SimBa: An Efficient Tool for Approximating Rips-Filtration Persistence via *Sim*plicial *Ba*tch-Collapse*

Tamal K. Dey[1], Dayu Shi[2], and Yusu Wang[3]

1 **Dept. of Computer Science and Engineering, and Dept. of Mathematics, The Ohio State University, Columbus, OH, USA**
tamaldey@cse.ohio-state.edu

2 **Dept. of Computer Science and Engineering, and Dept. of Mathematics, The Ohio State University, Columbus, OH, USA**
shiday@cse.ohio-state.edu

3 **Dept. of Computer Science and Engineering, and Dept. of Mathematics, The Ohio State University, Columbus, OH, USA**
wang.1016@osu.edu

Abstract

In topological data analysis, a point cloud data P extracted from a metric space is often analyzed by computing the persistence diagram or barcodes of a sequence of Rips complexes built on P indexed by a scale parameter. Unfortunately, even for input of moderate size, the size of the Rips complex may become prohibitively large as the scale parameter increases. Starting with the *Sparse Rips filtration* introduced by Sheehy, some existing methods aim to reduce the size of the complex so as to improve the time efficiency as well. However, as we demonstrate, existing approaches still fall short of scaling well, especially for high dimensional data. In this paper, we investigate the advantages and limitations of existing approaches. Based on insights gained from the experiments, we propose an efficient new algorithm, called *SimBa*, for approximating the persistent homology of Rips filtrations with quality guarantees. Our new algorithm leverages a batch collapse strategy as well as a new sparse Rips-like filtration. We experiment on a variety of low and high dimensional data sets. We show that our strategy presents a significant size reduction, and our algorithm for approximating Rips filtration persistence is order of magnitude faster than existing methods in practice.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Rips filtration, Homology groups, Persistence, Topological data analysis

Digital Object Identifier 10.4230/LIPIcs.ESA.2016.35

1 Introduction

In recent years, topological ideas and methods have emerged as a new paradigm for analyzing complex data [8, 24]. An important line of work in this direction is the theory and applications of persistent homology. It provides a powerful and flexible framework to inspect data for characterizing and summarizing important features that persist across different scales. Since its introduction [25, 26, 34], there have been many fundamental developments [7, 9, 10, 12, 14, 17, 18, 19, 39] both to generalize the framework and to provide theoretical understanding

* This work has been supported by NSF grants CCF-1318595 and CCF-1526513.

for various aspects of it (such as its stability). These developments help to provide foundation and justification of the practical usage of persistent homology; see e.g, [13, 15, 22, 37, 33].

A determining factor in applying persistent homology to a broad range of data analysis problems is the availability of efficient and scalable software. Given the rapidly increasing size of modern data, the "efficiency" necessarily concerns both time and space complexities. The original algorithm to compute persistent homology takes $O(n^3)$ time and $O(n^2)$ space for a filtration involving n number of total simplices [25]. Various practical improvements have been suggested [16, 20]. An early software widely used for computing persistence is Morozov's Dionysus [23]. Later, Bauer et al. developed the PHAT toolbox [3], based on several efficient matrix reduction strategies (mostly focusing on time efficiency) as described in [2]. A more recently developed library called GUDHI [38] considers the improvement both in time and space efficiencies. In particular, it uses an efficient data structure, called the simplex tree [5], to encode input simplicial complexes, and uses the *compressed annotation matrix* [4] to implement the persistent cohomology algorithm. Dionysus, PHAT, and GUDHI offer efficient software for computing persistence induced by inclusions. For our algorithm, we need persistence induced by more general simplicial maps for which we use Simpers [36] developed on the basis of the algorithm in [21].

The above results and software cater to general persistence computations. In practice, often the persistence needs to be computed for a particular filtration called the *Vietoris-Rips* or *Rips filtration* in short. Given a set of points P embedded in $\mathbb{R}^d$ (or in more general metric spaces), the Rips complex $\mathcal{R}^\alpha(P)$ with radius or scale α is the clique complex induced by the set of edges $\{(p, p') \mid d(p, p') \leq \alpha, p, p' \in P\}$. One is interested in the persistent homology induced by the sequence of Rips complexes $\mathcal{R}^{\alpha_1} \subseteq \mathcal{R}^{\alpha_2} \subseteq \cdots \subseteq \mathcal{R}^{\alpha_m}$ for a growing sequence of radii $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_m$. Intuitively, the Rips complex at a specific scale α approximates the union of radius-α balls around sample points in P. Thus, it captures the structure formed by input points P at different scales.

Unfortunately, even for a modest size of n (in the range of thousands), the size of Rips complex (as well as the slightly more economical Čech complex) becomes prohibitively large as the radius α increases. In [35], Sheehy proposed an elegant solution for this problem by introducing a *sparse Rips filtration* to approximate the persistent homology of the Rips filtration for a set of points P. An alternative approach of collapsing input points in batches with increasing radius α was proposed in [21], which leveraged the persistence algorithm proposed in the same paper for filtrations arising out of simplicial maps.

New work

Given the importance of the Rips filtration in practice, our goal is to investigate the practical performance of the existing proposed methods, understand their advantages and limitations, and develop an efficient implementation for approximating the persistent homology of Rips filtrations. To this end, we make the following contributions.

1. We investigate the advantages and limitations of three existing methods, two based on Sparse Rips [35, 11], and another on Batch-collapse [21]. Specifically, experiments show that while the sparse Rips algorithm by Sheehy [35] has a theoretical guarantee on the size of the filtration and gives good approximation of the persistence diagrams for the Rips filtration in practice, it generates simplicial complexes of large size even for input of moderate size. This problem becomes more severe as the dimension of the input data increases. The algorithm fails to finish for several high dimensional data sets of rather moderate size. See Table 1 for some examples. The batch-collapse approach is much more space efficient (which leads to time efficiency as well). Nevertheless, we find that its size still becomes prohibitive for high dimensional data.

2. Based on the insights gained from experimenting with the existing approaches, we propose a new algorithm called *SimBa* that approximates a Rips filtration persistence via simplicial batch-collapses. Our algorithm is a modification of the previous batch-collapse of Rips filtration proposed in [21]. While theoretically, the modification may not seem major, empirically, it reduces the size of the filtration significantly and thus leads to a much more efficient approximation of the Rips filtration persistence. Furthermore, we show that this modification maintains a similar approximation guarantee as the batch-collapse of Rips filtration proposed in [21]. We describe the details of an efficient and practical implementation of SimBa, the software for which has been made publicly available from [36].

Two concepts, homology groups of a simplicial complex, and simplicial maps between two complexes are used throughout this paper. We refer the reader to any standard text such as [32] for details. We denote the p-dimensional homology group of a simplicial complex $\mathcal{K}$ under $\mathbb{Z}_2$ coefficients by $H_p(\mathcal{K})$.

2 Rips filtration and its approximation

Given a set of points $P \subset \mathbb{R}^d$, let $\langle p_0, \ldots, p_s \rangle$ denote the s-dimensional simplex spanned by vertices $p_0, \ldots, p_s \in P$. The Rips complex at scalar α is defined as $\mathcal{R}^\alpha(P) = \{\langle p_0, \ldots, p_s \rangle \mid \|p_i - p_j\| \leq \alpha$, for any $i, j \in [0, s]\}$. Now consider the following *Rips filtration*:

$$\{\mathcal{R}^\alpha(P)\}_{\alpha>0} \;:=\; \mathcal{R}^{\alpha_1}(P) \hookrightarrow \mathcal{R}^{\alpha_2}(P) \cdots \hookrightarrow \mathcal{R}^{\alpha_n}(P) \cdots \tag{1}$$

The inclusion maps between consecutive complexes above induce homomorphisms between respective homology groups, giving rise to a so called *persistence module* for dimension p:

$$H_p(\mathcal{R}^{\alpha_1}(P)) \to H_p(\mathcal{R}^{\alpha_2}(P)) \to \ldots \to H_p(\mathcal{R}^{\alpha_n}(P)) \cdots \tag{2}$$

If a homology class is created at $\mathcal{R}^{\alpha_i}(P)$ (i.e, does not have pre-image under homomorphism $H_p(\mathcal{R}^{\alpha_{i-1}}) \to H_p(\mathcal{R}^{\alpha_i})$) and dies entering $\mathcal{R}^{\alpha_j}(P)$ (i.e, its image vanishes under homomorphism $H_p(\mathcal{R}^{\alpha_{j-1}}(P)) \to H_p(\mathcal{R}^{\alpha_j}(P))$, then α_i is its *birth time*, α_j is its *death time*, and the difference $\alpha_j - \alpha_i$ is called the *persistence* of the class. In each dimension, the persistence barcodes capture the persistence of such homology classes by using a horizontal bar with left and right end points at α_i and α_j respectively. These *persistence barcodes* of the above Rips filtration are often the target summary of P and/or of the space P samples, which one wishes to compute in topological data analysis.

The main bottleneck for computing the barcodes of a Rips filtration stems from its size blowup. As the parameter α grows, the Rips complex $\mathcal{R}^\alpha(P)$ can become huge very quickly. To address this blowup in size, Sheehy [35] suggested a novel approach of sparsifying the point set P as one proceeds with increasing α in a way that does not alter the barcodes too much. The idea is to replace the original Rips filtration $\{\mathcal{R}^\alpha(P)\}_{\alpha>0}$ on P with a sequence of smaller complexes $\{\mathcal{S}^\alpha\}_{\alpha>0}$ and show that the two sequences *interleave* at the homology level. Then, appealing to the results of interleaving persistence modules [12], one can show that the barcodes of $\{\mathcal{S}^\alpha\}_{\alpha>0}$ approximate those of $\{\mathcal{R}^\alpha(P)\}_{\alpha>0}$ reasonably. The complexes $\mathcal{S}^\alpha$ are constructed as the *union* of Rips-like complexes built on a sequence of subsets of P rather than the entire set P.

The union allows the complexes in $\{\mathcal{S}^\alpha\}_{\alpha>0}$ to be connected with inclusions and hence permits using efficient algorithms and software designed for inclusion induced persistence. However, the size of $\mathcal{S}^\alpha$ may still be large due to the union operation. An alternative is to

avoid the union operation but allow deletion or collapse of vertices (and simplices) at larger scale α [11, 35] resulting into a sequence of Rips-like complexes connected with simplicial maps instead of inclusions. This approach, which we refer to as *Sparse Rips with collapse*, however achieves only moderate improvements in size reduction. We find that much more aggressive size reduction can be achieved by considering the collapse in a batched fashion that gives rise to the approach of *Batch-collapsed Rips* [21].

Finally, building on the batch-collapse idea, we propose a new approach, called *SimBa* that significantly reduces the size of Rips-like complexes and their computations. This is achieved primarily by replacing inter-point distances with *set distances* while computing the complexes. We prove that this approach still provably approximates the barcodes of the original Rips filtration in sequence (1).

In what follows, we provide more details about each existing method along with its performance in practice, which explains the motivation behind the new tool SimBa.

2.1 Sparse Rips filtration (inclusions)

Let P be a set of points in a metric space $(\mathcal{M}, d)$. A *greedy permutation* $\{p_1, .., p_n\}$ of P is defined recursively as follows: Let $p_1 \in P$ be any point and define p_i recursively as $p_i = \text{argmax}_{p \in P \setminus P_{i-1}} d(p, P_{i-1})$, where $P_{i-1} = \{p_1, ..., p_{i-1}\}$. This gives rise to a nested sequence of subsets $P_1 \subset P_2 \subset \cdots P_n = P$. Furthermore, each subset P_i is locally dense and uniform (net) in the following sense. Define the insertion radius λ_{p_i} of a point p_i as $\lambda_{p_i} = d(p_i, P_{i-1})$. Each P_i is a λ_{p_i}-net of P, meaning that $d(p, P_i) \leq \lambda_{p_i}$ for every $p \in P$ and $d(p, q) \geq \lambda_{p_i}$ for every distinct pairs $p, q \in P_i$. These nets can be extended to a single-parameter family of nets as $\{N_\gamma\}$ where $N_\gamma = \{p \in P | \lambda_p > \gamma\}$ is a γ-net of P.

Using the idea of Sheehy [35], Buchet et al. [6] define a Rips-like filtration using the above specific nets and assigning weights to points whose geometric interpretation is explained in [11]. Each point $p \in P$ is associated with a weight $w_p(\alpha)$ at scale α as

$$w_p(\alpha) = \begin{cases} 0 & \text{if } \alpha \leq \frac{\lambda_p}{\varepsilon} \\ \alpha - \frac{\lambda_p}{\varepsilon} & \text{if } \frac{\lambda_p}{\varepsilon} < \alpha \leq \frac{\lambda_p}{\varepsilon(1-\varepsilon)} \\ \varepsilon\alpha & \text{if } \frac{\lambda_p}{\varepsilon(1-\varepsilon)} \leq \alpha \end{cases}$$

where $0 < \varepsilon < 1$ is an input constant that controls the sparsity of the filtration. Then, the perturbed distance between pairs of points is defined as

$$\hat{d}_\alpha(p, q) = d(p, q) + w_p(\alpha) + w_q(\alpha).$$

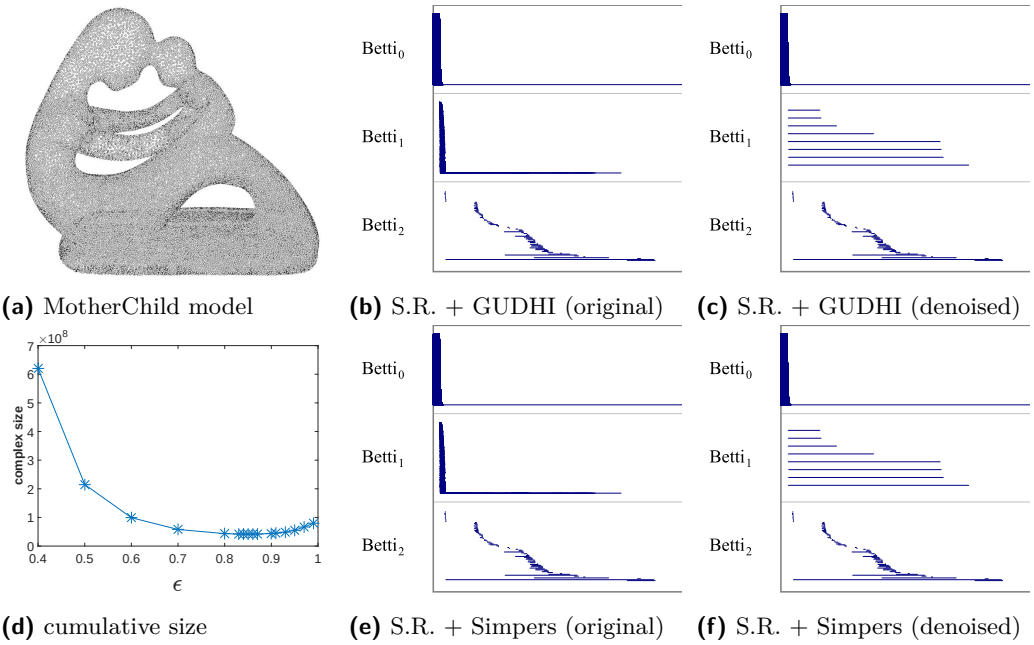
Using the perturbed distance $\hat{d}_\alpha$, the Sparse Rips complex at scale α is defined as

$$\mathcal{Q}^\alpha = \{\sigma \subset N_{\varepsilon(1-\varepsilon)\alpha} \mid \forall p, q \in \sigma,\ \hat{d}_\alpha(p, q) \leq 2\alpha\}.$$

The sequence of $\{\mathcal{Q}^\alpha\}_{\alpha>0}$ does not form a nested sequence of spaces because the vertex set of each $\mathcal{Q}^\alpha$ comes from the net $N_{\varepsilon(1-\varepsilon)\alpha}$ and may decrease as α increases. However, one can take $\mathcal{S}^\alpha = \bigcup_{\alpha' \leq \alpha} \mathcal{Q}^{\alpha'}$ and build a natural filtration $\{\mathcal{S}^\alpha \hookrightarrow \mathcal{S}^{\alpha'}\}_{\alpha<\alpha'}$ connected by inclusions. It is shown that the persistence barcodes of the filtration $\{\mathcal{S}^\alpha\}_{\alpha>0}$ approximate those of the Rips filtration $\{\mathcal{R}^\alpha\}_{\alpha>0}$ [35].

We use the code from [6] to compute this sparse Rips filtration $\{\mathcal{S}^\alpha\}_{\alpha>0}$. We then use GUDHI [38] to compute its persistent barcodes as GUDHI has the state-of-the-art performance for handling large complexes due to a compression technique [4] for inclusion-based filtrations.

As a common test case to illustrate the performance of various existing methods, we use a 3-dimensional point set sampled from a surface model called MotherChild; see Figure



(a) MotherChild model **(b)** S.R. + GUDHI (original) **(c)** S.R. + GUDHI (denoised)

(d) cumulative size **(e)** S.R. + Simpers (original) **(f)** S.R. + Simpers (denoised)

Figure 1 MotherChild surface model and its persistence barcodes computed by Sparse Rips (S.R.) based approaches. Since the surface has genus 4, its barcodes contain long bars: 1 for H_0, 8 for H_1, and 1 for H_2. The minimum cumulative size for complexes, which is about 43 million, is achieved around $\varepsilon = 0.8$.

1a. We choose this model because the ground truth is available and also because existing methods have trouble (to different degrees) handling high-dimensional data. The size of the point set is 23075. For indicating memory consumption, we refer to *cumulative size* which is the total number of simplices arising in the filtration, and also to *maximum size* which is the maximum over all complexes arising in the filtration. For Sparse Rips filtrations two sizes coincide at the last complex due to inclusions. Figure 1d shows the cumulative size with different Sparse Rips parameter ε. It is minimum when ε is between 0.8 and 0.9. So, we choose $\varepsilon = 0.8$ to achieve the best performance while observing that the approximation quality does not suffer much as predicted by the theory.

The original persistence barcodes are shown in Figure 1b. Since it becomes hard to see the main (long) bars in presence of all spurious ones creating excessive overlaps, we remove all short bars whose ratio between death and birth time is smaller than a threshold for 1-dimensional homology group H_1. The bars for H_0 and H_2 are not denoised. Unless specified otherwise, all barcodes are denoised in the same way. The denoised barcodes are shown in Figure 1c, one for H_0, four short and four long for H_1 (MotherChild has genus 4), and one for H_2. The cumulative size of the Sparse Rips complexes in the filtration is 43.5 million and the total time cost is about 350 seconds.

2.2 Sparse Rips with collapse

The persistence barcodes for the inclusion-based filtration $\{\mathcal{S}^\alpha\}_{\alpha>0}$ are the same as the barcodes of the filtration $\{\mathcal{Q}^\alpha\}_{\alpha>0}$ connected by simplicial maps $\mathcal{Q}^\alpha \to \mathcal{Q}^{\alpha'}$ for $\alpha < \alpha'$. Specifically, these simplicial maps originate from vertex collapses defined by the following

projection map:

$$\mu_\alpha(p) = \begin{cases} p & \text{if } p \in N_{\varepsilon(1-\varepsilon)\alpha} \\ \underset{q \in N_{\varepsilon\alpha}}{\operatorname{argmin}}\, d(p,q) & \text{otherwise} \end{cases}$$

For any scale α, the projection μ_α maps the points of P to the net $N_{\varepsilon(1-\varepsilon)\alpha} \supseteq N_{\varepsilon\alpha}$. One can view it as p being deleted at time (scale) $\alpha_p = \frac{\lambda_p}{\varepsilon(1-\varepsilon)}$. We can construct the sequence of Sparse Rips complexes $\{\mathcal{Q}^\alpha\}_{\alpha>0}$ connected with simplicial maps induced by insertions and vertex collapses as α increases: Specifically, we delete the vertex p and all its incident simplices by collapsing it to its projection $\mu_{\alpha_p}(p)$ where $\alpha_p = \frac{\lambda_p}{\varepsilon(1-\varepsilon)}$, when entering complex $\mathcal{Q}^{\alpha_p}$. See [11] for more details.

In this approach we need to compute the persistence induced with simplicial maps. For this, we use the only available software Simpers [36] based on the algorithm presented in [21]. Our experimental results on the MotherChild model of Figure 1a with $\varepsilon = 0.8$ are given in Figure 1e and 1f. The barcodes are exactly the same as those in Figure 1b and 1c. The cumulative size of the entire sequence is the same, 43.5 million, because the final complex in $\mathcal{S}^\alpha$ is the union of all complexes in $\{\mathcal{Q}^\alpha\}_{\alpha>0}$. However, the maximum size in the sequence is 24.9 million due to vertex collapses in contrast to the maximum size for $\{\mathcal{S}^\alpha\}_{\alpha>0}$ which equals the cumulative size. The time cost for this approach is 463.7 seconds which is larger than that for Sparse Rips with GUDHI. So compared to Sparse Rips with GUDHI, this approach has smaller maximum size due to collapse but costs more time for computing persistence since Simpers computes persistence over collapses which are slower operations than inclusions.

While the size of these Sparse Rips complexes is linear in the number of input points, the hidden constant factor depends exponentially on the doubling dimension of the metric space where points are sampled from. Empirically, we note that the size is still large, and becomes much worse as the dimension of data increases. For example, for the Gesture Phase data in Table 1 which has only 1747 points in $\mathbb{R}^{18}$, the cumulative size of the Sparse Rips filtration is 45.6 million, which approaches the limit GUDHI or Simpers can handle. For other larger data sets such as Primary Circle or Survivin, the complex reaches a size beyond this limit. Moreover, one has to pre-compute a greedy permutation of the input point set before constructing the Sparse Rips filtration. This computation is usually costly requiring furthest point computations for which software as efficient as ANN (for nearest neighbors) is not available. This motivates us to consider the batched approach considered next.

2.3 Batch-collapsed Rips

For handling large and high dimensional data, we need a more aggressive sparsification than the Sparse Rips filtration. We consider the Batch-collapsed Rips filtration, which has been proposed previously in [21] (section 6.1).

Given a set of points P, first set $V_0 := P$ and compute the shortest pairwise distance α. We next construct a sequence of vertex sets $V_k, k \in [0, m]$ such that V_{k+1} is an αc^{k+1}-net of V_k where $c > 1$ is a parameter that controls the rate of the scale increase. Consider the vertex map $\pi_k : V_k \to V_{k+1}$, for $k \in [0, m-1]$, such that for any $v \in V_k$, $\pi_k(v)$ is v's nearest neighbor in V_{k+1}. It can be shown that each vertex map π_k induces a well-defined simplicial map $s_k : \mathcal{R}^{\alpha c^k \frac{3c-1}{c-1}}(V_k) \to \mathcal{R}^{\alpha c^{k+1} \frac{3c-1}{c-1}}(V_{k+1})$. The *Batch-collapsed Rips filtration* is:

$$\mathcal{R}^0(V_0) \xrightarrow{s_0} \mathcal{R}^{\alpha c \frac{3c-1}{c-1}}(V_1) \cdots \xrightarrow{s_{m-1}} \mathcal{R}^{\alpha c^m \frac{3c-1}{c-1}}(V_m). \tag{3}$$

Using the line of proof in [21], one can show that the persistence of this sequence is a $3\log(\frac{2}{c-1}+3)$-approximation of the persistence diagram of Rips filtration given below.

$$\mathcal{R}^0(V_0) \hookrightarrow \mathcal{R}^{\alpha c}(V_0) \cdots \hookrightarrow \mathcal{R}^{\alpha c^m}(V_0). \tag{4}$$

The blowup in scale by the factor of $\frac{3c-1}{c-1}$ results from the proof, which in practice causes some problems. We elaborate this further. To satisfy the approximation guarantee, one has to show that the persistence modules arising from Batch-collapsed Rips in sequence (3) and the standard Rips in sequence (4) interleave. In particular, this requires that we have well-defined simplicial maps from complexes in sequence (3) to those in sequence (4) and vice versa. The multiplicative factor $\frac{3c-1}{c-1}$ is needed to ensure that there is a well-defined simplicial map $\mathcal{R}^{\alpha c^k}(V_0) \to \mathcal{R}^{\alpha c^k \frac{3c-1}{c-1}}(V_k)$, as $\mathcal{R}^{\alpha c^k \frac{3c-1}{c-1}}(V_k)$ has to be sufficiently connected to include all the images of the simplices in the domain Rips complex $\mathcal{R}^{\alpha c^k}(V_0)$. The side effect of this is that the Batch-collapsed Rips complex has to be built at a much larger scale than the Rips complex, and it ends up with many unnecessary connections and thus more simplices in practice. This also causes a trade-off: Larger c reduces the over-connection but results in a worse approximation factor leading to a worse approximation quality. It is not clear how to set an increase rate that achieves both good approximation quality and efficiency for a specific data set.

We experimented Batch-collapsed Rips with Simpers on the same MotherChild model. Figure 2a, 2b and 2c show the persistence barcodes for different values of c. Observe that smaller values of c give better approximation. The barcode for $c = 1.3$ is the most similar among the three to that of Sparse Rips filtration in Figure 1b which is supposed to be more accurate theoretically. On the other hand, when c grows more than 1.8, it starts to lose some main bars in H_1 and noisy bars get longer in H_2. On the other hand, Figure 3 shows that, as c increases, both complex size and time cost decrease drastically. When $c = 2.0$, it only involves less than $216K$ simplices and takes time 9.4 seconds while, although $c = 1.3$ gives more accurate barcode, its size (22.5 million) and time (325s) approach those of the Sparse Rips. This demonstrates the dilemma that Batch-collapsed Rips faces in practice. We address this issue in our new approach SimBa. In particular, when $c \leq 2$, SimBa performs better than Batch-collapsed Rips for both size and time as shown in Figure 3 while capturing all main bars correctly as shown in Figure 2.

3 SimBa

To tame the over-connection in Batch-collapsed Rips, we replace the sequence in (3) with the sequence below where the parameter does not incur the extra factor $\frac{3c-1}{c-1}$:

$$\mathcal{B}^0(V_0) \to \mathcal{B}^{\alpha c}(V_1) \to \cdots \mathcal{B}^{\alpha c^m}(V_m) \tag{5}$$

The complexes $\mathcal{B}^{\alpha c^k}(V_k)$ are built on the same vertex sequence $\{V_k\}$ as in Batch-collapsed Rips, but the distances among the vertices of V_k are replaced with a *set distance* which allows us to avoid the over-connection. For two sets of points (clusters) $A, B \subset P$, we define their set distance as $d(A, B) = \min_{a\in A, b\in B} d(a, b)$. The sets that we consider are the pre-images of the vertices in V_k under the composition of projections π_i's, namely, for a vertex $v \in V_k$, we consider the set

$$B_v^k = \{p \in V_0 \mid \hat{\pi}_k(p) = v\} \quad \text{where} \quad \hat{\pi}_k : V_0 \to V_k \quad \text{is defined as} \quad \hat{\pi}_k = \pi_{k-1} \circ \cdots \circ \pi_0.$$

The complex $\mathcal{B}^{\alpha c^k}(V_k)$ is simply the clique complex induced by edges $\{(u, v) \in V_k \mid d(B_u^k, B_v^k) \leq \alpha c^k\}$. Observe that $d(u, v) \geq d(B_u^k, B_v^k)$ which ensures that the normal connec-

tion between u and v for a Rips filtration at the respective scale is not missed by considering the set distance while still avoiding the over-connection.

It turns out that each vertex map (nearest neighbor projection) $\pi_k : V_k \to V_{k+1}$ induces a simplicial map $h_k : \mathcal{B}^{\alpha c^k}(V_k) \to \mathcal{B}^{\alpha c^{k+1}}(V_{k+1})$. Instead of recomputing the simplicial complex each time, we generate elementary insertion and collapse operations incrementally for each h_k in three steps: (i) collapse each $v \in V_k \setminus V_{k+1}$ to its image $\pi_k(v)$ in V_{k+1} along with all incident simplices, (ii) insert new edges between two vertices in V_{k+1} if the distance between the two sets they represent are smaller than or equal to the current scale, and (iii) insert all new clique simplices containing new edges generated by (i) and (ii). Each h_k is processed in one batch, starting from a simplicial complex on vertices in V_k and resulting in a simplicial complex on vertices in V_{k+1}. The collapse and insertions of new simplices are exactly what Simpers need for computing the persistence.

3.1 Implementation Details

The advantage of SimBa (and Batch-collapsed Rips) over Sparse Rips filtrations is mainly due to the batched approach, which requires us to compute δ-nets of a point set for some δ repeatedly. Its advantage over the Batch-collapsed Rips is credited to the use of set distances. These computations require k-nearest neighbor search and fixed radius search for which efficient library like ANN [31] exists. We take advantage of this available software.

To compute a δ-net of a given point set (to obtain V_{k+1} from V_k), we randomly pick an untouched point, say p, use fixed-radius search to find all points in the ball of radius δ around p, map them to it, and mark them processed. We do this repeatedly until there is no untouched point left. We observe that this sub-sampling procedure can be carried out faster at early stage when δ is small because those points whose nearest neighbor distances are larger than the current δ can be taken directly into the net–they are all mapped to themselves and no other points are mapped to them. So, we maintain a list L of the points ordered by their nearest neighbor distances in increasing order and process them sequentially for δ-net computations. To compute the net points V_{k+1} from V_k, we carry out the full sub-sampling process only on the points in V_k that are already known to have nearest neighbor distances below δ and the new ones that qualify from L for increased δ. After δ becomes more than the largest nearest neighbor distance, we convert to the usual net computation.

Next, we describe an efficient implementation of the set distance computation, which being a basic operation in SimBa, speeds it up significantly. A straightforward implementation requires quadratic time, but we can make it more efficient in practice with the help of the ANN library. We use a hybrid strategy as follows. The sets B_u^k for vertices $u \in V_k$ are maintained by a union-find data structure. As vertices are collapsed while going from V_k to V_{k+1}, the sets of the collapsed vertices are merged to that of the target vertex. At early stages, when the number of sets (i.e, the size of V_k) is large and the diameter of each set is potentially small, we avoid computing set distances for all pairs. For each processing set B_u^k, we only need to find all the sets B_v^k whose distances to B_u^k are smaller than the current scale $\alpha' = \alpha c^k$. If so, we add an edge between u and v. To find all these nearby sets, we can do a fixed-radius search in $V_0 = P$ around each point in B_u^k within α' distance. For each point v returned by the search, we find v in the union-find data structure to identify its image $\hat{\pi}_k(v) \in V_k$. If the representing set of v is different from that of u, we add the edge $\hat{\pi}_k(u)\hat{\pi}_k(v)$.

Later when α' becomes large, it may not be efficient to continue this fixed-radius search, as the number of candidate points from P may be too large (can be n in the worst case). So we fall back on pairwise set distance computation. In particular, when the cardinality

of V_k becomes lower than a threshold, say 1/10 of the number of input points, we compute a pairwise set distance matrix (of size $|V_k| \times |V_k|$) among the surviving sets once and then keep updating the matrix with batch collapse thereafter. In particular, note that given sets A, B, and C, the set distance $d(A \cup B, C) = \min\{d(A, C), d(B, C)\}$.

3.2 SimBa on MotherChild model

We compare SimBa with other approaches on the same MotherChild model. Figure 2d, 2e and 2f show the persistence barcodes computed by SimBa with different values of c. We see that SimBa captures all the main 0, 1, 2-dimensional bars for all values of c in the range from 1.3 to 2.0 as opposed to Batch-collapsed Rips which fails to capture the main H_1 bars for $c > 1.8$. It tolerates larger range of c and thus is more robust than Batch-collapsed Rips. As expected, larger values of c produce less bars since there are less batches. So, in practice, we should choose smaller c, say less than 1.5. More importantly, as Figure 3 shows, the size and time for SimBa are also stable against different values of c, all less than $100K$ simplices and 10 seconds respectively for $c \leq 2$. These are less than those for Batch-collapsed Rips and significantly less than those for Sparse Rips: In particular, when $c = 1.3$, the maximum size for SimBa is $100K$, similar to when $c = 2$. However, for Batch-collapsed Rips, the maximum size is closer to that of SimBa when $c = 2$, and is 22.5 and 1.4 million when $c = 1.3$ and $c = 1.5$ respectively. This size difference becomes even more prominent for high dimensional data, as Table 1 shows. Although the approximation quality of SimBa is slightly worse than that of Sparse Rips based approaches, it does capture all the main bars, and more importantly, costs significantly less time. This advantage allows SimBa to process much larger high dimensional data sets which no previous approaches can handle, as we illustrate in section 5.

4 Approximation guarantee of SimBa

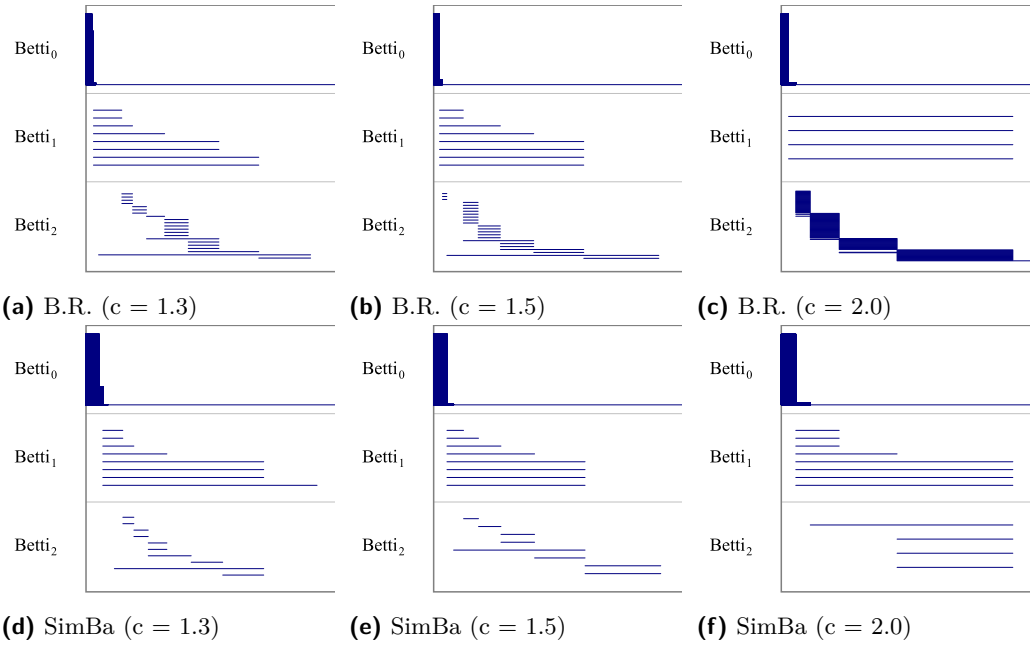
Recall that the simplicial complex $\mathcal{B}^\alpha(V_k)$ appearing in SimBa's filtration is defined as:

$$\mathcal{B}^\alpha(V_k) = \{\sigma \subset V_k \mid \forall u, v \in \sigma, d(B_u^k, B_v^k) \leq \alpha\}.$$

We prove that the persistence barcodes of SimBa's filtration in sequence (5) approximates those of the Rips filtration in (4) by showing that the persistence modules induced by these two sequences interleave.

First, observe that each vertex map π_k induces a well-defined simplicial map $h_k : \mathcal{B}^{\alpha c^k}(V_k) \to \mathcal{B}^{\alpha c^{k+1}}(V_{k+1})$. Indeed, for any edge $\{u, v\}$ in $\mathcal{B}^{\alpha c^k}(V_k)$, suppose $u' = \pi_k(u), v' = \pi_k(v)$, then $B_u^k \subset B_{u'}^{k+1}$ and $B_v^k \subset B_{v'}^{k+1}$. So we have $d(B_{u'}^{k+1}, B_{v'}^{k+1}) \leq d(B_u^k, B_v^k) \leq \alpha c^k < \alpha c^{k+1}$. Therefore $\{u', v'\}$ must be an edge in $\mathcal{B}^{\alpha c^{k+1}}(V_{k+1})$ as well. Since each complex in SimBa's filtration is a clique complex determined by edges, every simplex in $\mathcal{B}^{\alpha c^k}(V_k)$ has a well-defined image in $\mathcal{B}^{\alpha c^{k+1}}(V_{k+1})$. Thus, each h_k is well-defined.

Recall that the map $\hat{\pi}_k : V_0 \to V_{k+1}$ is defined as $\hat{\pi}_k(v) = \pi_k \circ \cdots \circ \pi_0(v)$, which tracks the image of any point in $V_0 = P$ during the batch collapse process. Observe that the vertex map $\hat{\pi}_k$ also induces a simplicial map $\hat{h}_k : \mathcal{R}^{\alpha c^k}(V_0) \to \mathcal{B}^{\alpha c^{k+1}}(V_{k+1})$: specifically, for any edge $(u, v) \in \mathcal{R}^{\alpha c^k}(V_0)$ with $d(u, v) \leq \alpha c^k$, it is easy to see that $d(B^k_{\hat{\pi}_k(u)}, B^k_{\hat{\pi}_k(v)}) \leq d(u, v) \leq \alpha c^k < \alpha c^{k+1}$, implying that $(\hat{\pi}_k(u), \hat{\pi}_k(v))$ is an edge in $\mathcal{B}^{\alpha c^{k+1}}(V_{k+1})$. The key observation is the following lemma.



(a) B.R. (c = 1.3) **(b)** B.R. (c = 1.5) **(c)** B.R. (c = 2.0)

(d) SimBa (c = 1.3) **(e)** SimBa (c = 1.5) **(f)** SimBa (c = 2.0)

Figure 2 Persistence barcodes computed by Batch-collapsed Rips plus Simpes (B.R.) and SimBa on the same MotherChild model. B.R. captures main bars for H_1 correctly for smaller values of c as shown in Figure (a) and (b) and loses some for $c = 2.0$ as shown in Figure (c), while SimBa works for $c = 2.0$.

▶ **Lemma 1.** *Each triangle in the diagram below commutes at homology level, where i_k and j_k are induced by inclusions, $h_{k,t} := h_{k+t-1} \circ \cdots \circ h_k$, $c > 1$, $t \geq \log_c(\frac{2}{c-1} + 3)$ and $t \in Z$.*

$$\begin{array}{ccc} \mathcal{R}^{\alpha c^k}(V_0) & \xhookrightarrow{i_k} & \mathcal{R}^{\alpha c^{k+t}}(V_0) \\ \downarrow \hat{h}_k & \nearrow_{j_k} & \downarrow \hat{h}_{k+t} \\ \mathcal{B}^{\alpha c^k}(V_k) & \xrightarrow{h_{k,t}} & \mathcal{B}^{\alpha c^{k+t}}(V_{k+t}) \end{array}$$

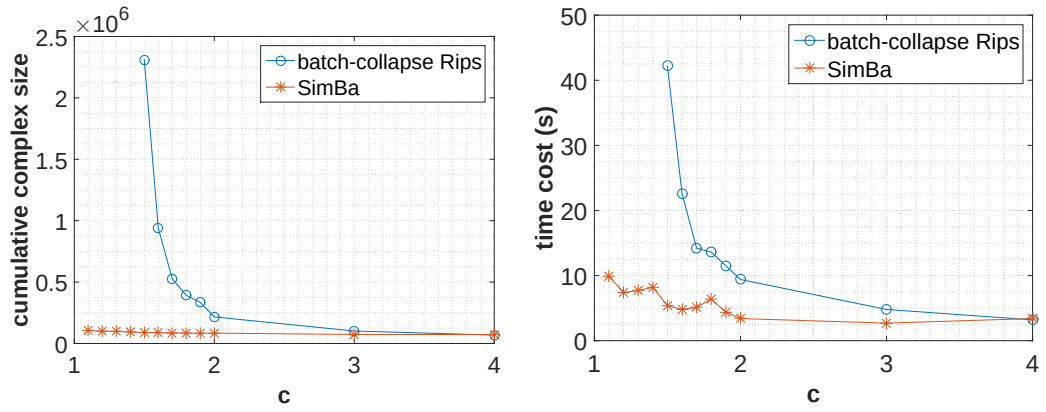
Proof. First, we prove that there is indeed an inclusion map $j_k : \mathcal{B}^{\alpha c^k}(V_k) \hookrightarrow \mathcal{R}^{\alpha c^{k+t}}(V_0)$. In particular, we show for each edge (u, v) in $\mathcal{B}^{\alpha c^k}(V_k)$, it's also an edge in $\mathcal{R}^{\alpha c^{k+t}}(V_0)$. Suppose the set distance $d(B_u^k, B_v^k)$ is achieved by the closest pair (u_0, v_0) between the two sets where $u_0 \in B_u^k, v_0 \in B_v^k$. Then $d(B_u^k, B_v^k) = d(u_0, v_0) \leq \alpha c^k$. Since V_{i+1} is an αc^{i+1}-net of V_i for each $i \in [0, k-1]$, it follows that $d(u, u_0) \leq \alpha c^k \sum_{i=0}^{k-1} \frac{1}{c^i} < \alpha c^k \frac{c}{c-1}$. Similar bound holds for $d(v, v_0)$. Thus:

$$d(u, v) \leq d(u, u_0) + d(v, v_0) + d(u_0, v_0) \leq \alpha c^k (\frac{2c}{c-1} + 1) = \alpha c^k (\frac{2}{c-1} + 3) \leq \alpha c^{k+t}.$$

Hence u, v is an edge in $\mathcal{R}^{\alpha c^{k+t}}(V_0)$.

Next, observe that the vertex map $\hat{\pi}_{k+t}$ restricted on the set of vertices V_k is exactly the same as the vertex map $\pi_{k,t} := \pi_{k+t-1} \circ \cdots \circ \pi_k$ (this vertex map induces the simplicial map $h_{k,t}$ in the diagram). Namely, for a vertex $u \in V_k \subseteq V_0$, $h_{k,t}(u) = \hat{h}_{k+t}(u)$. Thus $h_{k,t} = \hat{h}_{k+t} \circ j_k$. Hence the bottom triangle commutes both at the complex and the homology level.

We now consider the top triangle. Specifically, we prove that the map $j_k \circ \hat{h}_k$ is contiguous to the inclusion map i_k. Since two contiguous maps induce the same homomorphisms at the homology level, the top triangle commutes at the homology level.



(a) cumulative size **(b)** time cost

Figure 3 Complex size and time cost comparison between Batch-collapsed Rips and SimBa. SimBa beats Batch-collapsed Rips for both size and time when $c \leq 2$. For $c > 2$, the barcodes of both batch-based approaches become too coarse to be useful in practice.

Indeed, given a simplex $\sigma \in \mathcal{R}^{\alpha c^k}(V_0)$, we need to show that vertices from $i_k(\sigma) \cup j_k \circ \hat{h}_k(\sigma)$ span a simplex in $\mathcal{R}^{\alpha c^{k+t}}(V_0)$. Since both are Rips complexes and i_k and j_k are inclusion maps, we only need to prove that for any two vertices u and v from $\sigma \cup \hat{h}_k(\sigma)$, $d(u,v) \leq \alpha c^{k+t}$ (namely, (u,v) is an edge in $\mathcal{R}^{\alpha c^{k+t}}(V_0)$). If u and v are both from σ or both from $\hat{h}_k(\sigma)$, then $d(u,v) \leq \alpha c^{k+t}$ trivially. Otherwise, assume without loss of generality that $v \in \sigma$ and $u \in \hat{h}_k(\sigma)$, where $u = \hat{\pi}_k(u')$ for some $u' \in \sigma$. Since V_{i+1} is an is an αc^{i+1}-net of V_i for each $i \in [0, k-1]$, it follows that $d(u,u') \leq \alpha c^k \sum_{i=0}^{k-1} \frac{1}{c^i} < \alpha c^k \frac{c}{c-1}$. One then has

$$d(u,v) \quad \leq \quad d(u,u') + d(u',v) \leq \alpha c^k \frac{c}{c-1} + \alpha c^k = \alpha c^k \frac{2c-1}{c-1} < \alpha c^k (\frac{2}{c-1} + 3) \leq \alpha c^{k+t}.$$

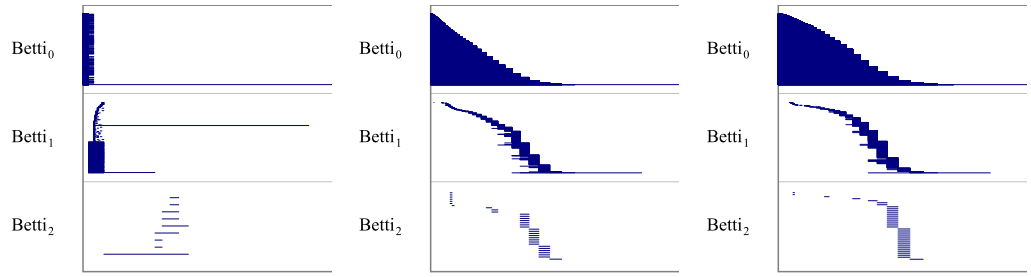
Thus i_k is contiguous to $j_k \circ \hat{h}_k$ and the lemma follows. ◀

The above result implies that the persistence modules induced by sequences (5) and (4) are weakly $\log c^t$-interleaved at the log-scale. Since $t \geq \log_c(\frac{2}{c-1} + 3)$, we have $c^t \geq \frac{2}{c-1} + 3$. By Theorem 4.3 of [12], we conclude with the following:

▶ **Theorem 2.** *The persistence diagram of the sequence (5) provides a* $3\log(\frac{2}{c-1} + 3)$*-approximation of the persistence diagram of the sequence (4) at the log-scale for* $c > 1$.

5 Experiments

In this section, we report some experimental results of SimBa on large high dimensional data sets from other fields such as image processing, machine learning, and computational biology. For most of the data sets, previous approaches are not efficient enough to finish processing. They either ran out of memory ('∞' in size) or ran more than one day ('∞' in time). Table 1 at the end of this section provides the cumulative size and time cost for all four approaches mentioned in this paper. All approaches are implemented in C++. Note that we only compute persistences up to dimension 2 (which means we build simplicial complexes up to dimension 3). For Sparse Rips with GUDHI and Sparse Rips with Simpers, we choose parameter $\varepsilon = 0.8$ which gives the best performance while not sacrificing much of the approximation quality. For Batch-collapsed Rips with Simpers, we choose $c = 1.5$



(a) Klein Bottle in $\mathbb{R}^4$ **(b)** Primary Circle in $\mathbb{R}^{25}$ **(c)** Primary Circle in $\mathbb{R}^{49}$

Figure 4 Original persistence barcodes computed by SimBa on data sets with ground truth.

which appears to reach a good trade-off between efficiency and quality. For SimBa, we choose $c = 1.1$ which in practice appears to have best quality – note that the choice of c does not seem to change the empirical efficiency much as Figure 3 illustrates. All experiments were run on a 64-bit Windows machine with a 3.50GHz Intel processor and 16GB RAM.

Data with ground truth

We first test with two data sets whose ground truth persistences are known. They help demonstrate that SimBa works properly and efficiently in practice. All persistence barcodes shown in Figure 4 are original and not cleaned up.

We first consider a uniform sample of 22500 points from a Klein bottle in $\mathbb{R}^4$, and use SimBa to compute its barcode which is shown in Figure 4a. There are two main bars for H_1 and one for H_2 as expected.

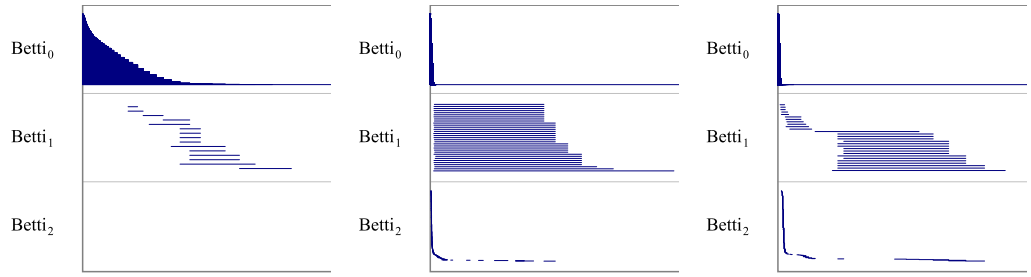
Next, we consider the primary circle of natural image data in [1], which has 15000 points. Each point is a 5×5 or 7×7 image patch, thus considered as a point in $\mathbb{R}^{25}$ or $\mathbb{R}^{49}$. From Figures 4b and 4c, we can see the primary circle bar for H_1 for data both in $\mathbb{R}^{25}$ and $\mathbb{R}^{49}$. All short bars for H_2 persist for only one batch step and thus can be regarded as noise.

Data without ground truth

Next, we provide some experiments on the data sets whose ground truth persistences are not known. We used SimBa to compute their persistences and found some relatively long bars which are likely to be features and may worth further investigation by domain experts. The persistence barcodes shown in Figure 5 and 6d are denoised for H_1. The rest of Figure 6 are original.

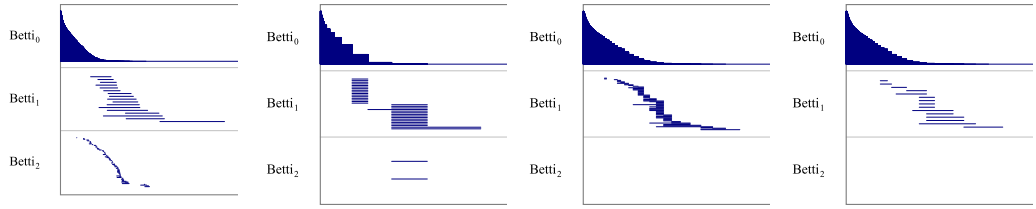
We first take the Gesture Phase Segmentation data set [30] from UCI machine learning repository [28]. This data set was used in [29]. It comprises of features extracted from 7 videos with people gesticulating. Each video is represented by a raw file that contains the positions of hands, wrists, head, and spine of the user in each frame. We took the raw file from video A1 of 1747 frames. Since there are six sensors each with x, y, z coordinates, we have in total 1747 points in $\mathbb{R}^{18}$. There are five gesture phases in the videos: rest, preparation, stroke, hold, and retraction. Indeed, there are five long bars for H_0 in 5a (although they overlap and do not stand out in the picture), which seems to match the five clusters of different phases. We see some long bars for H_1, which could be created due to periodic patterns in these gesture movements.

Another data set is the Survivin protein data from [27]. There are totally 252996 protein conformations and each conformation is considered as a point in $\mathbb{R}^{150}$. We used PCA to



(a) Gesture Phase data in $\mathbb{R}^{18}$ **(b)** Survivin data in $\mathbb{R}^3$ **(c)** Survivin data in $\mathbb{R}^{150}$

Figure 5 Denoised persistence barcodes computed by SimBa on data sets without ground truth.



(a) S.R.+GUDHI **(b)** B.R.+Simpers **(c)** SimBa **(d)** SimBa (denoised)

Figure 6 Persistence barcodes computed by different approaches on Gesture Phase Segmentation data in $\mathbb{R}^{18}$.

Table 1 cumulative size and time cost.

				S.R.+GUDHI		**S.R.+Simpers**		**B.R.+Simpers**		**SimBa**	
Data	n	D	d	size	time(s)	size	time(s)	size	time(s)	size	time(s)
Mother	23075	3	2	$43.5 \cdot 10^6$	350	$43.5 \cdot 10^6$	463.7	$2.3 \cdot 10^6$	42.3	104701	8.8
KlBt	22500	4	2	$20.9 \cdot 10^6$	205.3	$20.9 \cdot 10^6$	303.5	440049	8	78064	6.6
PrCi25	15000	25	?	∞	–	∞	–	–	∞	$4.8 \cdot 10^6$	216
PrCi49	15000	49	?	∞	–	∞	–	–	∞	$10.2 \cdot 10^6$	585
GePh	1747	18	?	$45.6 \cdot 10^6$	282.5	$45.6 \cdot 10^6$	432.8	$1.4 \cdot 10^6$	29	7145	0.83
Sur3	252996	3	?	∞	–	∞	–	$15.7 \cdot 10^6$	1056.4	915110	1079.6
Sur150	252996	150	?	∞	–	∞	–	–	∞	$3.1 \cdot 10^6$	5089.7

reduce the data dimension to 3. We ran SimBa on both data sets and show the barcodes in Figure 5c and 5b. We can see that there are some long bars for H_1.

Performance results

We provide the performance results for all data sets mentioned in Table 1, which includes cumulative size and time cost of each approach. The time is obtained by adding the time to construct the complexes and the time to compute persistence. **S.R.+GUDHI**, **S.R.+Simpers**, **B.R.+Simpers** and **SimBa** stand for Sparse Rips plus GUDHI, Sparse Rips plus Simpers, Batch-collapsed Rips plus Simpers, and SimBa respectively. **Mother**, **KlBt**, **PrCi25**, **PrCi49**, **GePh**, **Sur3** and **Sur150** stand for MotherChild model, Klein Bottle, Primary Circle in $\mathbb{R}^{25}$, Primary Circle in $\mathbb{R}^{25}$, Gesture Phase Segmentation data, Survivin protein data in $\mathbb{R}^3$ and in $\mathbb{R}^{150}$ respectively. Each data set has size n, ambient dimension D, and intrinsic dimension d. The symbol ∞ means that the program either ran out of memory or did not finish after a day. From the table, we can see that SimBa

out-performed the other three approaches significantly. Notice that for those larger cases of SimBa, the nearest neighbor search operations (ANN) usually take most of time and become the bottleneck. This is why **Sur150** costs much more time than **PrCi49** while its cumulative size is smaller. It would be an interesting future work to make nearest neighbor search more efficient so that SimBa performs better even for such cases.

References

1 H. Adams and G. Carlsson. On the nonlinear statistics of range image patches. *SIAM J. Img. Sci.*, 2(1):110–117, 2009. doi:10.1137/070711669.

2 U. Bauer, M. Kerber, and J. Reininghaus. *Topological Methods in Data Analysis and Visualization III: Theory, Algorithms, and Applications*, chapter Clear and Compress: Computing Persistent Homology in Chunks, pages 103–117. Springer International Publishing, Cham, 2014. doi:10.1007/978-3-319-04099-8_7.

3 U. Bauer, M. Kerber, J. Reininghaus, and H. Wagner. *Mathematical Software – ICMS 2014: 4th International Congress, Seoul, South Korea, August 5-9, 2014. Proceedings*, chapter PHAT – Persistent Homology Algorithms Toolbox, pages 137–143. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. Project URL: https://bitbucket.org/phat-code/phat.

4 J.-D. Boissonnat, T. K. Dey, and C. Maria. *Algorithms – ESA 2013: 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, chapter The Compressed Annotation Matrix: An Efficient Data Structure for Computing Persistent Cohomology, pages 695–706. Springer, Berlin, Heidelberg, 2013. doi:10.1007/978-3-642-40450-4_59.

5 J.-D. Boissonnat and C. Maria. *Algorithms – ESA 2012: 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*, chapter The Simplex Tree: An Efficient Data Structure for General Simplicial Complexes, pages 731–742. Springer, 2012.

6 M. Buchet, F. Chazal, S. Y. Oudot, and D. R. Sheehy. Efficient and robust persistent homology for measures. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 168–180, 2015. doi:10.1137/1.9781611973730.13.

7 D. Burghelea and T. K. Dey. Topological persistence for circle-valued maps. *Discrete Comput. Geom.*, 50:69–98, 2013.

8 G. Carlsson. Topology and data. *Bull. Amer. Math. Soc.*, 46:255–308, 2009.

9 G. Carlsson and V. de Silva. Zigzag persistence. *Foundations of computational mathematics*, 10(4):367–405, 2010.

10 G. Carlsson and A. Zomorodian. The theory of multidimensional persistence. *Discrete & Computational Geometry*, 42(1):71–93, 2009. doi:10.1007/s00454-009-9176-0.

11 N. J. Cavanna, M. Jahanseir, and D. R. Sheehy. A geometric perspective on sparse filtrations. In *Canadian Conf. Comput. Geom. (CCCG)*, 2015. URL: http://dblp.uni-trier.de/db/conf/cccg/cccg2015.html#CavannaJS15.

12 F. Chazal, D. Cohen-Steiner, M. Glisse, L. J. Guibas, and S. Y. Oudot. Proximity of persistence modules and their diagrams. In *Proceedings of the Twenty-fifth Annual Symposium on Computational Geometry*, SCG'09, pages 237–246, New York, NY, USA, 2009. ACM. doi:10.1145/1542362.1542407.

13 F. Chazal, D. Cohen-Steiner, L. Guibas, F. Mémoli, and S. Y. Oudot. Gromov-Hausdorff stable signatures for shapes using persistence. In *Proc. of SGP*, 2009.

14 F. Chazal, V. de Silva, M. Glisse, and S. Oudot. The structure and stability of persistence modules. *CoRR*, abs/1207.3674, 2012.

15 F. Chazal, L. J. Guibas, S. Y. Oudot, and P. Skraba. Persistence-based clustering in Riemannian manifolds. In *Proc. 27th Annu. ACM Sympos. Comput. Geom.*, pages 97–106, 2011.

16 C. Chen and M. Kerber. An output-sensitive algorithm for persistent homology. *Comput. Geom. Theory Appl.*, 46(4):435–447, May 2013. doi:10.1016/j.comgeo.2012.02.010.

17 D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

18 D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Extending persistence using Poincaré and Lefschetz duality. *Foundations of Computational Mathematics*, 9(1):79–103, 2009.

19 D. Cohen-Steiner, H. Edelsbrunner, and D. Morozov. Vines and vineyards by updating persistence in linear time. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 119–126. ACM, 2006.

20 V. de Silva, D. Morozov, and M. Vejdemo-Johansson. Persistent cohomology and circular coordinates. *Discrete Comput. Geom.*, 45(4):737–759, 2011.

21 T. K. Dey, F. Fan, and Y. Wang. Computing topological persistence for simplicial maps. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, SOCG'14, pages 345–354. ACM, 2014. doi:10.1145/2582112.2582165.

22 T. K. Dey, K. Li, C. Luo, P. Ranjan, I. Safa, and Y. Wang. Persistent heat signature for pose-oblivious matching of incomplete models. *Comput. Graph. Forum. (special issue from* Sympos. Geom. Process.*)*, 29(5):1545–1554, 2010.

23 Dmitriy Morozov. Dionysus Software. http://mrzv.org/software/dionysus/, 2012.

24 H. Edelsbrunner and J. Harer. *Computational Topology – an Introduction.* American Mathematical Society, 2010.

25 H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete Comput. Geom.*, 28:511–533, 2002.

26 P. Frosini. A distance for similarity classes of submanifolds of a euclidean space. *Bulletin of the Australian Mathematical Society*, 42(3):407–416, 1990.

27 W. Harvey, I.-H. Park, O. Rübel, V. Pascucci, P.-T. Bremer, C. Li, and Y. Wang. A collaborative visual analytics suite for protein folding research. *Journal of Molecular Graphics and Modelling*, 53:59–71, 2014. doi:10.1016/j.jmgm.2014.06.003.

28 M. Lichman. UCI machine learning repository, 2013. Project URL: http://archive.ics.uci.edu/ml.

29 R. C. B. Madeo, S. M. Peres, and C. A. de M. Lima. Gesture phase segmentation using support vector machines. *Expert Systems with Applications*, 56:100–115, 2016. doi:10.1016/j.eswa.2016.02.021.

30 R. C. B. Madeo, P. K. Wagner, and S. M. Peres. Gesture Phase Segmentation Data Set, 2014. Project URL: http://archive.ics.uci.edu/ml/datasets/Gesture+Phase+Segmentation.

31 D. M. Mount and S. Arya. ANN: A Library for Approximate Nearest Neighbor Searching, 2010. Project URL: https://www.cs.umd.edu/~mount/ANN/.

32 James R. Munkres. *Elements of Algebraic Topology.* Addison-Wesley, 1993.

33 J. Reininghaus, S. Huber, U. Bauer, and R. Kwitt. A stable multi-scale kernel for topological machine learning. In *Proc. IEEE Conf. Comp. Vision & Pat. Rec. (CVPR)*, pages 4741–4748, 2015.

34 V. Robins. Towards computing homology from finite approximations. *Topology Proceedings*, 24(1):503–532, 1999.

35 D. R. Sheehy. Linear-size approximations to the vietoris-rips filtration. In *Proceedings of the Twenty-eighth Annual Symposium on Computational Geometry*, SoCG'12, pages 239–248. ACM, 2012. doi:10.1145/2261250.2261286.

36 Simpers Software, 2015. Project URL: http://web.cse.ohio-state.edu/~tamaldey/SimPers/Simpers.html.

37 G. Singh, F. Mémoli, T. Ishkhanov, G. Sapiro, G. Carlsson, and D. L Ringach. Topological analysis of population activity in visual cortex. *Journal of vision*, 8(8):11, 2008.

38 The GUDHI Project. GUDHI user and reference manual, 2015. Project URL: http://gudhi.gforge.inria.fr/doc/latest/.
39 A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete Comput. Geom.*, 33(2):249–274, 2005.