpha$ .

**The derived skeletons** are most persistent subgraphs of  $\text{HoPeS}(C)$  depending on integer indices  $k, l \geq 1$ , which are easier to choose a posteriori rather than a continuous scale  $\alpha$  a priori. We may start from the ‘simplest guess’  $k = l = 1$  and then try  $k = 2, l = 1$  without re-running the algorithm, but only selecting a different subgraph of  $\text{HoPeS}(C)$ .

#### 4. The Mapper Algorithm

The Mapper algorithm is a computational method introduced by G. Carlsson et al. [14] that aims to give a simple description of a dataset in the form of a simplicial complex. It reveals interactions between partial clusters of preimages of a function from the dataset to a given parameter space. Typically, Mapper takes as **input a point cloud  $C$  along with additional parameters**, and **outputs a simplicial complex**. The Mapper algorithm requires:

- **A filter function:** This is a function  $f : C \rightarrow Y$  whose value is known for all points in the dataset. Typically, the parameter space  $Y$  will be  $\mathbb{R}$ , but could also be other spaces such as  $\mathbb{R}^2$  or  $S^1$ . The type of filter chosen is up to the user, with common examples being a density estimator, the Euclidean distance from a root point, or the distance from a root point within a neighbourhood graph of the dataset. It is likely that in order for a good choice of filter function to be selected, an existing knowledge of the dataset being studied is required, which is therefore a drawback of Mapper.

- **A covering of the range of  $f$ :** The range of the filter function  $f$  must be divided into overlapping regions. For example, if the parameter space is  $\mathbb{R}$ , we divide the range of the filter function into overlapping intervals. It is fundamental to the process that the sets in the covering overlap with adjacent sets. This gives the user two additional parameters – number of intervals and amount of overlap – which can be used to control the resolution of the output simplicial complex.
- **A clustering algorithm:** The process requires subsets of the input point cloud to be clustered. Therefore, a clustering algorithm must be chosen by the user.

The Mapper algorithm is now described below:

- **Stage 1:** All points of  $C$  are mapped to the parameter space by the chosen filter function.
- **Stage 2:** For each set in the covering of the range of the filter function, we consider the set of points in the preimage of this set, and cluster these points according to the given clustering algorithm. We call this partial clustering. Each cluster is represented as a vertex in the output complex.
- **Stage 3:** The final step is to incorporate the interactions of the clusters into the output complex . Specifically, an edge is added between two vertices in the output complex if the intersection of the two clusters represented by the vertices is non-empty. Namely, there is at least one point of  $C$  that is a member of both clusters being considered. It is possible for complexes of a dimension higher than 1 to be included. For example, if the parameter space of the filter function is  $\mathbb{R}^2$ , it is possible to have two-dimensional simplices in the output complex, which would be included if the intersection of three clusters is non-empty. We end up with a simplicial complex which, if the additional parameters have been chosen well, will capture topological features of the dataset.

Mapper is a versatile tool that if used rightly can be a useful method in visualising large datasets. A significant drawback to the method however is that with so many choices to be made, the user often requires existing knowledge of the dataset in order to select parameters that will give meaningful outputs.

In our experiments in Section 10, we use a filter function mapping points in a point cloud to  $\mathbb{R}$ , and therefore Mapper outputs a one-dimensional complex which equally can be considered as a graph. Specifically, we simply use the Euclidean distance from an extremal point. We fixed the overlap percentage of two adjacent intervals in the covering to be 50%, and used DBSCAN as our clustering algorithm. This left us with two parameters – the number of intervals, and a parameter  $\epsilon$  used in the DBSCAN clustering algorithm – that we optimised in order to achieve the best results.

#### 4.1. DBSCAN

DBSCAN is a well-used clustering algorithm introduced in [15]. It’s main advantage for our purposes compared to  $k$ -means clustering is that it does not require the user to specify the number of clusters beforehand. DBSCAN requires two parameters:  $\epsilon$  – which is the radius around a point within which we search for neighbours; and minPts – the number of points required within a neighbourhood of a point before a cluster can be formed. The algorithm works as follows:

- **Stage 1:** A single point  $p_1$  is selected at random, and we compute the set  $\text{Nbhd}(p_1)$  of all points within a distance  $\epsilon$  of  $p_1$ . If  $|\text{Nbhd}(p_1)| < \text{minPts}$ , then  $p_1$  is labelled as noise. If however  $|\text{Nbhd}(p_1)| \geq \text{minPts}$ , we label all points in  $\text{Nbhd}(p_1)$  (not already belonging to another cluster, but may have previously been labelled as noise) as belonging to the cluster of  $p_1$ .
- **Stage 2:** Then, looping over all points  $p_i$  in  $\text{Nbhd}(p_1)$  (apart from  $p_1$ ), we compute  $\text{Nbhd}(p_i)$ . All points in  $\text{Nbhd}(p_i)$  (not already belonging to another cluster) are labelled as belonging to the cluster of  $p_1$ , and, if

$|\text{Nbhd}(p_i)| \geq \text{minPts}$ , we add all points in  $\text{Nbhd}(p_i)$  to  $\text{Nbhd}(p_1)$ . Hence, this loop will finish only when all points in  $p_1$ 's cluster have been identified.

- **Stage 3:** A new point  $p_2$ , that is not already labelled, is selected, and we continue as in the first two stages for  $p_2$  instead of  $p_1$ . We continue in this way until all points are assigned a cluster or are labelled as noise.

In our implementation, we found it acceptable to fix  $\text{minPts}$  at 5, but deemed it necessary to optimise  $\epsilon$  (see Section 10.2).

## 5. The $\alpha$ -Reeb Algorithm

A Reeb graph, introduced in Definition 2.9, is a simplified representation of a simplicial complex. If, instead of having an entire simplicial complex  $X$ , we only have a finite number of points sampled from  $X$ , even approximating the Reeb graph of  $(X, f_X)$  is not straightforward. Therefore, to bridge this barrier, we introduce a variant of the Reeb graph, known as the  $\alpha$ -Reeb graph.

**Definition 5.1** ( $\alpha$ -Reeb graphs). Let  $(X, f_X)$  be a simplicial complex  $X$  with a continuous function  $f_X : X \rightarrow \mathbb{R}$ . Let  $\alpha > 0$  and let  $\mathcal{I} = \{I_i\}$  be a covering of the range of the function  $f$  where each  $I_i$  is an open interval of length  $\alpha$ . We define the equivalence relation  $\sim_\alpha$  such that  $x \sim_\alpha y$  if and only if  $d(x) = d(y)$  and  $x$  and  $y$  are in the same connected component of  $d^{-1}(I_i)$  for some interval  $I_i \in \mathcal{I}$ . Here,  $d^{-1}(I_i) = \{x \in X \mid c \in I_i, x \in L_c(f)\}$ . (Notice how this last condition differs to the definition of a Reeb graph.)

We define the  $\alpha$ -Reeb graph,  $\tilde{G}$ , associated to a covering  $\mathcal{I}$  of a simplicial complex  $X$  to be the space formed from  $X$  by mapping all points that are equivalent under the equivalence relation  $\sim_\alpha$  to a single point.

Point clouds are finite metric spaces, so trying to obtain  $\alpha$ -Reeb graphs of point clouds is pointless. However, we can produce an  $\alpha$ -Reeb graph of the underlying metric space of the point cloud if we take instead a neighbourhood graph of the point cloud. This is done in practise via the  $\alpha$ -Reeb algorithm, which takes as **input a connected metric space** and **outputs a metric**

**graph.** The stages of the  $\alpha$ -Reeb algorithm are listed below, where stage 0 shows how we can get a connected metric space from a point cloud.

- Stage 0: We create a neighbourhood graph,  $N(C, \epsilon)$ , for  $C$ . This is a graph where all points at a pairwise distance less than a specified threshold  $\epsilon$  are connected by an edge.
- Stage 1: We select an extremal vertex in our neighbourhood graph as our root vertex, and for each vertex, we calculate the distance that it is from the root vertex within the graph. We denote this function as  $f : V \rightarrow \mathbb{R}$ , where  $V$  is the set of vertices of  $N(C, \epsilon)$ .
- Stage 2: Let  $M$  be the maximal value of  $f$ . The range of the function  $f$  is split into the following overlapping covering:  $\mathcal{I} = \{I_i\}_{0 \leq i \leq m}$ , where  $I_i = \{[\frac{i\alpha}{2}, \frac{i\alpha}{2} + \alpha]\}$  and  $m$  is the smallest integer such that  $m \geq \frac{2(M-\alpha)}{\alpha}$ .
- Stage 3: For each interval in the covering, we consider its preimage (a set of vertices), and include an edge between two vertices in the preimage if an edge also existed between the two vertices in the neighbourhood graph. Therefore, each preimage is a (possibly disconnected) subgraph of the neighbourhood graph. Each connected component corresponds to a vertex in an intermediate graph.
- Stage 4: In a similar way to Mapper, we add an edge between two vertices in our intermediate graph if the intersection of their corresponding connected components is non-empty,
- Stage 5: In the final step, for each vertex in our intermediate graph, we take a copy of the interval  $I_i$ . We obtain the  $\alpha$ -Reeb graph as the quotient of the disjoint union of these copies of intervals. Specifically, we split each interval in half, and identify the top half of an interval corresponding to a vertex  $v$  with the bottom half of all higher intervals that correspond to a vertex that shares an edge in the intermediate graph with  $v$ .

In our implementation, when creating a neighbourhood graph, we took our threshold  $\epsilon$  to be the maximum death of dots above the first widest gap in the persistence diagram  $\text{PD}\{C^\alpha\}$ . We tried to optimise the parameter  $\alpha$ , which we expand on in Section 10.2.

## 6. HoPeS: A Homologically Persistent Skeleton of a Point Cloud

The algorithm HoPeS – a Homologically Persistent Skeleton – uses persistent homology to identify the dominant features implicit in a point cloud.

### 6.1. Persistent Homology

**Definition 6.1** (births and deaths). For any filtration  $\{Q(C; \alpha)\}$  of complexes on a cloud  $C$  in a metric space, a homology class  $\gamma \in H_1(Q(C; \alpha_i))$  is *born* at  $\alpha_i = \text{birth}(\gamma)$  if  $\gamma$  is not in the full image under the induced homomorphism  $H_1(Q(C; \alpha)) \rightarrow H_1(Q(C; \alpha_i))$  for any  $\alpha < \alpha_i$ . The class  $\gamma$  *dies* at  $\alpha_j = \text{death}(\gamma) \geq \alpha_i$  when the image of  $\gamma$  under  $H_1(Q(C; \alpha_i)) \rightarrow H_1(Q(C; \alpha_j))$  merges into the full image under  $H_1(Q(C; \alpha)) \rightarrow H_1(Q(C; \alpha_j))$  for some  $\alpha < \alpha_i$ .

**Definition 6.2** (persistence diagram). Fix a filtration  $\{Q(C; \alpha)\}$  of complexes on a set  $C$ . Let  $\alpha_1, \dots, \alpha_m$  be all critical scales when the homology group  $H_1(Q(C; \alpha))$  changes. Let  $\mu_{ij}$  be the number of independent classes in  $H_1(Q(C; \alpha))$  that are born at  $\alpha_i$  and die at  $\alpha_j$ . The *persistence diagram*  $\text{PD}\{Q(C; \alpha)\} \subset \mathbb{R}^2$  is the multi-set consisting of all dots  $(\alpha_i, \alpha_j) \in \mathbb{R}^2$  with the multiplicities  $\mu_{ij} \geq 1$  and all diagonal dots  $(x, x)$  with infinite multiplicity.

For the cloud  $C$  in Fig. 1 and the filtration  $\{C^\alpha\}$ , the homology  $H_1$  has 2 classes that persist over  $1.5 \leq \alpha < R$  and  $2 \leq \alpha < R$ , where  $R = \frac{15}{8}\sqrt{17}$  is the circumradius of the largest Delaunay triangle with sides  $4, \sqrt{17}, 5$ . We use the approximate value  $R \approx 2.577$  for simplicity. Hence the persistence diagram  $\text{PD}\{C^\alpha\}$  has 2 off-diagonal red dots  $(1.5, 2.577)$  and  $(2, 2.577)$  in the last picture of Fig. 1.

We define the bottleneck distance  $d_B$  between persistence diagrams needed for Stability Theorem 6.4 below.

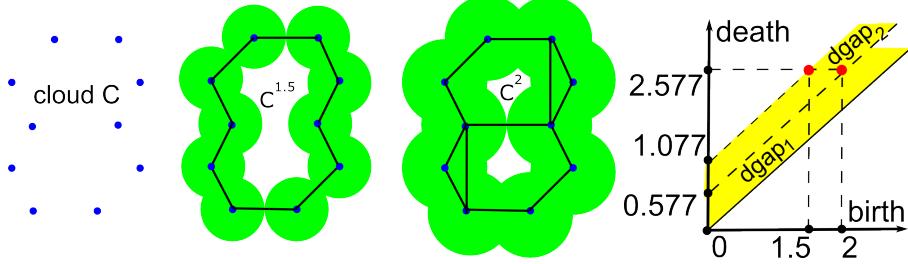


Figure 1: The  $\alpha$ -offsets  $C^{1.5}$ ,  $C^2$  and the 1-dimensional persistence diagram  $\text{PD}\{C^\alpha\}$  for the cloud  $C$  (left).

**Definition 6.3** (bottleneck distance  $d_B$ ). For points  $p = (x_1, y_1)$ ,  $q = (x_2, y_2)$  in  $\mathbb{R}^2$ , we recall the  $L_\infty$ -distance  $\|p - q\|_\infty = \max\{|x_1 - x_2|, |y_1 - y_2|\}$ . The *bottleneck distance* between persistence diagrams  $\text{PD}, \text{PD}'$  is defined by  $d_B = \inf_{\psi} \sup_{q \in \text{PD}} \|q - \psi(q)\|_\infty$  for all bijections  $\psi : \text{PD} \rightarrow \text{PD}'$ .

Stability Theorem 6.4 below informally says that any small perturbation of original data leads to a small perturbation of the persistence diagram. A metric space  $M$  is *totally bounded* if  $M$  has a finite  $\epsilon$ -sample  $C \subset M$  for any  $\epsilon > 0$ .

**Theorem 6.4** (Stability of Persistence). [1, simplified Theorem 5.6] Let  $C$  be an  $\epsilon$ -sample of a graph  $G$  in a totally bounded metric space  $M$ . Then the 1-dimensional persistence diagrams of Čech filtrations on  $G$  and  $C$  are  $\epsilon$ -close, namely  $d_B(\text{PD}\{\check{C}h(G, M; \alpha)\}, \text{PD}\{\check{C}h(C, M; \alpha)\}) \leq \epsilon$ .

The same inequality holds for the filtrations of  $\alpha$ -offsets by Nerve Lemma 2.6, namely  $d_B(\text{PD}\{G^\alpha\}, \text{PD}\{C^\alpha\}) \leq \epsilon$ .

## 6.2. Structure of a Persistence Diagram

**Definition 6.5** (gap  $\text{dgap}_k$ , subdiagram  $\text{DS}_k$ , scale  $\text{ds}_k$ ). For any subspace  $C$  of a metric space, a *diagonal gap* is a strip  $\{0 \leq a < y - x < b\}$  that has dots of  $\text{PD}\{C^\alpha\}$  in both boundary lines  $\{y - x = a\}$  and  $\{y - x = b\}$ , but not inside the strip. For any integer  $k \geq 1$ , the  $k$ -th *widest diagonal gap*  $\text{dgap}_k(C)$  has the  $k$ -th largest vertical width  $|\text{dgap}_k(C)| = b - a$ .

The *diagonal subdiagram*  $\text{DS}_k(C) \subset \text{PD}\{C^\alpha\}$  consists of only the dots above

the lowest of the first  $k$  widest  $\text{dgap}_i(C)$ ,  $i = 1, \dots, k$ . So each  $\text{DS}_k(C)$  is bounded below by  $y - x = a$  and has the *diagonal scale*  $\text{ds}_k(C) = a$ .

In Definition 6.5 if  $\text{PD}\{C^\alpha\}$  has different diagonal gaps with the same width, we say that a lower diagonal gap has a larger width. If  $\text{PD}\{C^\alpha\}$  has dots only in  $m$  different lines  $\{y - x = a_i > 0\}$ ,  $i = 1, \dots, m$ , we have exactly  $m$  diagonal gaps  $\text{dgap}_i(C)$ . We ignore the highest gap  $\{y - x > \max a_i\}$ , so we set  $\text{dgap}_i(C) = \emptyset$  and  $|\text{dgap}_i(C)| = 0$  for  $i > m$ .

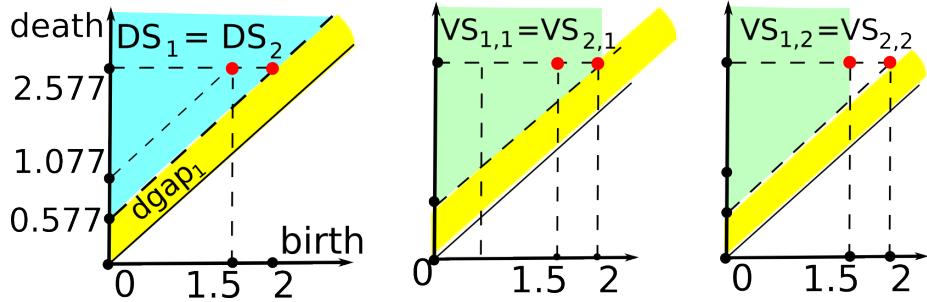


Figure 2: Subdiagrams  $\text{DS}_k(C), \text{VS}_{k,l}(C) \subset \text{PD}\{C^\alpha\}$  from Definitions 6.5 and 6.6 for the cloud  $C$  in Fig. 1.

The cloud  $C$  in Fig. 1 has the persistence diagram  $\text{PD}\{C^\alpha\}$  with only 2 off-diagonal dots  $(1.5, 2.577)$  and  $(2, 2.577)$ . Then  $\text{dgap}_1(C) = \{0 < y - x < 0.577\}$  is below  $\text{dgap}_2(C) = \{0.577 < y - x < 1.577\}$ . So  $\text{DS}_1(C) = \text{DS}_2(C)$  consists of  $(1.5, 2.577), (2, 2.577)$ ,  $\text{ds}_1(C) = \text{ds}_2(C) = 0.577$  in Fig. 2.

Definition 6.5 makes sense in any persistence diagram, say for  $\alpha$ -offsets  $G^\alpha$  of a graph  $G \subset M$ . The graph  $\theta$  in Fig. 3 has the 1-dimensional persistence diagram  $\text{PD}\{\theta^\alpha\}$  containing only one off-diagonal dot  $(0, 2.577)$  of multiplicity 2. Then  $\text{dgap}_1(\theta) = \{0 < y - x < 2.577\}$ ,  $|\text{dgap}_2(\theta)| = 0$  and  $\text{DS}_1(\theta) = \{(0, 2.577), (0, 2.577)\}$  and  $\text{ds}_1(\theta) = 2.577$ .

If we reconstruct cycles of  $G$  from a noisy sample  $C$  using a close diagram  $\text{PD}\{C^\alpha\}$ , we should look for dots (birth, death) having a high persistence death–birth and small birth. Hence we search for a vertical gap that separates dots (birth, death) having a small birth and all other dots.

**Definition 6.6** (gap  $\text{vgap}_{k,l}$ , subdiagram  $\text{VS}_{k,l}$ , scale  $\text{vs}_{k,l}$ ). In the subdiagram  $\text{DS}_k(C)$  from Definition 6.5 a *vertical gap* is a widest vertical strip  $\{a < x < b\}$  whose boundary contains a dot from  $\text{DS}_k(C)$  in the line  $\{x = a\}$ , but not inside the strip. For  $l \geq 1$ , the  $l$ -th *widest vertical gap*  $\text{vgap}_{k,l}(C)$  has the  $l$ -th largest horizontal width  $|\text{vgap}_{k,l}(C)| = b - a$ . The *vertical subdiagram*  $\text{VS}_{k,l}(C) \subset \text{DS}_k(C)$  consists of only the dots to the left of the first  $l$  widest vertical gaps  $\text{vgap}_{k,j}(C)$ ,  $j = 1, \dots, l$ . So each  $\text{VS}_{k,l}(C)$  is bounded on the right by the line  $x = b$  and has the *vertical scale*  $\text{vs}_{k,l}(C) = a$ .

In Definition 6.6 if there are different vertical gaps with the same horizontal width, we say that the leftmost vertical gap has a larger width. So we prefer the leftmost vertical gap, while in Definition 6.5 we prefer the lowest diagonal gap.

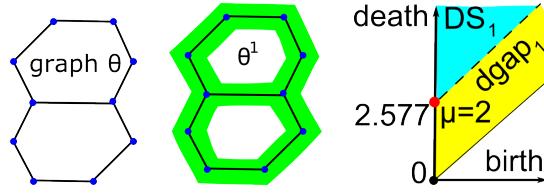


Figure 3: The graph  $\theta \subset \mathbb{R}^2$  with the  $\alpha$ -offset  $\theta^1$  and the 1-dimensional persistence diagram  $\text{PD}\{\theta^\alpha\}$ .

We allow the case  $b = +\infty$ , so the widest vertical gap  $\text{vgap}_{k,1}(C)$  always has the form  $\{x > a\}$ ,  $\text{VS}_{1,1}(C) = \text{DS}_1(C)$  and we set  $|\text{vgap}_{k,1}(C)| = +\infty$ . If  $\text{DS}_k(C)$  has dots in  $m \geq 1$  different lines  $\{x = b_l \geq 0\}$ ,  $l = 1, \dots, m$ , then  $\text{DS}_k(C)$  has exactly  $m$  vertical gaps  $\text{vgap}_{k,l}(C)$ .

For the cloud  $C$  in Fig. 1, the diagonal subdiagram  $\text{DS}_1(C)$  has  $\text{vgap}_{1,1}(C) = \{x > 2\}$ ,  $\text{vgap}_{1,2}(C) = \{1.5 < x < 2\}$ ,  $\text{VS}_{1,1}(C) = \{(1.5, 2.577), (2, 2.577)\}$  and  $\text{VS}_{1,2}(C) = \{(1.5, 2.577)\}$ , so  $\text{vs}_{1,1}(C) = 2$ ,  $\text{vs}_{1,2}(C) = 1.5$ , see Fig. 2. Any finite cloud  $C$  has no cycles at  $\alpha = 0$ , hence  $\text{PD}\{C^\alpha\}$  has no dots in the vertical death axis, so any  $\text{vgap}_{k,l}(C)$  has a left boundary line  $\{x = a > 0\}$  and all  $\text{vs}_{k,l}(C) > 0$ .

Definition 6.6 makes sense for any persistence diagram, say for  $\alpha$ -offsets  $G^\alpha$

of a graph  $G$  in a metric space. The diagonal subdiagram  $\text{DS}_1(\theta)$  for the graph  $\theta$  in Fig. 3 consists of the doubled dot  $(0, 2.577)$ . The only vertical subdiagram  $\text{VS}_{1,1}(\theta)$  is within the vertical death axis and consists of the same doubled dot  $(0, 2.577)$ , so  $\text{vs}_{1,1}(\theta) = 0$ .

### 6.3. HoPeS

**Definition 6.7** (MST). Fix a filtration  $\{Q(C; \alpha)\}$  of complexes on a cloud  $C$ . A *Minimum Spanning Tree*  $\text{MST}(C)$  is a connected graph with the vertex set  $C$  and the minimum total length of edges relative to the filtration  $\{Q(C; \alpha)\}$ . For any scale  $\alpha \geq 0$ , a *forest*  $\text{MST}(C; \alpha)$  is obtained from  $\text{MST}(C)$  by removing all open edges that are longer than  $2\alpha$ .

A graph  $G$  *spans* a possibly disconnected complex  $Q$  on a cloud  $C$  if  $G$  has the vertex set  $C$ , every edge of  $G$  belongs to  $Q$  and the inclusion  $G \subset Q$  induces a 1-1 correspondence between connected components.

**Lemma 6.8.** Fix a filtration  $\{Q(C; \alpha)\}$  of complexes on a cloud  $C$  in a metric space. For any fixed scale  $\alpha \geq 0$ , the forest  $\text{MST}(C; \alpha)$  has the minimum total length of edges among all graphs that span  $Q(C; \alpha)$  at the same scale  $\alpha$ .

*Proof.* Let  $e_1, \dots, e_m \subset \text{MST}(C)$  be all edges longer than  $2\alpha$ , so  $\text{MST}(C) = e_1 \cup \dots \cup e_m \cup \text{MST}(C; \alpha)$ . Assume that there is a graph  $G$  that spans  $Q(C; \alpha)$  and is shorter than  $\text{MST}(C; \alpha)$ . Then the connected graph  $G \cup e_1 \cup \dots \cup e_m$  has the vertex set  $C$  and is shorter than a Minimum Spanning Tree  $\text{MST}(C)$ , which contradicts Definition 6.7.  $\square$

**Definition 6.9** (critical edges). [16, Modified from Definition 4.5] Fix a filtration  $\{Q(C; \alpha)\}$  of complexes on a cloud  $C$ . We call an edge *critical* if the addition of the edge to the filtration  $\{Q(C; \alpha)\}$  creates a new homology class that does not immediately die.

We define the *birth* of a critical edge to be the value of the scale  $\alpha$  when the edge first enters the filtration.

Defining the *death* of a critical edge is less simple. Define  $\tilde{\mathcal{K}}_\alpha = \{K_1, \dots, K_s\}$  to be the set of critical edges with  $\text{birth}(K_i) \leq \alpha$  that have not yet been assigned a death value. Then  $[K_1], \dots, [K_s]$  form a basis of  $H_1((\text{MST}(C; \alpha) \cup \tilde{\mathcal{K}}_\alpha)/\text{MST}(C; \alpha))$ . Denote  $f : H_1((\text{MST}(C; \alpha) \cup \tilde{\mathcal{K}}_\alpha)/\text{MST}(C; \alpha)) \rightarrow H_1(Q(C; \alpha)/\text{MST}(C; \alpha))$  to be the map on homology induced by the inclusion of  $(\text{MST}(C; \alpha) \cup \tilde{\mathcal{K}}_\alpha) / \text{MST}(C; \alpha)$  into  $Q(C; \alpha) / \text{MST}(C; \alpha)$ . Let  $\{b_1, \dots, b_r\}$  to be a basis of  $\ker(f)$ , where  $r = \dim \ker(f)$  is equal to the number of critical edges that die exactly at scale  $\alpha$ , so we have  $r \leq s$ . We can expand each basis element as  $b_i = \sum_{j=1}^s c_{ij}[K_j]$ , with  $c_{ij} \in \mathbb{F}$ . We consider (in  $\mathbb{F}$ ) the system of equations  $\sum_{j=1}^s c_{ij}x_j = 0$  for  $i \in \{1, \dots, r\}$ . Since basis elements are linearly independent, so are these equations. This implies that this system of equations can be solved with  $r$  leading variables, each of which is expressed in terms of the remaining  $s - r$  free variables. Let  $I \subseteq \{1, \dots, s\}$  be the set of all indices of leading variables ( $I$  could be empty if  $f$  is trivial). For each  $i \in I$ , we declare the death value of  $K_i$  to be  $\alpha$ .

**Remark 6.10.** It is possible that there is a choice of leading variables. In this situation, we are required to employ the elder rule. Namely, the set of leading variables selected should be the one whose corresponding set of critical edges has the greatest combined birth value. If there are distinct sets of leading variables with the greatest combined birth value, then we have a genuine choice as to which set to choose, in which case the optimality theorem will be valid regardless of the choice.

**Definition 6.11** (HoPeS). A *Homologically Persistent Skeleton* HoPeS( $C$ ) is the union of a minimum spanning tree  $\text{MST}(C)$  and all critical edges with their labels (birth, death).

For any scale  $\alpha \geq 0$ , the *reduced* skeleton HoPeS( $C; \alpha$ ) is obtained from HoPeS( $C$ ) by removing all edges that are longer than  $2\alpha$  and all critical edges  $e$  with  $\text{death}(e) \leq \alpha$ .

## 7. Optimality Guarantees for HoPeS

Theorem 7.8 will follow from Lemma 7.1 and Propositions 7.5 and 7.7, which require Lemmas 7.2, 7.3, 7.4 below. Recall that we fix a filtration  $\{Q(C; \alpha)\}$  of complexes on a cloud  $C$ , but use the simpler notation  $\text{HoPeS}(C)$  for the skeleton.

**Lemma 7.1.** Let a class  $\gamma \in H_1(Q(C; \alpha))$  be born due to a critical edge  $e$  added to  $Q(\alpha)$ . Then  $2 \cdot \text{birth}(\gamma)$  equals the length  $|e|$  relative to the filtration  $\{Q(C; \alpha)\}$ .

*Proof.* By Definition 6.9 the critical edge  $e(\gamma)$  is the last edge added to a cycle  $L \subset Q(C; \alpha)$  giving birth to the homology class  $\gamma$  at  $\alpha = \text{birth}(\gamma)$ . The length  $|e|$  equals the doubled scale  $2\alpha$  when  $e$  enters  $Q(C; \alpha)$ , so  $|e(\gamma)| = 2 \cdot \text{birth}(\gamma)$ .  $\square$

**Lemma 7.2.** For any scale  $\alpha > 0$ , the reduced skeleton  $\text{HoPeS}(C; \alpha)$  is a subgraph of the complex  $Q(C; \alpha)$ .

*Proof.* By Definition 6.11, for any  $\alpha > 0$ , the skeleton  $\text{HoPeS}(C; \alpha)$  consists of the forest  $\text{MST}(C; \alpha)$  and all critical edges with lengths  $\text{death}(e) \leq |e| \leq 2\alpha$ . Lemma 6.8 implies that  $\text{MST}(C; \alpha) \subset Q(C; \alpha)$ . Any critical edge  $e \subset \text{HoPeS}(C)$  belongs to the complex  $Q(C; \alpha)$  for  $2\alpha \geq |e|$ . Hence  $\text{HoPeS}(C; \alpha) \subset Q(C; \alpha)$ .  $\square$

The inclusion  $\text{HoPeS}(C; \alpha) \subset Q(C; \alpha)$  in Lemma 7.2 induces a homomorphism  $f_*$  in  $H_1$ . Lemmas 7.3 and 7.4 analyze what happens with  $f_*$  when a critical edge  $e$  is added to (or deleted from)  $Q = Q(C; \alpha)$  and  $G = \text{HoPeS}(C; \alpha)$ .

**Lemma 7.3** (addition of a critical edge). Let an inclusion  $f : G \rightarrow Q$  of a graph  $G$  into a simplicial complex  $Q$  induce an isomorphism  $f_* : H_1(G) \rightarrow H_1(Q)$ . Between some vertices  $u, v \in G$  let us add an edge  $e$  to both  $G$  and  $Q$  that creates a new homology class  $\gamma \in H_1(Q \cup e)$ . Then  $f_*$  extends to an isomorphism  $H_1(G \cup e) \rightarrow H_1(Q \cup e)$ .

*Proof.* Let  $L \subset G \cup e$  be a cycle containing the added edge  $e$ . Then  $f$  extends to the inclusion  $G \cup e \rightarrow Q \cup e$  and induces the isomorphism  $H_1(G \cup e) \cong$

$H_1(G) \oplus \langle [L] \rangle$ . Now  $f(L)$  is a cycle in  $Q \cup e$  and  $H_1(Q \cup e) \cong H_1(Q) \oplus \langle [f(L)] \rangle$ . Mapping  $[L]$  to  $[f(L)] \in H_1(Q \cup e)$ , we extend  $f_*$  to a required isomorphism  $H_1(G) \oplus \langle [L] \rangle \rightarrow H_1(Q) \oplus \langle [f(L)] \rangle$ .

□

**Lemma 7.4** (deletion of a critical edge). Let an inclusion  $f : G \rightarrow Q$  of a graph  $G$  into a simplicial complex  $Q$  induce an isomorphism  $f_* : H_1(G) \rightarrow H_1(Q)$ . Let a homology class  $\gamma \in H_1(Q)$  die after adding a triangle  $T$  to the complex  $Q$ . Let a  $e$  be a longest open edge of the triangle  $T$ . Then  $f_*$  descends to an isomorphism  $H_1(G - e) \rightarrow H_1(Q \cup T)$ .

*Proof.* Adding the triangle  $T$  to the complex  $Q$  kills the homology class  $\partial T$  of the boundary  $\partial T \subset G$ . Then  $H_1(Q \cup T) \cong H_1(Q)/\langle [\partial T] \rangle$ . Deleting the open edge  $e$  from the boundary  $\partial T \subset G$  makes the homology group smaller:  $H_1(G - e) \cong H_1(G)/\langle [\partial T] \rangle$ . The isomorphism  $f_*$  descends to the isomorphism  $H_1(G)/\langle [\partial T] \rangle \rightarrow H_1(Q)/\langle [\partial T] \rangle$ .

□

**Proposition 7.5.** The inclusion  $\text{HoPeS}(C; \alpha) \rightarrow Q(C; \alpha)$  from Lemma 7.2 induces an isomorphism of 1-dimensional homology groups:  $H_1(\text{HoPeS}(C; \alpha)) \rightarrow H_1(Q(C; \alpha))$ .

*Proof.* At the scale  $\alpha = 0$ , the reduced skeleton  $\text{HoPeS}(C; 0)$  and the complex  $Q(C; 0)$  coincide with the cloud  $C$ , so their 1-dimensional homology groups are trivial. Each time when a homology class is born or dies in  $H_1(Q(C; \alpha))$ , the isomorphism in  $H_1$  induced by the inclusion  $\text{HoPeS}(C; \alpha) \rightarrow Q(C; \alpha)$  is preserved by Lemmas 7.3 and 7.4.

□

**Lemma 7.6.** Let a cycle  $L \subset Q(C; \alpha)$  represent a homology class  $\gamma \in H_1(Q(C; \alpha))$  in the diagram  $\text{PD}\{Q(C; \alpha)\}$ . Then any longest edge  $e \subset L$  has the length  $|e| \geq 2 \cdot \text{birth}(\gamma)$ .

*Proof.* Let a longest edge  $e$  of a cycle  $L \subset Q(C; \alpha)$  representing the class  $\gamma$  have a half-length  $\alpha < \text{birth}(\gamma)$ . Then  $L$  enters the complex  $Q(C; \alpha)$  earlier than  $\text{birth}(\gamma)$  and can not represent the class  $\gamma$  that starts living from  $\alpha = \text{birth}(\gamma)$ .

□

Recall that the forest  $\text{MST}(C; \alpha)$  on cloud  $C$  at a scale  $\alpha$  is obtained from a minimum spanning tree  $\text{MST}(C)$  by removing all open edges that are longer than  $2\alpha$ .

An edge  $e$  is *splitting* a graph  $G$  if removing the open edge  $e$  makes  $G$  disconnected. Otherwise the edge  $e$  will be called *non-splitting* and should be in a cycle of the graph  $G$ .

**Proposition 7.7.** Let a graph  $G \subset Q(C; \alpha)$  span  $Q(C; \alpha)$  and  $H_1(G) \rightarrow H_1(Q(C; \alpha))$  be an isomorphism induced by the inclusion. Let  $(b_i, d_i)$ ,  $i = 1, \dots, m$ , be all dots in the persistence diagram  $\text{PD}\{Q(C; \alpha)\}$  such that  $\text{birth} \leq \alpha < \text{death}$ . Then the total length of  $G$  is bounded below by the total length of edges of the forest  $\text{MST}(C; \alpha)$  plus  $2 \sum_{i=1}^m b_i$ .

*Proof.* Let the subgraph  $G_1 \subset G$  consist of all non-splitting edges of  $G$  and  $e_1 \subset G_1$  be a longest open edge. Removing  $e_1$  from  $G$  makes  $H_1(G)$  smaller. Hence there is a cycle  $L_1 \subset G_1$  containing  $e_1$  and representing a class  $\gamma_1 \in H_1(Q(C; \alpha))$  that corresponds to some off-diagonal dot in  $\text{PD}\{Q(C; \alpha)\}$ , say  $(b_1, d_1)$ . Then  $\gamma_1$  lives over  $b_1 = \text{birth}(\gamma_1) \leq \alpha < d_1 = \text{death}(\gamma_1)$ . Lemma 7.6 implies that  $|e_1| \geq 2b_1$ .

Let the graph  $G_2 \subset G - e_1$  consist of all non-splitting edges and  $e_2 \subset G_2$  be a longest open edge. Similarly, find the corresponding point  $(b_2, d_2)$ , conclude that  $|e_2| \geq 2b_2$  and so on until we get  $\sum_{i=1}^m |e_i| \geq 2 \sum_{i=1}^m b_i$ . After removing all open edges  $e_1, \dots, e_m$ , the remaining graph  $G - (e_1 \cup \dots \cup e_m)$  still spans the (possibly disconnected) complex  $Q(\alpha)$ . Indeed, each time we removed a non-splitting edge. So the total length of  $G - (e_1 \cup \dots \cup e_m)$  is not smaller than the total length of  $\text{MST}(C; \alpha)$  by Lemma 6.8.

□

**Theorem 7.8** (optimality of reduced skeletons at all scales). Fix a filtration  $\{Q(C; \alpha)\}$  of complexes on a cloud  $C$  in a metric space. For any fixed scale  $\alpha > 0$ , the reduced skeleton  $\text{HoPeS}(C; \alpha)$  has the minimum total length of edges over all graphs  $G \subset Q(C; \alpha)$  that span  $Q(C; \alpha)$  and induce an isomorphism in homology  $H_1(G) \rightarrow H_1(Q(C; \alpha))$ .

*Proof.* For any  $\alpha > 0$ , the inclusion  $\text{HoPeS}(C; \alpha) \rightarrow Q(C; \alpha)$  induces an isomorphism in  $H_1$  by Proposition 7.5. Let classes  $\gamma_1, \dots, \gamma_m$  correspond to all  $m$  dots counted with multiplicities in the ‘rectangle’  $\{\text{birth} \leq \alpha < \text{death}\} \subset \text{PD}\{Q(C; \alpha)\}$ . Then  $\gamma_1, \dots, \gamma_m$  form a basis of  $H_1(Q(C; \alpha)) \cong H_1(\text{HoPeS}(C; \alpha))$  by Definition 6.2.

The total length of  $\text{HoPeS}(C; \alpha)$  equals the total length of  $\text{MST}(C; \alpha)$  plus  $2 \sum_{i=1}^m \text{birth}(\gamma_i)$  by Lemma 7.1. By Proposition 7.7 this length is minimal over all graphs  $G \subset Q(C; \alpha)$  that span  $Q(C; \alpha)$  and have the same  $H_1$  as  $Q(C; \alpha)$ . □

## 8. Reconstruction Guarantees for HoPeS

**Definition 8.1** (derived skeletons  $\text{HoPeS}_{k,l}$ ). Let  $C$  be a finite cloud in a metric space. For integers  $k, l \geq 1$ , the *derived skeleton*  $\text{HoPeS}_{k,l}(C)$  is obtained from a full  $\text{HoPeS}(C)$  by removing all edges that are longer than  $2\text{vs}_{k,l}(C)$  and keeping only critical edges with  $(\text{birth}, \text{death}) \in \text{VS}_{k,l}(C)$  and with  $\text{death} > \text{vs}_{k,l}(C)$ .

**Definition 8.2** (radius  $\rho$  of a cycle and thickness  $\theta(G)$  of a graph). For a graph  $G \subset \mathbb{R}^2$ , the *radius*  $\rho$  of a non-self-intersecting cycle  $L \subset G^\alpha$  is the persistence of its corresponding homology class in the filtration. In general, when  $\alpha$  is increasing, new holes can be born in  $G^\alpha$ . We define the *thickness* of  $G$ ,  $\theta(G)$ , to be the maximum persistence of any hole born after  $\alpha = 0$ .

**Lemma 8.3.** Let  $G$  be a graph in a metric space  $M$  with  $\text{rank}(H_1(G)) = m$ . Then the 1-dimensional persistence diagram  $\text{PD}(G^\alpha)$  of  $\alpha$ -offsets of  $G$  has  $m$  dots on the death axis. (For simplicity, a single dot with multiplicity  $k > 1$  is counted as  $k$  dots.)

*Proof.* At  $\alpha = 0$ ,  $G^\alpha = G$ , and so  $\text{rank}(H_1(G^\alpha)) = \text{rank}(H_1(G)) = m$ . So there are no more than  $m$  dots on the death axis. All 1-dimensional cycles die at some finite scale  $\alpha$ , and so there are no fewer than  $m$  dots on the death axis. Hence the number of dots on the death axis is  $m$ .

□

**Theorem 8.4.** Let  $M$  be a metric space, and let  $C$  be any  $\epsilon$ -sample of a connected graph  $G \subset M$  with  $\text{rank}(H_1(G)) = m$ . Then by Lemma 8.3, the persistence diagram  $\text{PD}(G^\alpha)$  has  $m$  dots on the death axis. Let these  $m$  dots have ordered  $y$ -values  $y_1 \leq \dots \leq y_m$ . If  $y_1 > 7\epsilon + 2\theta(G) + \max_{i=1,\dots,m-1} \{y_{i+1} - y_i\}$ , then we get the lower bound for noise  $\text{vs}_{1,1}(C) \leq \epsilon$  and the derived skeleton  $\text{HoPeS}_{1,1}(C) \subset G^{2\epsilon}$  has the same  $H_1$  as  $G$ .

*Proof.* Apart from the dots on the  $y$ -axis, all other dots in the 1D persistence diagram  $\text{PD}\{G^\alpha\}$  come from smaller holes in the  $\alpha$ -offsets  $G^\alpha$  that were born later. The maximum persistence of these holes is bounded above by the thickness  $\theta(G)$ .

The given inequality  $y_1 > 7\epsilon + 2\theta(G) + \max_{i=1,\dots,m-1} \{y_{i+1} - y_i\}$  guarantees that the gap  $\{\theta(G) < y - x < y_1\}$  in  $\text{PD}\{G^\alpha\}$  is wider than any other gaps including the higher gaps  $y_{i+1} - y_i$  between the dots  $(0, y_i) \in \text{PD}\{G^\alpha\}$ ,  $i = 1, \dots, m-1$ , since the inequality implies  $y_1 - \theta(G) > \max_{i=1,\dots,m-1} \{y_{i+1} - y_i\}$  and  $y_1 - \theta(G) > \theta(G)$ .

By Stability Theorem 6.4, the perturbed diagram  $\text{PD}\{C^\alpha\}$  is in the  $\epsilon$ -offset of  $\text{PD}\{G^\alpha\} \subset \cup_{i=1}^m (0, y_i) \cup \{y - x < \theta(G)\}$  with respect to the  $L_\infty$  metric on  $\mathbb{R}^2$ . All noisy dots near the diagonal in  $\text{PD}\{C^\alpha\}$  can not be higher than  $\theta(G) + 2\epsilon$  after projecting along the diagonal  $\{x = y\}$  to the vertical axis  $\{x = 0\}$ . The remaining dots can not be lower than  $y_1 - 2\epsilon$  after the same projection  $(x, y) \rightarrow y - x$ . Hence the smaller diagonal strip  $\{\theta(G) + 2\epsilon < y - x < y_1 - 2\epsilon\}$  of vertical width  $y_1 - 4\epsilon - \theta(G)$  is still empty in the perturbed diagram  $\text{PD}\{C^\alpha\}$ .

By Stability Theorem 6.4, any dot  $(0, y_i) \in \text{PD}\{G^\alpha\}$ ,  $i \geq 1$ , can not jump lower than the line  $y - x = y_i - 2\epsilon$  or higher than  $y - x = y_i + \epsilon$  (not  $y - x = y_i + 2\epsilon$  because the dot is on the  $y$ -axis and hence can not move in the negative  $x$

direction). Then the widest diagonal gap between these perturbed dots has a vertical width at most  $\max_{i=1,\dots,m-1} \{y_{i+1} - y_i\} + 3\epsilon$ .

Since all dots near the diagonal have diagonal gaps not wider than  $\theta(G) + 2\epsilon$ , by applying the given inequality, all other diagonal gaps have a vertical width smaller than  $y_1 - 4\epsilon - \theta(G)$ . Hence the first widest gap in  $\text{PD}\{C^\alpha\}$  covers the diagonal strip  $\{\theta(G) + 2\epsilon < y - x < y_1 - 2\epsilon\}$ , which is within the first widest gap  $\{\theta(G) < y - x < y_1\}$  in the original diagram  $\text{PD}\{G^\alpha\}$ .

Then the subdiagram  $\text{DS}_1(C^\alpha)$  above the line  $y - x = y_1 - 2\epsilon$  contains exactly  $m$  perturbations of the original dots  $(0, y_i)$  in the vertical strip  $\{0 \leq x \leq \epsilon\}$ . Hence the reduced skeleton  $\text{HoPeS}_{1,1}(C)$  contains exactly  $m$  critical edges corresponding to all  $m$  dots of the subdiagram  $\text{DS}_1(C^\alpha)$ .

It remains to prove that  $\text{HoPeS}_{1,1}(C)$  is  $2\epsilon$ -close to  $G$ . Let the critical scale  $\alpha(C)$  be the maximum birth over all dots in  $\text{DS}_1(C^\alpha)$ . These dots are at most  $\epsilon$  away from their corresponding points  $(0, y_i)$  in the vertical axis, so critical scale  $\alpha(C)$  is at most  $\epsilon$ . A longest edge  $e$  of any cycle in  $\text{HoPeS}_{1,1}(C)$  persisting over birth  $\leq \alpha <$  death has the half-length equal to its birth, because adding  $e$  gave birth to the homology class. Then all edges in  $\text{HoPeS}_{1,1}(C)$  have half-lengths at most  $\alpha(C) \leq \epsilon$ . Hence  $\text{HoPeS}_{1,1}(C)$  is covered by the disks with the radius  $\epsilon$  and centres at all points of  $C$ , so  $\text{HoPeS}_{1,1}(C) \subset C^\epsilon \subset G^{2\epsilon}$ .

□

Reconstruction Theorem 8.9 will follow from Lemma 8.7, Propositions 8.5, 8.6, 8.8 and will imply Corollary 8.10.

**Proposition 8.5.** Let  $C$  be any  $\epsilon$ -sample of a graph  $G$ . If  $|\text{dgap}_k(G)| - |\text{dgap}_{k+1}(G)| > 8\epsilon$ , there is a bijection  $\psi : \text{DS}_k(G) \rightarrow \text{DS}_k(C)$  so that  $\|q - \psi(q)\|_\infty \leq \epsilon$ ,  $q \in \text{DS}_k(G)$ .

*Proof.* By Stability Theorem 6.4 there is a bijection  $\psi : \text{PD}\{G^\alpha\} \rightarrow \text{PD}\{C^\alpha\}$  such that  $q, \psi(q)$  are  $\epsilon$ -close in the  $L_\infty$  distance on  $\mathbb{R}^2$  for all  $q \in \text{PD}\{G^\alpha\}$ . The  $\epsilon$ -neighbourhood of a dot  $q = (x, y)$  in the  $L_\infty$  distance is the square  $[x - \epsilon, x + \epsilon] \times [y - \epsilon, y + \epsilon]$ . Under the diagonal projection  $\text{pr}$  to the vertical death axis, this square maps to the interval  $[y - x - 2\epsilon, y - x + 2\epsilon]$ . Hence

any diagonal gap  $\{a < y - x < b\}$  in  $\text{PD}\{G^\alpha\}$  can become thinner or wider in  $\text{PD}\{C^\alpha\}$  by at most  $4\epsilon$  due to dots  $q$  ‘jumping’ to  $\psi(q)$  by at most  $2\epsilon$  relative to the projection  $\text{pr}(x, y) = y - x$ .

By the given inequality the first  $k$  widest gaps  $\text{dgap}_i(G)$  in  $\text{PD}\{G^\alpha\}$  for  $i = 1, \dots, k$  are wider by at least  $8\epsilon$  than all other  $\text{dgap}_i(G)$  for  $i > k$ . Hence all dots between any two successive gaps from the first  $k$  widest can not ‘jump’ over these wide gaps and remain ‘trapped’ between corresponding diagonal gaps in the perturbed diagram  $\text{PD}\{C^\alpha\}$ .

Despite the order of the first  $k$  widest gaps  $\text{dgap}_i(G)$ ,  $i = 1, \dots, k$ , may not be preserved under  $\psi$ , the lowest  $\{a < y - x < b\}$  of these  $k$  diagonal gaps is respected by  $\psi$  as follows. The thinner strip  $S = \{a + 2\epsilon < y - x < b - 2\epsilon\}$  has no dots from  $\text{PD}\{C^\alpha\}$  and has the vertical width  $|S| \geq |\text{dgap}_k(G)| - 4\epsilon > |\text{dgap}_{k+1}(G)| + 4\epsilon \geq |\text{dgap}_{k+1}(C)|$ .

Then the diagonal strip  $S = \{a + 2\epsilon < y - x < b - 2\epsilon\}$  is within the lowest gap of the first  $k$  widest gaps  $\text{dgap}_i(C)$ ,  $i = 1, \dots, k$ . Hence all dots above  $S$  remain above  $S$  under the bijection  $\psi$ . By Definition 6.5 all these dots above  $S$  in  $\text{PD}\{G^\alpha\}$  and in  $\text{PD}\{C^\alpha\}$  form the diagonal subdiagrams  $\text{DS}_k(G)$  and  $\text{DS}_k(C)$ , respectively. So  $\psi$  descends to a bijection  $\text{DS}_k(G) \rightarrow \text{DS}_k(C)$  between finite subdiagrams.

□

**Proposition 8.6.** Let  $\psi : \text{DS}_k(G) \rightarrow \text{DS}_k(C)$  be a bijection such that  $\|q - \psi(q)\|_\infty \leq \epsilon$  for all dots  $q \in \text{DS}_k(G)$  as in Proposition 8.5. If  $|\text{vgap}_{k,l}(G)| - |\text{vgap}_{k,l+1}(G)| > 4\epsilon$  for some  $l \geq 1$ , then  $\psi$  restricts to a bijection  $\text{VS}_{k,l}(G) \rightarrow \text{VS}_{k,l}(C)$  between smaller vertical subdiagrams.

*Proof.* The  $x$ -coordinate of any dot  $q \in \text{DS}_k(G)$  changes under the given bijection  $\psi$  by at most  $\epsilon$ . Similarly to the proof of Proposition 8.5 each vertical gap  $\text{vgap}_{k,l}(G)$  becomes thinner or wider by at most  $2\epsilon$ .

By the given inequality the first  $l$  gaps  $\text{vgap}_{k,j}(G)$  in  $\text{PD}_k\{G^\alpha\}$  for  $j = 1, \dots, l$  are wider by at least  $4\epsilon$  than all other  $\text{vgap}_{k,j}(G)$  for  $j > l$ . Hence all dots between any two successive gaps from the first  $k$  widest can not ‘jump’

over these wide gaps and remain ‘trapped’ between corresponding vertical gaps in the perturbed diagram  $\text{PD}\{C^\alpha\}$ .

Despite the order of the first  $l$  widest  $\text{vgap}_{k,j}(G)$ ,  $j = 1, \dots, l$ , may not be preserved under  $\psi$ , the leftmost  $\{a < x < b\}$  of these  $l$  vertical gaps is respected by  $\psi$  in the following sense. The thinner strip  $S = \{a + \epsilon < x < b - \epsilon\}$  contains no dots from  $\text{DS}_k(C)$  and has the horizontal width  $|S| \geq |\text{vgap}_{k,l}(G)| - 2\epsilon > |\text{dgap}_{k,l+1}(G)| + 2\epsilon \geq |\text{dgap}_{k,l+1}(C)|$ .

Then the vertical strip  $S = \{a + \epsilon < x < b - \epsilon\}$  is within the leftmost of the first  $l$  widest  $\text{vgap}_{k,j}(C)$ ,  $j = 1, \dots, l$ . Hence all dots to the left of  $S$  remain to the left of  $S$  under the bijection  $\psi$ . By Definition 6.6 all these dots to the left of  $S$  in  $\text{PD}\{G^\alpha\}$  and in  $\text{PD}\{C^\alpha\}$  form the vertical subdiagrams  $\text{VS}_{k,l}(G)$  and  $\text{VS}_{k,l}(C)$ , respectively. So  $\psi$  descends to a bijection  $\text{VS}_{k,l}(G) \rightarrow \text{VS}_{k,l}(C)$  between finite sets.

□

**Lemma 8.7.** The derived skeleton  $\text{HoPeS}_{k,l}(C)$  is a subgraph of the reduced skeleton  $\text{HoPeS}(C; \text{vs}_{k,l}(C))$ .

*Proof.* By Definition 6.11 all edges of the reduced skeleton  $\text{HoPeS}(C; \text{vs}_{k,l}(C))$  have a half-length at most  $\text{vs}_{k,l}(C)$  and  $\text{death} > \text{vs}_{k,l}(C)$  for all critical edges. Definition 8.1 also imposes the extra restriction on critical edges of  $\text{HoPeS}_{k,l}(C)$ , namely each dot (birth, death) is in the vertical subdiagram  $\text{VS}_{k,l}(C)$ . So  $\text{HoPeS}_{k,l}(C) \subset \text{HoPeS}(C; \text{vs}_{k,l}(C))$ .

□

**Proposition 8.8** (approximation by reduced HoPeS). Let  $C$  be any finite  $\epsilon$ -sample of a subspace  $G$  in a metric space  $M$ . Then the reduced skeleton  $\text{HoPeS}(C; \alpha)$  for any scale  $\alpha > 0$  is contained within the  $(\epsilon + \alpha)$ -offset  $G^{\epsilon+\alpha} \subset M$ .

*Proof.* Since the cloud  $C$  is an  $\epsilon$ -sample of  $G$ , we get  $C \subset G^\epsilon$ . Every edge of  $\text{HoPeS}(C; \alpha)$  has a half-length at most  $\alpha$  by Definition 6.11. The edge between any points  $p, q \in C$  is covered by the balls with the radius  $\alpha$  and the centres  $p, q$ . Hence  $\text{HoPeS}(C; \alpha) \subset C^\alpha \subset G^{\epsilon+\alpha}$ .

□

**Theorem 8.9** (reconstruction by derived skeletons). Let  $C$  be any  $\epsilon$ -sample of an unknown graph  $G$  in a metric space, so  $C \subset G^\epsilon$  and  $G \subset C^\epsilon$ . Let  $G$  satisfy the following conditions.

- (1) All cycles  $L \subset G$  are ‘persistent’, namely  $\text{death}(L) \geq \text{ds}_k(G)$  for some index  $k \geq 1$ .
- (2) The width  $|\text{dgap}_k(G)|$  ‘jumps’, namely  $|\text{dgap}_k(G)| - |\text{dgap}_{k+1}(G)| > 8\epsilon$  for the same  $k$  as in (1).
- (3) No cycles are born in  $\alpha$ -offsets  $G^\alpha$  for ‘small’  $\alpha > 0$ , namely  $\text{vs}_{k,l}(G) = 0$  for some  $l \geq 1$ .
- (4) The width  $|\text{vgap}_{k,l}(G)|$  ‘jumps’, so  $|\text{vgap}_{k,l}(G)| - |\text{vgap}_{k,l+1}(G)| > 4\epsilon$  for the same  $k, l$  as above.

Then we get the lower bound for noise  $\text{vs}_{k,l}(C) \leq \epsilon$  and the derived skeleton  $\text{HoPeS}_{k,l}(C) \subset G^{2\epsilon}$  has the same  $H_1$  as  $G$ .

*Proof.* Proposition 8.5 due to condition (2) implies that there is a bijection  $\psi : \text{DS}_k(G) \rightarrow \text{DS}_k(C)$  so that  $\|q - \psi(q)\|_\infty \leq \epsilon$  for all  $q \in \text{DS}_k(G)$ . Proposition 8.6 due to condition (4) implies that  $\psi$  descends to a bijection  $\text{VS}_{k,l}(G) \rightarrow \text{VS}_{k,l}(C)$  between vertical subdiagrams.

In general, all cycles in a graph  $G$  give birth to corresponding homology classes in  $H_1(G^\alpha)$  at the scale  $\alpha = 0$ . These classes may split later at  $\alpha > 0$ , but will eventually die and always give dots  $(0, \text{death}) \in \text{PD}\{G^\alpha\}$  in the vertical death axis. For any cycle  $L \subset G$ , let  $\text{death}(L)$  be the maximum  $\alpha$  such that  $H_1(G^\alpha)$  contains the class  $[L]$ . The graph  $\theta$  in Fig. 3 has 3 cycles with the same  $\text{death}(L) = 2.577$ .

Let  $L_1, \dots, L_m \subset G$  be all  $m$  cycles generating  $H_1(G)$ . Then the 1-dimensional persistence diagram  $\text{PD}\{G^\alpha\}$  contains  $m$  dots  $(0, \text{death}(L_i))$ ,  $i = 1, \dots, m$ , because each class  $[L_i]$  persists over  $0 \leq \alpha < \text{death}(L_i)$  by Definition 6.2. Condition (1) implies that all dots  $(0, \text{death}(L_i))$  belong to the diagonal subdiagram  $\text{DS}_k(G)$ , hence to  $\text{VS}_{k,l}(C)$ .

Condition (3)  $\text{vs}_{k,l}(G) = 0$  means that the leftmost of the first  $l$  widest  $\text{vgap}_{k,j}(G)$ ,  $j = 1, \dots, l$ , is attached to the vertical death axis in  $\text{PD}\{G^\alpha\}$ , which should contain the vertical subdiagram  $\text{VS}_{k,l}(G)$ . So  $\text{VS}_{k,l}(G)$  consists of the  $m$  dots  $(0, \text{death}(L_i))$ ,  $i = 1, \dots, m$ . Then the vertical subdiagram  $\text{VS}_{k,l}(C)$  for the cloud  $C$  also has exactly  $m$  dots, which are ‘noisy’ images  $\psi(0, \text{death}(L_i))$ ,  $i = 1, \dots, m$ . Moreover, all these dots of  $\text{VS}_{k,l}(C)$  are at most  $\epsilon$  away from the vertical axis, so the vertical scale  $\text{vs}_{k,l}(C)$  is at most  $\epsilon$ .

The lowest of the points  $\psi(0, \text{death}(L_i))$  has a death  $\geq \min_{i=1, \dots, m} \text{death}(L_i) - \epsilon \geq \text{DS}_k(G) - \epsilon > |\text{dgap}_k(G)| - \epsilon > 7\epsilon > \text{vs}_{k,l}(C)$ . So the condition  $\text{death} > \text{vs}_{k,l}(C)$  from Definition 8.1 is satisfied. Hence the reduced skeleton  $\text{HoPeS}_{k,l}(C)$  contains exactly  $m$  critical edges corresponding to all  $m$  dots of  $\text{VS}_{k,l}(C)$ . All these  $m$  critical edges of  $\text{HoPeS}_{k,l}(C)$  generate  $H_1$  of the required rank  $m$ . The geometric approximation  $\text{HoPeS}_{k,l}(C) \subset G^{2\epsilon}$  follows from Proposition 8.8 and Lemma 8.7 for the vertical scale  $\alpha = \text{vs}_{k,l}(C) \leq \epsilon$ .

□

**Corollary 8.10.** In the conditions of Theorem 8.9 if another cloud  $\tilde{C}$  is  $\delta$ -close to  $C$ , then the perturbed derived skeleton  $\text{HoPeS}_{k,l}(\tilde{C})$  is  $(2\delta + 4\epsilon)$ -close to  $\text{HoPeS}_{k,l}(C)$ .

*Proof.* The condition that the perturbed cloud  $\tilde{C}$  is  $\delta$ -close to the original cloud  $C$ , which is  $\epsilon$ -closed to the graph  $G$ , implies that  $\tilde{C}$  is  $(\delta + \epsilon)$ -close to  $G$ .

Reconstruction Theorem 8.9 for the  $\epsilon$ -sample  $C$  and  $(\delta + \epsilon)$ -sample  $\tilde{C}$  of  $G$  says that  $\text{HoPeS}_{k,l}(C)$  is  $2\epsilon$ -close to  $G$  and  $\text{HoPeS}_{k,l}(\tilde{C})$  is  $(2\delta + 2\epsilon)$ -close to  $G$ . Hence  $\text{HoPeS}_{k,l}(C)$  and  $\text{HoPeS}_{k,l}(\tilde{C})$  are  $(2\delta + 4\epsilon)$ -close as required.

□

## 9. Dataset of Random Point Clouds around Planar Graphs

In order to compare the abilities of the skeletonisation algorithms, we require a large dataset of point clouds upon which we can apply the algorithms and perform analysis on the results.

To produce a point cloud for our dataset, we start with a connected graph, which we call a pattern. Then, a point is selected uniformly from the pattern, and is perturbed according to a given type and magnitude of noise, before being added to the point cloud. This is repeated until the point cloud reaches the required size. Patterns, noise and the size of point clouds are explained in more detail below.

### 9.1. Patterns

We used 3 different types of patterns – wheel, grid and concentric squares:

- *The wheel pattern* of size  $k$  (or wheel  $k$  pattern) consists of a central vertex in addition to  $k$  circumference vertices equally distributed along the circumference of an imaginary circle of unit radius centred at the central vertex. There exists edges between the central vertex and all circumference vertices, and between adjacent circumference vertices. See figure 4 for an example.
- *The grid pattern* of size  $k \times l$  (or grid  $k, l$  pattern) consists of a  $k$ -by- $l$  grid of unit squares. See figure 4 for an example.
- *The concentric squares pattern* of size  $k, l$  (or squares  $k, l$  pattern) consists of  $k$  concentric squares such that the centres of all the squares coincide. The innermost square is the unit square, and the diagonal distance between two corresponding vertices of any two adjacent squares is of unit length. Furthermore, in order that these graphs are connected, we add diagonal connections between any two adjacent squares. Between the two outermost squares, we add  $l$  diagonal edges, whilst between any other pair of adjacent squares there is a complete set of 4 diagonal edges. See figure 4 for an example.

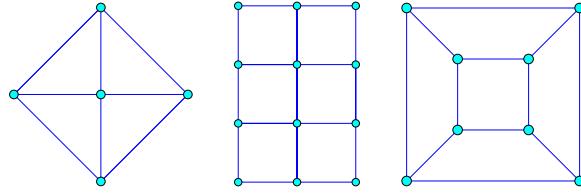


Figure 4: Examples of patterns used: From left to right: A wheel 4 pattern; a grid 3, 2 pattern; and a squares 2, 4 pattern.

### 9.2. Noise

Once a point from the pattern has been selected uniformly, that point is then perturbed before being added to the point cloud. We used two types of noise to perturb a point  $p$  lying on an edge  $e$ :

- **Uniform noise** with bound  $\mu$ : Two values  $d_1, d_2$  in the interval  $[-\mu, \mu]$  are uniformly selected, and then the point is moved a distance  $d_1$  in the direction perpendicular to  $e$ , and a distance  $d_2$  in the direction parallel to  $e$ .
- **Gaussian noise** with parameter  $\sigma$ : Two distances  $d_1, d_2$  are generated in correspondence with the Gaussian distribution with standard deviation  $\sigma$ , which has a probability density function  $f(x, \sigma) = \frac{1}{\sqrt{2\pi}\sigma^2}e^{-\frac{x^2}{2\sigma^2}}$ . The point is moved a distance  $d_1$  in the direction perpendicular to  $e$ , and a distance  $d_2$  in the direction parallel to  $e$ .

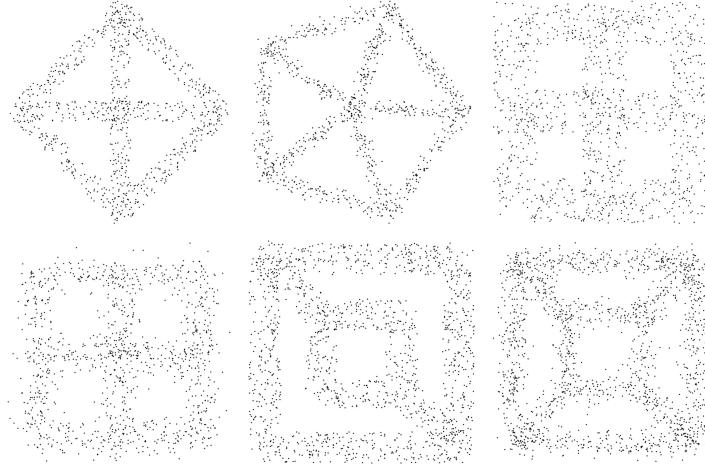


Figure 5: Examples of clouds in the dataset. Top-left: Wheel 4 pattern with uniform noise of 0.1; Top-middle: Wheel 5 pattern with Gaussian noise of 0.04; Top-right: Grid 2,2 pattern with uniform noise of 0.25; Bottom-left: Grid 2,2 pattern with Gaussian noise of 0.12; Bottom-middle: Squares 2,2 pattern with uniform noise of 0.2; Bottom-right: Squares 2,4 pattern with Gaussian noise of 0.1.

### 9.3. Size of Clouds

The size of a cloud is dependent upon the pattern from which the cloud is obtained. All of the clouds in our dataset have 100 points per unit length of the pattern.

### 9.4. Summary of the Dataset

Here is a list of the types of cloud present in the dataset. There are 200 randomly produced point clouds of each type.

- Wheel  $k$  pattern
  - $k$  runs from 3 to 9.
  - Uniform noise varies from 0.05 to 0.25 in 0.05 intervals.
  - Gaussian noise varies from 0.02 to 0.1 in 0.02 intervals.
  - The number of clouds from wheel patterns in the dataset is 14000.

- Grid  $k, l$  pattern
  - $k$  and  $l$  run from 1 to 3, with  $l \leq k$ , so the number of rows is always greater than or equal to the number of columns.
  - Uniform noise varies from 0.05 to 0.4 in 0.05 intervals.
  - Gaussian noise varies from 0.02 to 0.2 in 0.02 intervals.
  - The number of clouds from grid patterns in the dataset is 21600.
  
- Squares  $k, l$  pattern
  - $k$  is either 2 or 3, and  $l$  runs from 1 to 4.
  - Uniform noise varies from 0.05 to 0.25 in 0.05 intervals.
  - Gaussian noise varies from 0.02 to 0.1 in 0.02 intervals.
  - The number of clouds from concentric squares patterns in the dataset is 16000.

So the total number of clouds in the dataset is 51600.

## 10. Experimental Comparison, Discussion and Conclusion

### 10.1. Simplification of HoPeS

Usually a point cloud will contain many points. Since HoPeS extends a minimum spanning tree of a point cloud by adding critical edges, the HoPeS output will often have many noisy short branches that are unnecessary. Therefore, we sometimes wish to simplify the output to reduce the number of edges. This is done via a two-step process:

- Firstly, as all of our patterns in our dataset do not contain any vertices of degree 1, we remove all vertices of degree 1 (along with their corresponding edge). We do this iteratively to obtain a graph with no degree 1 vertices. This simplification process is not just done for HoPeS, but also for Mapper and for the  $\alpha$ -Reeb output (see Figures 6 and 7).

- Secondly, for a given threshold  $\epsilon$ , we iteratively collapse all edges with length less than  $\epsilon$ , starting from the shortest edge. (Any edge shorter than the threshold that is a side of a triangle is not collapsed, nor is an edge collapsed if such an operation would lead to the output having self-intersections. Therefore, we preserve the first homology group of the graph, see Corollary 10.1.) The threshold  $\epsilon$  that we use is the maximum death of any dot above the first widest gap in the persistence diagram  $\text{PD}\{C^\alpha\}$ . To collapse an edge  $e$ , we first identify the two vertices  $v_1, v_2$  that are the endpoints of  $e$ . We remove  $v_1, v_2$  and all edges incident on the two vertices. We now add a new vertex  $v$  to the graph. If  $\deg(v_1) = \deg(v_2) = 2$ , or  $\deg(v_1) \neq 2$  and  $\deg(v_2) \neq 2$ ,  $v$  is placed at the midpoint of  $v_1$  and  $v_2$ . On the other hand, if for instance  $\deg(v_1) \neq 2$  but  $\deg(v_2) = 2$ , then we place  $v$  at the position where  $v_1$  was, in order to best preserve the geometry of the output graph. We then add an edge from  $v$  to any original neighbour of  $v_1$  and  $v_2$  (see Figure 8).

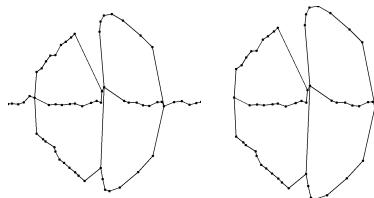


Figure 6: Left: An unsimplified Mapper output for a cloud from a wheel 4 pattern; Right: The simplified output by removing vertices of degree 1.

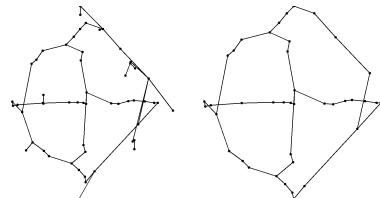


Figure 7: Left: An unsimplified  $\alpha$ -Reeb output for a cloud from a wheel 4 pattern; Right: The simplified output by removing vertices of degree 1.

**Corollary 10.1.** Under the conditions in Theorems 8.4 and 8.9, the simplified HoPeS output has the same first homology group as  $G$ .

*Proof.* Under the given conditions in Theorems 8.4 and 8.9, these theorems state that the HoPeS output has the same  $H_1$  as  $G$ . During simplification, the removal of degree 1 vertices and their corresponding edges does not change  $H_1$ . The second stage – collapsing short edges – would only change  $H_1$  if we collapse

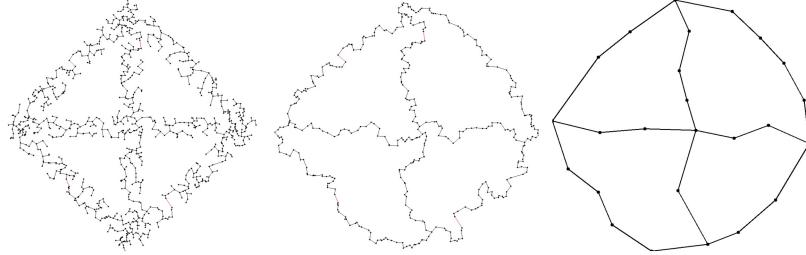


Figure 8: The stages of simplification of the HoPeS output. All graphs are for clouds produced from the wheel 4 pattern. Left: Unimplified HoPeS; Middle: HoPeS with degree 1 vertices removed; Right: Simplified HoPeS.

an edge of a triangle, or, if considering the output as an embedded graph in  $\mathbb{R}^2$ , if a collapse of an edge leads to self-intersections of the graph, but the algorithm prevents such operations. Therefore,  $H_1$  remains unchanged under simplification.  $\square$

### 10.2. Optimisation of Additional Parameters

Unlike HoPeS, Mapper and the  $\alpha$ -Reeb algorithm require additional parameters that must be optimised. For all types of clouds, we deemed it both realistic and fair to optimise these parameters over a fixed range of values that would not vary between different types of clouds.

For Mapper, the amount of overlap of the intervals in the covering of the image of the filter function we fixed at 50%. However, there were two parameters – the number of intervals and the  $\epsilon$  parameter used in DBSCAN (Section 4.1) – that we needed to optimise. For the number of intervals, we made it dependent on the number of points in the cloud, and so in fact we would be varying a parameter  $x$ , where, for a cloud with  $N$  points, the number of intervals equals  $\frac{xN}{100}$ , rounded to the nearest integer.  $x$  took 10 values, which varied between 1.5 and 3.3 in 0.2 increments.  $\epsilon$  also took 10 values, which varied between 0.05 and 0.5 in 0.05 increments. So in total we used 100 configurations of the parameters for Mapper.

For the  $\alpha$ -Reeb algorithm,  $\alpha$  took 10 values which varied between 0.15 and 0.6 in 0.05 increments.

### 10.3. Example Outputs

Both Mapper and  $\alpha$ -Reeb graphs are abstract graphs, and so do not naturally embed in  $\mathbb{R}^2$ . Therefore, in order to draw the graphs in the plane, it was necessary for us to embed them as naturally as possible.

For Mapper, vertices correspond to clusters of the point cloud. Therefore, it is natural to place vertices at the geometric centre of the clusters, by which we mean the coordinates of a vertex  $v_i$  corresponding to a cluster  $C_i$  will be  $(x, y) = \frac{1}{|C_i|} \sum_{p \in C_i} (p_x, p_y)$ .

For  $\alpha$ -Reeb graphs, it is far less natural to appropriately embed them in  $\mathbb{R}^2$ . This is because of the final stage in the algorithm, where we are taking the quotient of disjoint copies of intervals. This means that sometimes there is not a natural connection between vertices of the  $\alpha$ -Reeb graph and points in the point cloud, like there is for Mapper. This limits how naturally we can embed the graph, though we have done the best we can, and we suggest that this difficulty of embedding  $\alpha$ -Reeb graphs is another drawback of the algorithm.

Below are examples of outputs produced by the three algorithms. Since Mapper and  $\alpha$ -Reeb graphs are abstract, their outputs can have self-intersections when the graphs are embedded into  $\mathbb{R}^2$ . The  $\alpha$ -Reeb graph in Figure 9 testifies to this. Conversely, neither the HoPeS output before nor after simplification can have self-intersections (Corollary 10.1).

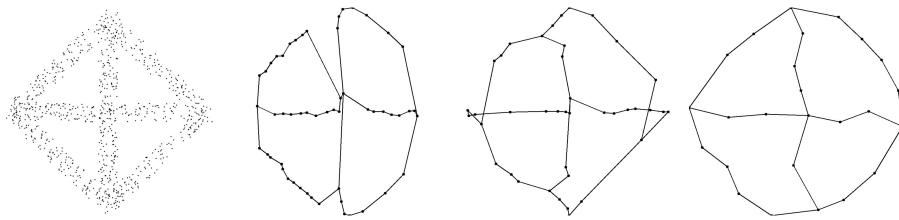


Figure 9: Left: A cloud produced from a wheel 4 pattern with uniform noise of a magnitude of 0.1. The other three images are the corresponding outputs of the algorithms. Middle-left: Mapper; Middle-right:  $\alpha$ -Reeb; Right: HoPeS.

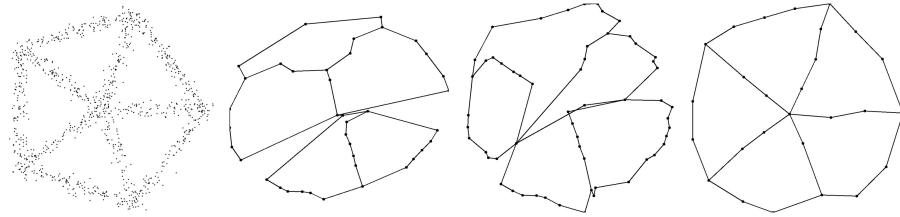


Figure 10: Left: A cloud produced from a wheel 5 pattern with Gaussian noise of a magnitude of 0.04. The other three images are the corresponding outputs of the algorithms. Middle-left: Mapper; Middle-right:  $\alpha$ -Reeb; Right: HoPeS.

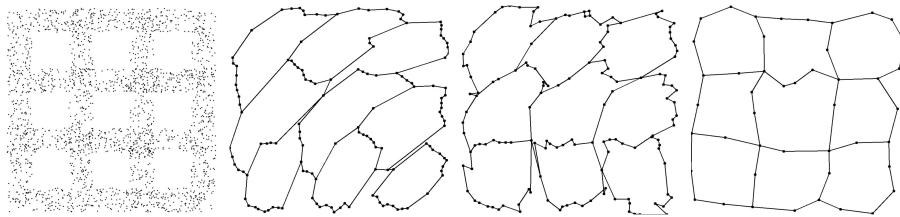


Figure 11: Left: A cloud produced from a grid 3, 3 pattern with uniform noise of a magnitude of 0.2. The other three images are the corresponding outputs of the algorithms. Middle-left: Mapper; Middle-right:  $\alpha$ -Reeb; Right: HoPeS.

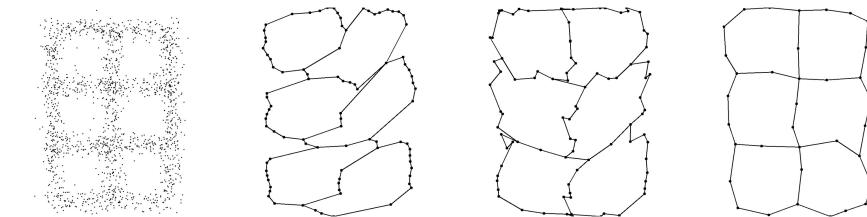


Figure 12: Left: A cloud produced from a grid 3, 2 pattern with Gaussian noise of a magnitude of 0.1. The other three images are the corresponding outputs of the algorithms. Middle-left: Mapper; Middle-right:  $\alpha$ -Reeb; Right: HoPeS.

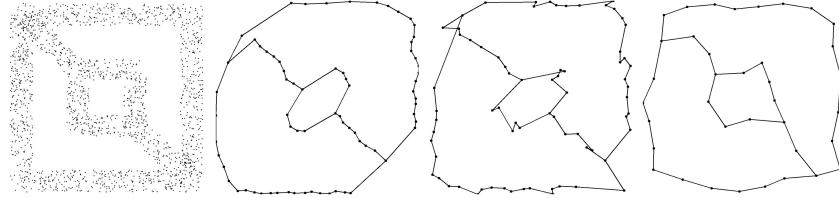


Figure 13: Each output here is for a squares 2,2 pattern with uniform noise of a magnitude of 0.2. Left: Mapper; Middle:  $\alpha$ -Reeb; Right: HoPeS.

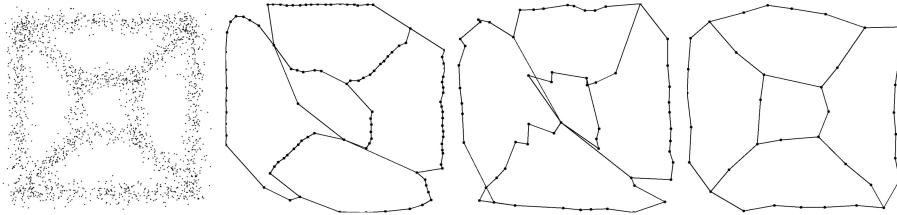


Figure 14: Left: A cloud produced from a squares 2,4 pattern with Gaussian noise of a magnitude of 0.1. The other three images are the corresponding outputs of the algorithms. Middle-left: Mapper; Middle-right:  $\alpha$ -Reeb; Right: HoPeS.

#### 10.4. Criteria for the Experimental Comparison, and the Results

We chose 4 criteria with which to compare Mapper, the  $\alpha$ -Reeb algorithm, and HoPeS.

- **Betti success rate:** If there are dominant cycles implicitly present in a point cloud, then it is important that a data skeletonisation algorithm captures this information. Therefore, the first criteria is whether the output has the correct first Betti number. Specifically, for a point cloud  $C$  formed from a pattern  $P$  with first Betti number  $k$ , we check whether the first Betti number of the output of the algorithm when  $C$  is inputted is also  $k$ . Since our dataset contains 200 examples for each type of cloud, we carry out this check on the 200 outputs for each algorithm to obtain a Betti success rate – the chance for a given type of cloud that the algorithm produces an output that has the same first Betti number as  $P$ .

- **Homeomorphism success rate:** This is an extension of the first criteria. Namely, we are checking whether the output is homeomorphic to the pattern  $P$ . We check this only for outputs that already are successful for the first Betti number, and so obtain the homeomorphism success rate – the chance that an output agreeing with the first Betti number of  $P$  is also homeomorphic to  $P$ .
- **The root mean square error:** This criteria reveals how close geometrically an output is to the point cloud  $C$ . For each point  $p$  in  $C$ , we compute the distance,  $\min_p$ , from  $p$  to the closest edge of the output. Then the RMS error =  $\sqrt{\sum_{p \in C} \min_p^2}$ . For each type of cloud, we take the mean RMS error over all outputs that are successful on the first Betti number.
- **Time:** This is simply the mean time for the algorithm to produce an output when given a cloud.

Below is a selection of the results obtained. We see from the graphs below that in general it is HoPeS that outperforms Mapper and the  $\alpha$ -Reeb algorithm.

In terms of Betti success rates, HoPeS often performs slightly better than Mapper, while the  $\alpha$ -Reeb algorithm usually performs less well. We comment that HoPeS is not quite as strong on Gaussian noise, suggesting that it may struggle if the data has outliers, particularly within cycles. It can also struggle on squares patterns, which is likely to be due to the more unusual shapes of the holes.

For homeomorphism success rates, HoPeS outperforms the other two algorithms across the board. We also see that, according to the RMS errors, HoPeS usually does better at representing the data geometrically. Finally, the overall better performance of the HoPeS algorithm is not payed for in the runtime, since the runtime of the HoPeS algorithm is less than the Mapper algorithm, which in turn is less than the  $\alpha$ -Reeb algorithm.

We highlight the difficulty it is to select good values for the parameters of Mapper and the  $\alpha$ -Reeb algorithm so that their output graphs agree on the first

Betti number with the pattern that the point cloud came from. If parameters such as  $\alpha$  in the  $\alpha$ -Reeb algorithm, or  $\epsilon$ , used for DBSCAN in the Mapper algorithm, are too small, then noisy cycles will also be captured in addition to the dominant cycles. If such parameters are too large, then even the dominant cycles will not be captured. HoPeS does not have this issue, since it does not have any additional parameters.

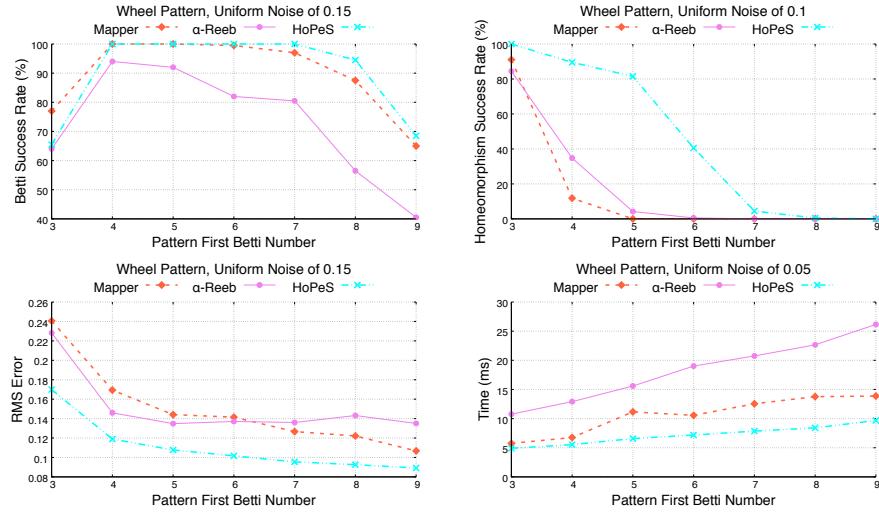


Figure 15: All results here are for clouds from wheel patterns with uniform noise. Along the  $x$ -axis of these graphs we have the first Betti number of the pattern that produced the cloud. Top-left: Betti success rate; Top-right: Homeomorphism success rate; Bottom-left: RMS error; Bottom-right: Time.

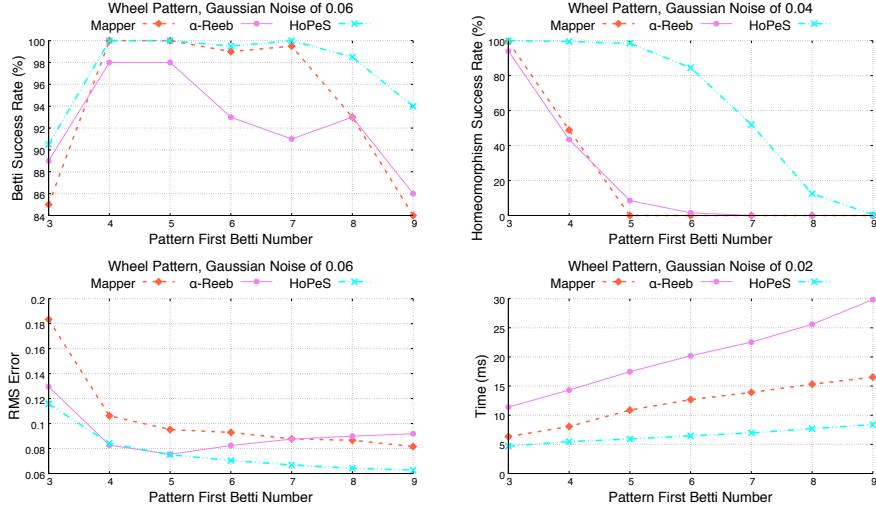


Figure 16: All results here are for clouds from wheel patterns with Gaussian noise. Along the  $x$ -axis of these graphs we have the first Betti number of the pattern that produced the cloud. Top-left: Betti success rate; Top-right: Homeomorphism success rate; Bottom-left: RMS error; Bottom-right: Time.

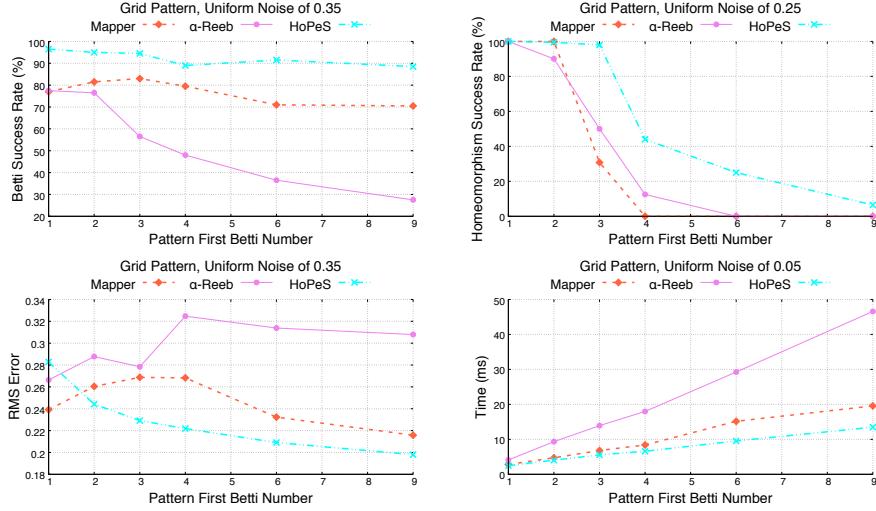


Figure 17: All results here are for clouds from grid patterns with uniform noise. Along the  $x$ -axis of these graphs we have the first Betti number of the pattern that produced the cloud. Top-left: Betti success rate; Top-right: Homeomorphism success rate; Bottom-left: RMS error; Bottom-right: Time.

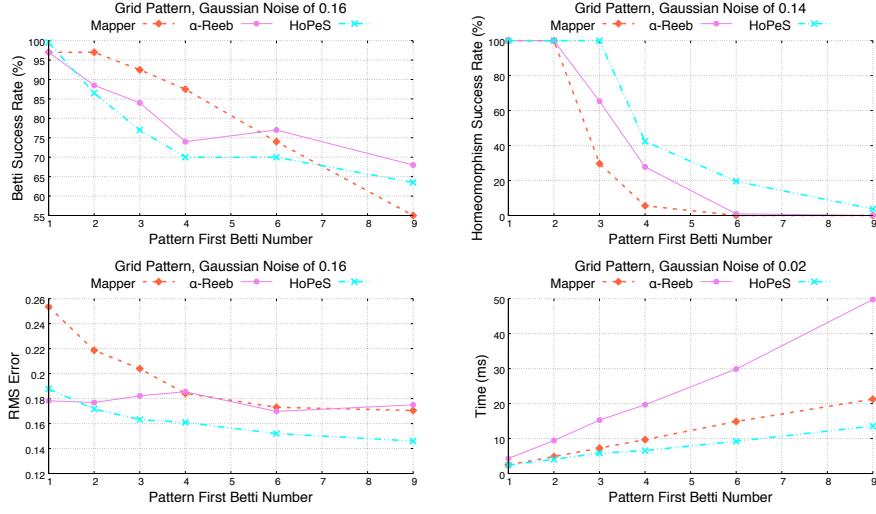


Figure 18: All results here are for clouds from grid patterns with Gaussian noise. Along the  $x$ -axis of these graphs we have the first Betti number of the pattern that produced the cloud. Top-left: Betti success rate; Top-right: Homeomorphism success rate; Bottom-left: RMS error; Bottom-right: Time.

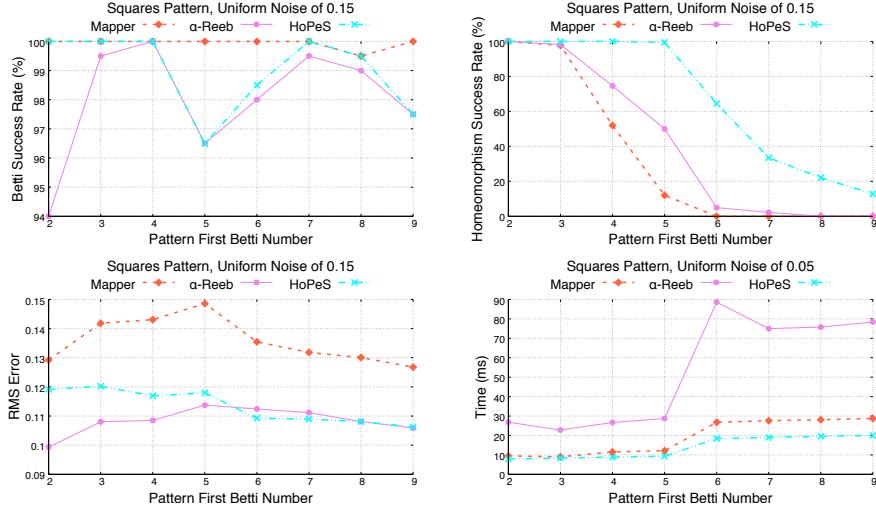


Figure 19: All results here are for clouds from squares patterns with uniform noise. Along the  $x$ -axis of these graphs we have the first Betti number of the pattern that produced the cloud. Top-left: Betti success rate; Top-right: Homeomorphism success rate; Bottom-left: RMS error; Bottom-right: Time.

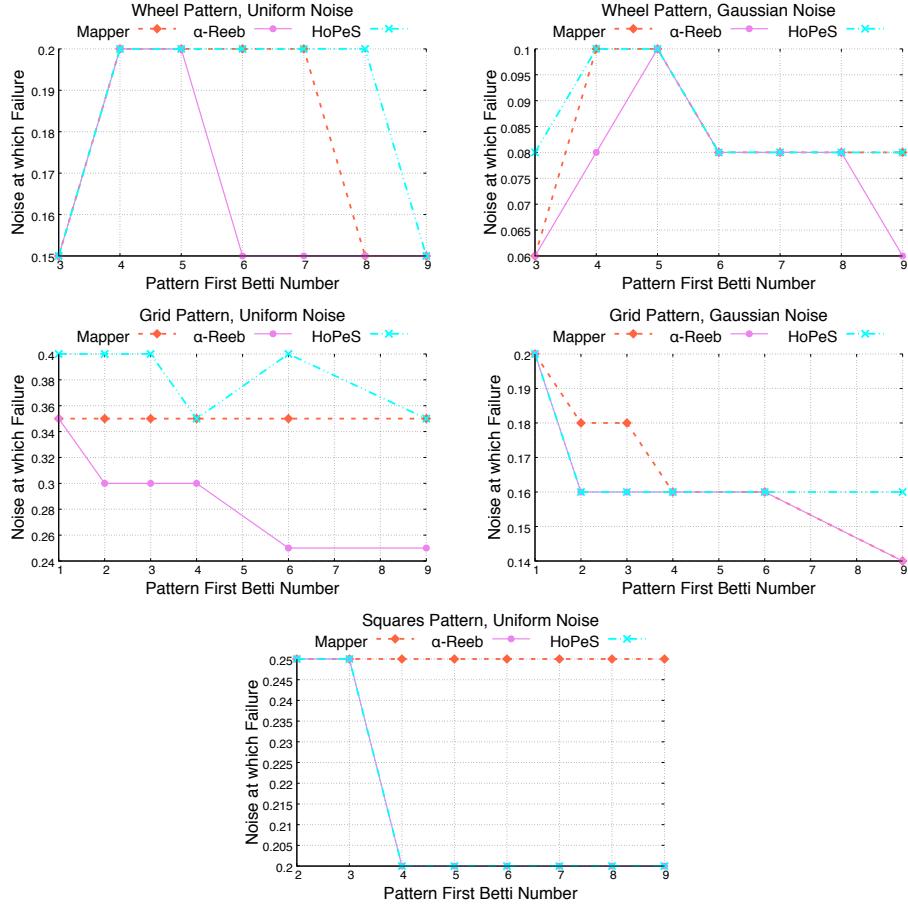


Figure 20: These graphs show the magnitude of noise at which the Betti success rate drops below 90%. Top-left: wheel pattern with uniform noise; Top-right: wheel pattern with Gaussian noise; Middle-left: grid pattern with uniform noise; Middle-right: grid pattern with Gaussian noise; Bottom: squares pattern with uniform noise.

### 10.5. Outputs on Real Data

Alongside our experiments on synthetic data, we can also compare these algorithms on real data. Using images from the BSDS500, we detected boundaries in the image along which there are significant colour changes. We could then extract from these colour boundaries a point cloud that we inputted into the algorithms. Below is an example of this for a picture of a woman.

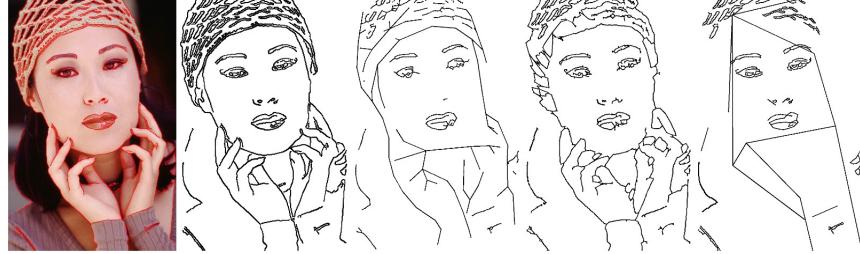


Figure 21: From left to right: Real picture with colour boundaries in red; Extracted point cloud; Mapper output;  $\alpha$ -Reeb output; Simplified HoPeS output.

Furthermore, we ran the algorithms over the entire BSDS500 database, which consists of 500 real images, and for each output, we calculated the RMS error of the output graph with respect to the point cloud. Below is a bar chart showing each algorithm's mean RMS error, and we see that again it is HoPeS that outperforms the other two algorithms.

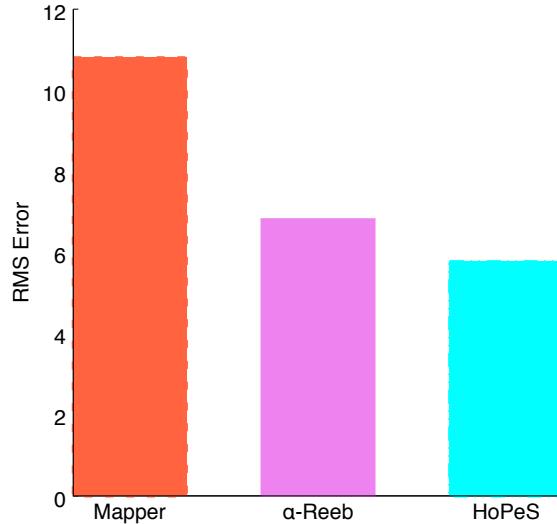


Figure 22: This barchart shows the mean RMS error for each algorithm over all real images from the BSDS500 database. HoPeS's RMS error is less than that of the  $\alpha$ -Reeb algorithm's, with both having a significantly less RMS error than Mapper.

### 10.6. Conclusion

In conclusion, we have presented here a comparison of 3 data skeletonisation algorithms – Mapper, the  $\alpha$ -Reeb algorithm and HoPeS.

With HoPeS, we have removed a potential ambiguity when assigning death values to critical edges. Moreover, we have proved guarantees of the HoPeS algorithm relating to its optimality and the reconstruction.

Finally, we have carried out a comparison of the three algorithms' abilities via four criteria, which has revealed that in general HoPeS outperforms the other two algorithms, as well as being easier to use due to the lack of additional parameters. The dataset and C++ code will be made freely available soon.

## References

### References

- [1] F. Chazal, V. de Silva, S. Oudot, Persistence stability for geometric complexes, *Geometriae Dedicata* 173 (2014) 193–214.
- [2] A. Hatcher, *Algebraic Topology*, Cambridge University Press, 2001.
- [3] R. Singh, V. Cherkassky, N. Papanikopoulos, Self-organizing maps for the skeletonization of sparse shapes, *IEEE Tran. Neural Networks* 11 (2000) 241–248.
- [4] B. Kégl, A. Krzyzak, Piecewise linear skeletonization using principal curves, *Tran. PAMI* 24 (2002) 59–74.
- [5] X. Ge, I. Safa, M. Belkin, Y. Wang, Data skeletonization via reeb graphs, in: *Proceedings of NIPS*, 2011.
- [6] M. Aanjaneya, F. Chazal, D. Chen, M. Glisse, L. Guibas, D. Morozov, Metric graph reconstruction from noisy data, *Int. J. Comp. Geometry Appl.* 22 (2012) 305–325.

- [7] M. Carrière, S. Oudot, Structure and stability of the 1-dimensional mapper, *Foundations of Computational Mathematics* 18 (6) (2018) 1333–1396.
- [8] F. Chazal, J. Sun, Gromov-hausdorff approximation of filament structure using reeb-type graph, *Discrete Computational Geometry* 53 (2015) 621–649.
- [9] T. K. Dey, F. Fan, Y. Wang, Graph induced complex on point data, *Computational Geometry* 48 (2015) 575–588.
- [10] A. Chernov, V. Kurlin, Reconstructing persistent graph structures from noisy images, *Image-A* 3 (2013) 19–22.
- [11] V. Kurlin, A fast and robust algorithm to count topologically persistent holes in noisy clouds, in: *Proceedings of CVPR, 2014*, pp. 1458–1463.
- [12] V. Kurlin, Auto-completion of contours in sketches, maps and sparse 2d images based on topological persistence, in: *Proceedings of SYNASC workshop CTIC: Computational Topology in Image Context, 2014*, pp. 594–601.
- [13] V. Kurlin, A homologically persistent skeleton is a fast and robust descriptor for a sparse cloud of interest points in noisy 2d images, in: *Proceedings of CAIP, 2015*.
- [14] G. Singh, F. Mémoli, G. Carlsson, Topological methods for the analysis of high dimensional data sets and 3d object recognition, *Eurographics Symposium of Point-Based Graphics* (2007) 91–100.
- [15] M. Ester, H. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, AAAI Press, 1996, pp. 226–231.
- [16] S. Kališnik, V. Kurlin, D. Lešnik, A higher-dimensional homologically persistent skeleton, *Advances in Applied Mathematics* 102 (2019) 113 – 142.