Fast Construction of the Vietoris-Rips Complex

Afra Zomorodian*

(To appear in Computer & Graphics)

Abstract

The Vietoris-Rips complex characterizes the topology of a point set. This complex is popular in topological data analysis as its construction extends easily to higher dimensions. We formulate a two-phase approach for its construction that separates geometry from topology. We survey methods for the first phase, give three algorithms for the second phase, implement all algorithms, and present experimental results. Our software can also be used for constructing any clique complex, such as the weak witness complex.

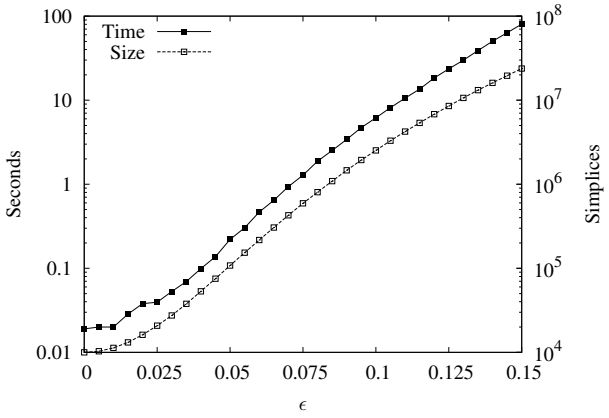


Figure 1: Construction time and complex size of the 3-dimensional Vietoris-Rips complex at scale ϵ for a set of 10,000 uniformly sampled points from the unit 2-sphere in $\mathbb{R}^3$. The results are averaged over 20 independent runs per plot point.

1 Introduction

In this paper, we present fast algorithms for constructing the filtered Vietoris-Rips complex of a point set. Our software can compute arbitrary dimensional complexes for point sets in arbitrary dimensions, and may be applied toward constructing other clique complexes, such as the weak witness complex. Figure 1 shows the performance of our fastest algorithm to inspire interest. On this point set, we compute a complex consisting of 24M simplices in about 80 seconds at scale $\epsilon = 0.15$. All our timings are done on a 64-bit GNU/Linux machine with two dual-core 3 GHz Xeon processors, although our software is not threaded and uses only one core.

1.1 Motivation

Scientific data, whether acquired or simulated, may be modeled as *point cloud data*, finite sets of points embedded in metric spaces. Analysis assumes that data has structure: It is sampled from some underlying geometric space. Within computational topology, the emerging area of *topological data analysis* focuses on the recovery of the lost topology of this underlying space. In topological analysis, we generally follow a two step process. In the first step, we approximate the underlying structure of the point set using a combinatorial structure, such as a simplicial or cubical complex. In the second step, we utilize techniques from algebraic topology to compute topological invariants of these structures. One popular invariant is persistent homology [14, 39] which analyzes the relationship between structures at different scales.

There are a number of methods for completing the first step of topological analysis. We may partition these methods roughly into geometric and algebraic techniques. Geometric methods include the alpha complex [15], its conformal variant [6], and the flow complex [18], to name a few. When the data is embedded in $\mathbb{R}^2$ or $\mathbb{R}^3$, geometric methods are ideal as they are fast and produce small embedded complexes. Unfortunately, these methods depend on the Delaunay complex [10]. Currently, we do not have robust software for computing the Delaunay complex in dimensions higher than three, although progress is being made [3].

The classic algebraic method is the Čech complex [20] whose construction is infeasible in practice. For this reason, this complex is often approximated by the Vietoris-Rips complex or VR complex for short, the focus of this paper [35]. The VR complex is currently the only practical complex for analyzing datasets in higher dimensions. We have used it, for instance, to compute shape descriptors based on curvature [8] and to characterize the local structure of natural images [5]. Another popular method is the family of witness complexes [12] which approximate topology. As we will see, the variant of this complex often used in practice, the weak witness complex, is related to the VR complex, so our work is immediately applicable to its construction.

*Department of Computer Science, Dartmouth College, Hanover, New Hampshire

1.2 Prior Work

Due to its simplicity, the VR complex has been computed often using ad-hoc algorithms in the past, including in our prior work [5]. These implementations were sufficient because the datasets were small or sampled to be small, generally a few hundred points in size. Also, only low-dimensional complexes were built. Currently, public implementations are available in PLEX [30], a MATLAB library, and its java descendant JPlex [32]. We compare our implementation to JPlex which may be viewed as the state of the art. We did not have access to the source code of JPlex, but corresponded with its authors regarding the algorithm used. JPlex takes an inductive approach as does our first algorithm. We know of no other publicly available software for computing the VR complex.

1.3 Contributions

As detailed in Section 3, our approach is to separate the construction of the filtered VR complex into two phases. In the first phase, we compute a neighborhood graph on the input points based on the ambient metric. We review several methods for computing this graph, including exact, approximate, and randomized nearest neighbors, as well as landmarking. In the second phase, we expand the graph to include higher-dimensional simplices, an operation often called the Vietoris-Rips expansion. In Section 4, we describe three algorithms for expansion. We begin with an inductive algorithm that is similar to algorithms that have been implemented in the past. We then introduce two novel algorithms: an incremental algorithm and one employing a dramatically different approach: enumeration of maximal cliques. We have implemented all of our algorithms and examine their performance in practice in Section 5, including comparisons to existing software, and computing clique complexes of graphs.

Our work represents the first systematic examination of all the stages of the construction of the Vietoris-Rips complex. It also lays the foundation of our current work in computing smaller representations for point cloud data [38]. As Figure 1 demonstrates, our software is currently the fastest available, constructing complexes with millions of simplices in arbitrary dimensions in seconds.

2 Background

In this section, we begin a quick review of simplicial complexes, the objects that we build. We then define the Vietoris-Rips complex and an associated weight function on its simplices. For an accessible introduction to computational topology, we recommend a recent survey [37].

A *simplicial complex* is a set K of finite sets such that if $\sigma \in K$ and $\tau \subseteq \sigma$, then $\tau \in K$. For every $\tau \subseteq \sigma \in K$, we say τ is a *face* of σ, its *coface*. The (-1)*-simplex* $\emptyset$ is a face of any simplex. A simplex is *maximal* if it has no proper coface in K. If $\sigma \in K$ has cardinality $|\sigma| = k+1$, we call σ a k*-simplex* of *dimension* k, $\dim(\sigma) = k$. This name stems from our ability to realize a k-simplex geometrically as a k-dimensional subspace of $\mathbb{R}^d$, $d \geq k$, namely, the convex hull of $k+1$ affinely-independent points. Given this view, a k-simplex is called a *vertex*, an *edge*, a *triangle*, or a *tetrahedron* for $0 \leq k \leq 3$, respectively. A simplicial complex may be embedded in Euclidean space as the union of its geometrically realized simplices such that they only intersect along shared faces.

A *subcomplex* is a subset $L \subseteq K$ that is also a simplicial complex. An important subcomplex is the k*-skeleton* consisting of simplices in K of dimension less than or equal to k. A *filtration* of a complex K is a sequence of nested subcomplexes $\emptyset = K_0 \subseteq K_1 \subseteq \ldots \subseteq K_m = K$. A complex with a filtration is a *filtered complex*.

Suppose we are given a finite set of d-dimensional points $S \subseteq \mathbb{R}^d$, such as the set of 13 points in the plane in Figure 2(a). The *Vietoris-Rips complex* (*VR complex*) $\mathcal{V}_\epsilon(S)$ of S at scale ϵ is

$$\mathcal{V}_\epsilon(S) = \{\sigma \subseteq S \mid \mathrm{d}(u,v) \leq \epsilon, \forall u \neq v \in \sigma\}, \quad (1)$$

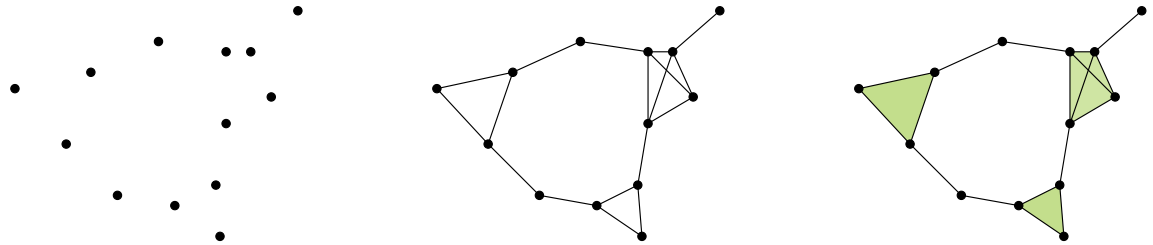
where d is the Euclidean metric [19, 35]. In other words, each simplex σ in $\mathcal{V}_\epsilon(S)$ has vertices that are pairwise within distance ϵ. In practice, we compute the VR complex for some maximum scale $\hat{\epsilon} \in \mathbb{R}$ and then extract the complex at any lower scale $\epsilon \leq \hat{\epsilon}$. To do so, we define a *weight* function $\omega \colon \mathcal{V}_{\hat{\epsilon}}(S) \to \mathbb{R}$ over the simplices. Given $\sigma \in \mathcal{V}_{\hat{\epsilon}}(S)$, we define:

$$\omega(\sigma) = \begin{cases} 0, & \dim(\sigma) \leq 0, \\ \mathrm{d}(u,v), & \sigma = \{u,v\} \\ \max\limits_{\tau \subset \sigma} \omega(\tau), & \text{otherwise.} \end{cases} \quad (2)$$

That is, the weight function is the minimum ϵ at which a simplex σ enters the VR complex, equal to the maximum of the weights (lengths) of all its edges. Since σ's faces share its edges, above we equivalently define the weight of a simplex to be the maximum of the weights of all its faces. This inductive definition translates directly into a recursive algorithm in Section 4.4. We may now sort the simplices according to their weights, extracting the VR complex for any $\epsilon \leq \hat{\epsilon}$ as a prefix of this ordering. That is, we extract the filtration of the $\mathcal{V}_{\hat{\epsilon}}(S)$. This is a particular representation of a filtered complex that is very useful in practice.

Definition 1 (weight-filtered complex) *A* weight-filtered complex *is the tuple* (K, f)*, where* K *is a simplicial complex and* $f \colon K \to \mathbb{R}$ *is a discrete* weight function *such that* $K_\epsilon = \{\sigma \mid f(\sigma) \leq \epsilon\}$ *yields a filtration.*

Our goal in this paper is to compute the weight-filtered VR complex $(\mathcal{V}_{\hat{\epsilon}}(S), \omega)$. This filtered complex is the required input to the persistence algorithm [39] for topological analysis.



(a) S (b) $G_\epsilon(S)$ (c) $\mathcal{V}_\epsilon(S)$

Figure 2: Construction of the Vietoris-Rips complex. Our input (a) is a set of points S. Section 3.2 describes the first phase, the geometric process of going from (a) to a neighborhood graph (b). Section 4 describes the second phase, the combinatorial process of expanding from the graph (b) to the Vietoris-Rips complex (c).

3 Approach

In this section, we describe our approach to constructing the VR complex. We divide this construction into two phases: one geometric and one topological, as illustrated in Figure 2. We then survey algorithms for completing the first phase, the construction of the neighborhood graph.

3.1 Two-Phase Construction

Consider the definition of the VR complex in Equation (1). Points within distance ϵ create edges in the VR complex, and together with those edges constitute the 1-skeleton of the complex, which is equivalently a *graph*. We name this graph for ease and separate its construction.

Definition 2 (neighborhood graph) *A* neighborhood graph *is (G, w), where $G = (V, E)$ is an undirected graph, and $w\colon E \to \mathbb{R}$ is its* weight function*, defined on its edges.*

The 1-skeleton of the VR complex is a neighborhood graph.

Definition 3 (VR neighborhood graph) *Given $S \subseteq \mathbb{R}^d$ and scale $\epsilon \in \mathbb{R}$, the* VR neighborhood graph *is a neighborhood graph $(G_\epsilon(S), w)$, where $G_\epsilon(S) = (S, E_\epsilon(S))$ and*

$$E_\epsilon(S) = \{\{u, v\} \mid \mathrm{d}(u, v) \leq \epsilon, u \neq v \in S\}, \quad (3)$$

$$w(\{u, v\}) = \mathrm{d}(u, v),\ \forall \{u, v\} \in E_\epsilon(S). \quad (4)$$

Figure 2(b) shows a VR neighborhood graph with 18 edges, where w is the length of the edges. We may expand any neighborhood graph up to a weight-filtered complex.

Definition 4 (Vietoris-Rips expansion) *Given a neighborhood graph (G, w), we compute the weight-filtered Vietoris-Rips complex $(\mathcal{V}(G), \omega)$ via* Vietoris-Rips expansion*: If all the edges of a simplex σ are in G, then σ is in $\mathcal{V}(G)$. For $G = (V, E)$, we have*

$$\mathcal{V}(G) = V \cup E \cup \left\{\sigma \mid \tbinom{\sigma}{2} \subseteq E\right\}, \quad (5)$$

For $\sigma \in \mathcal{V}(G)$,

$$\omega(\sigma) = \begin{cases} 0, & \sigma = \{v\}, v \in V, \\ w(\{u, v\}), & \sigma = \{u, v\} \in E \\ \max\limits_{\tau \subset \sigma} \omega(\tau), & \textit{otherwise.} \end{cases} \quad (6)$$

Note that the definition of $\mathcal{V}(G)$ is completely combinatorial and makes no reference to the metric on the embedding space. The definition of ω is inductive as before. Figure 2(c) shows the result of VR expansion on a neighborhood graph, adding two triangles and a tetrahedron along with its four triangular faces.

This approach gives us a two-phase scheme for constructing the $\mathcal{V}_{\hat{\epsilon}}(S)$ for input $S \subseteq \mathbb{R}^d$ and maximum scale $\hat{\epsilon} \in \mathbb{R}$.

1. Compute $(G_{\hat{\epsilon}}(S), w)$ by Definition 3. This is the VR neighborhood graph and its construction constitutes a *geometric* phase as it requires the distance metric. We survey methods for completing this phase next.

2. Compute $(\mathcal{V}(G_{\hat{\epsilon}}(S)), \omega)$ by Definition 4. This is the weight-filtered VR complex computed via expansion and its construction is purely combinatorial or *topological*. We give three algorithms for this phase in Section 4.

Now $\mathcal{V}_{\hat{\epsilon}}(S) = \mathcal{V}(G_{\hat{\epsilon}}(S))$, so we are done. This characterization exhibits the beauty of the VR complex in its separation of geometry from topology. It also liberates us from Euclidean geometry, defining a family of complexes that are based on VR expansion from any type of neighborhood graphs.

3.2 Computing a Neighborhood Graph

We now focus on completing the first phase: constructing the neighborhood graph. We review eight algorithms from two main approaches: nearest neighbors and landmarking. All the algorithms have the following interface.

- Input: finite set $S \subseteq \mathbb{R}^d$ and scale $\hat{\epsilon} \in \mathbb{R}$
- Output: Neighborhood graph (G, w)

The first approach is a classic problem in computational geometry with a rich literature: the *nearest neighbor problem*: Preprocess S so that given any query q, the closest $p \in S$ can be reported quickly [26]. This problem has a variety of geometric applications, such as pattern classification [13] and quantization in image compression [21]. We need neighbors of all points in S within distance ϵ, so we have a combination of two variants of this problem: the *ϵ-nearest neighbor problem* and the *all nearest neighbors problem*. There are a number of approaches.

Exact We may use the *all-pairs* brute-force algorithm to report pairs in S within distance ϵ in $O(n^2)$ time without any preprocessing. A simple but empirically faster approach is to *scanning*. We project the points onto a random vector in $\mathbb{R}^d$ and sort the points along that vector, using the projection distance as a lower-bound on the actual distance. In practice, we use all-pairs and scanning to verify the more sophisticated methods. Friedman et al. combine *kd-trees* with a search algorithm to achieve an $O(\log n)$ query time with $O(n)$ space in the expected case [17]. Arya et al. describe the *balanced box-decomposition tree (BBD-tree)* and *priority search* to develop a method with an error parameter that may be set to zero for an exact algorithm [2]. Any exact solution hides exponential dependence on the embedding dimension in its constants as it suffers from the so-called *curse of dimensionality*. In particular, in any fixed dimension greater than 2, we cannot achieve linear space and logarithmic query time [2].

Approximate Since we are approximating the underlying space of S to recover its topology, we may argue that computing an exact solution is not necessary if the computation is intractable. Therefore, we may use approximate methods if they are faster. It should be clear, however, that approximating the neighborhood graph will not yield the exact VR complex but an approximation of the complex. Given error $\delta > 0$, a point $p \in S$ is a *$(1+\delta)$-approximate neighbor* of a query q if $d(p,q) \leq (1+\delta) \cdot d(p',q)$, where p' is the true nearest neighbor to q. The algorithm of Arya et al. above is an asymptotically optimal solution to this problem. We may also replace the BBD-tree with the kd-tree or adapt Friedman et al.'s search scheme for approximate solutions with either tree.

Randomized For high dimensions (greater than 20), space partitioning schemes such as the kd-tree or BBD-tree offer little improvement over brute-force search [2]. Instead, we may employ a probabilistic approach. The *$(1-\gamma)$-nearest neighbor problem* reports the nearest neighbors with probability at least $1-\gamma$. One solution to this problem is the dimensionality reduction scheme of *locality sensitive hashing (LSH)*. Andoni and Indyk give an algorithm for Euclidean L_2 spaces from this family [1].

Landmarking Yet another approach to approximating the neighborhood graph is by constructing it on a subset of the input points. We begin by selecting a set of *landmarks* $L \subseteq S$. The remaining points, $W = S - L$, are the set of potential *witnesses*. The *witness graph* at scale $\epsilon \in \mathbb{R}$ is the graph $G = (L, E_{\epsilon,L})$, where $\{l_1, l_2\} \in E_{\epsilon,L}$ if there exists a *witness* $w \in W$ that is closer to l_i than any other landmark, and $\mathrm{d}(w, l_i) \leq \epsilon$ for $i = 1, 2$. The weight function is $\mathrm{d}(u, v)$ as before. This approach relaxes the Delaunay test that requires an *equidistant* point from two vertices for an edge to exist [10]. The family of complexes based on this notion are called *witness complexes* [12]. Searching for witnesses is easy as we only have to compute an $|S| \times |L|$ distance matrix, and by design, $|L| \ll |S|$. Once we have the graph, we may VR expand to construct the *weak witness complex*. In other words, our algorithms for VR expansion in the next section apply toward constructing this complex.

We now have several algorithms for completing the first phase of our construction: computing the neighborhood graph. We will look at the performance of most of these algorithms in Section 5.2. We do not implement landmarking, however, due to our focus on the VR complex.

4 VR Expansion Algorithms

In this section, we focus on the second phase of constructing the VR complex: Vietoris-Rips expansion of the neighborhood graph, as shown in Figure 2(c). Recall that this phase is purely combinatorial. We assume that our input is a neighborhood graph constructed by one of the methods in the previous section. Our task is to augment this graph with higher-dimensional simplices to complete the VR complex. We give three algorithms for accomplishing this task, ending the section with an algorithm to compute the associated weight function.

We generally characterize the topology of space up to *homology*. In this formalism, a space only has trivial invariants in dimensions higher than its embedding dimension [20]. Also, if we need to compute i-dimensional homology, we require only the $(i+1)$-skeleton. Therefore, we usually compute the k-skeleton of for some $k \leq d$. The VR expansion algorithms have the following interface.

- Input: Neighborhood graph (G, w) and maximum dimension k.
- Output: k-skeleton of weight-filtered VR complex $(\mathcal{V}(G), \omega)$.

In the rest of this section, we assume $k > 1$ since G itself is the 1-skeleton.

4.1 Inductive Algorithm

For our first algorithm, we induct on dimension. We may construct any simplicial complex inductively by gluing

simplices of dimension i to simplices of dimension $i-1$. Throughout this section, we assume we have a total ordering on the vertices in the neighborhood graph (an arbitrary ordering will do.) We begin with a simple utility function that finds all neighbors of vertex u within G that precede it in the given ordering. The function uses dot notation from C++ to access the elements of G.

LOWER-NBRS(G, u)
1 **return** $\{v \in G.V \mid u > v,\ \{u, v\} \in G.E\}$

We use this function in our first algorithm below.

INDUCTIVE-VR(G, k)
1 $\mathcal{V} \leftarrow G.V \cup G.E$
2 **for** $i \leftarrow 1$ **to** k
3 **do foreach** i-simplex $\tau \in \mathcal{V}$
4 **do** $N \leftarrow \bigcap_{u \in \tau}$ LOWER-NBRS(G, u)
5 **foreach** $v \in N$
6 **do** $\mathcal{V} \leftarrow \mathcal{V} \cup \{\tau \cup \{v\}\}$
7 **return** $\mathcal{V}$

The algorithm is reminiscent of the inductive definition of the CW complex [20]. We describe the algorithm in the proof of the following theorem.

Theorem 1 INDUCTIVE-VR(G, k) *computes the k-skeleton of $\mathcal{V}(G)$.*

Proof: We begin by initializing the $\mathcal{V}$ on line 1 with the input graph, so it now contains its 1-skeleton. We establish the invariant that $\mathcal{V}$ must contain the i-skeleton at the beginning of the ith iteration of the loop on line 2. This invariant clearly holds for $i = 1$. We maintain this invariant by adding co-dimension one cofaces of the i-simplices in the ith iteration of the loop. For each i-simplex τ on line 3, we compute the set N of the vertices in G that are neighbors of all the vertices of τ and precede τ's vertices in the ordering. For any vertex $v \in N$, we now claim that $\sigma = \tau \cup \{v\}$ is an $(i+1)$-simplex as all its edges are in G. To see this, take any two distinct vertices in $u \neq w \in \sigma$. If both $u, w \in \tau$, then $\{u, w\} \in G$ as τ is a simplex in $\mathcal{V}$. For the other case, assume WLOG that only $w \in \tau$ and $u = v$. Then since $v \in N$, $\{u, w\} \in G$. Therefore, all edges of σ are in G and σ is an $(i+1)$-simplex in $\mathcal{V}$. We add σ to the complex on line 6. We end by noting that the inequality within LOWER-NBRS enforces the invariant that v is the minimal vertex of σ according to the ordering. Since any simplex has a minimal vertex, we report each simplex only once. □

Analyzing the complexity of VR expansion algorithms is difficult, as the output has exponential size, so we need output-sensitive complexity. Here, we consider the first expansion from edges to triangles, when $i = 1$. For each edge, we look at all the neighbors of the endpoints on line 4 to compute the set of shared neighbors. Each vertex is visited as many times as the degrees of its neighbors. By double counting, the total number of visits is $\sum_{v \in V} \deg(v)^2$. This sum was bounded by de Caen [11]:

$$\frac{4|E|^2}{|V|} \leq \sum_{v \in V} \deg(v)^2 \leq |E| \cdot \left(\frac{2|E|}{|V-1|} + |V| - 2 \right),$$

with equality on the lower bound iff G is regular [36]. In the worst-case, $|E| = O(|V|^2) = O(n^2)$ for n points, giving us a cubic bound, which is the same as the brute-force method that considers all $\binom{n}{3}$ triples of points. In practice, however, we have sparse graphs. For instance, when $|E| = O(|V|)$, the upper bound is quadratic. For the next iteration, we only need to sum the degrees of the vertices of triangles in the complex. Worst-case analysis becomes meaningless as the triangles are very sparse in practice. It is not clear how to analyze the general situation when we have a bound on the number of triangles.

4.2 Incremental Algorithm

While the algorithm from the previous section is intuitive, it is not necessarily efficient as it computes the simplices in order of dimension. As a result, it repeats some of the computations on line 4 as it collects the neighbors of a simplex and its faces in different iterations. We now take an alternate approach of adding vertices incrementally. When a vertex is added, we construct all needed cofaces for which the vertex is maximal, eliminating the repeated computations.

INCREMENTAL-VR(G, k)
1 $\mathcal{V} \leftarrow \emptyset$
2 **foreach** $u \in G.V$
3 **do** $N \leftarrow$ LOWER-NBRS(G, u)
4 ADD-COFACES$(G, k, \{u\}, N, \mathcal{V})$
5 **return** $\mathcal{V}$

ADD-COFACES$(G, k, \tau, N, \mathcal{V})$
1 $\mathcal{V} \leftarrow \mathcal{V} \cup \{\tau\}$
2 **if** $\dim(\tau) \geq k$
3 **then return**
4 **else foreach** $v \in N$
5 **do** $\sigma \leftarrow \tau \cup \{v\}$
6 $M \leftarrow N \cap$ LOWER-NBRS(G, v)
7 ADD-COFACES$(G, k, \sigma, M, \mathcal{V})$

Theorem 2 INCREMENTAL-VR(G, k) *computes the k-skeleton of $\mathcal{V}(G)$.*

Proof: At the start of each iteration of the loop in INCREMENTAL-VR, $\mathcal{V}$ contains all simplices of the k-skeleton whose maximal vertex (according to the ordering) precede the current vertex u. This invariant is maintained in the body of the loop: ADD-COFACES adds all simplices of the k-skeleton whose maximal vertex is u. This recursive procedure has the following requirements on its input:

1. τ is a simplex of the k-skeleton

2. N is the set of the lower neighbors of the vertices of τ

These are initially correct as INCREMENTAL-VR calls this procedure with a vertex and its lower neighbors and by Definition (4), all vertices are simplices in $\mathcal{V}$. The body of ADD-COFACES maintains this invariant. On line 1, it adds τ to $\mathcal{V}$ as it is a legitimate simplex due to the invariant. Line 2 is the base case of the recursion. At the start of the loop on line 4, we know τ is a simplex in $\mathcal{V}$ and any vertex $v \in N$ is a neighbor of the vertices of τ. By the argument in the proof of Theorem 1, $\sigma = \tau \cup \{v\}$ must also be a simplex of $\mathcal{V}$. Moreover, $\dim(\sigma) \leq k$ because of the base case. Therefore, we compute σ's neighbors M on line 6 and add σ's cofaces via a recursive call on line 7. Note that both invariants have been maintained and the procedure recursively adds all cofaces of input τ of dimension k or lower as required. □

The incremental algorithm may be used for applications where the point set S itself is filtered, such as a manifold equipped with a Morse function. For example, our shape description application generates point sets filtered by curvature [8]. We simply add the points according to the filtration ordering and modify the definition of the weight function ω to be the maximum of the weights of the vertices.

4.3 Maximal Algorithm

Both our previous algorithms work bottom-up, starting with lower-dimensional faces and adding cofaces. We now present a top-down algorithm based on the following simple observation: *The VR complex is the clique complex of the VR graph.* Recall that a *clique* is a set of vertices in a graph that induces a complete subgraph [9]. A clique is *maximal* if it cannot be made any larger. The *clique complex*, also called the *flag complex*, has the maximal cliques of a graph as its maximal simplices [25]. Figure 3 highlights the maximal cliques of our example graph that become simplices in the VR complex in Figure 2(c). The algorithm, then, is simple: First enumerate the all maximal cliques C; Then, generate all $(k+1)$-combinations of these cliques to get the k-skeleton of $\mathcal{V}$.

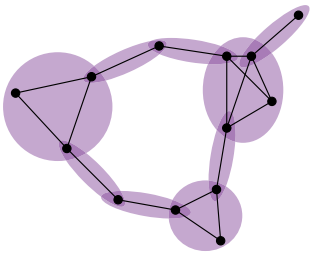


Figure 3: Maximal cliques. The oval regions highlight the 9 maximal cliques of the graph in Figure 2(b) that become maximal simplices in the VR complex in Figure 2(c).

MAXIMAL-VR(G, k)
1 $C \leftarrow$ IK-GX(G)
2 $\mathcal{V} \leftarrow$ GENERATE-COMBINATIONS$(C, k+1)$
3 **return** $\mathcal{V}$

The problem of enumerating *maximal* cliques should not be confused with the *maximum clique problem*, which is a classic NP-complete problem [23]. The enumeration problem is harder since an n-vertex graph may have up to $3^{n/2}$ maximal cliques in the worst case [27], so it is not possible to have polynomial-time algorithms with respect to input size. There have been two approaches to this problem:

- Greedy: Greedy algorithms use a depth-first-search approach for enumeration, starting with the algorithm of Bron and Kerbosch [4]. Koch introduces a pivoting heuristic to reduce the size of the recursion tree by eliminating repeated subtrees [24].

- Output sensitive: These algorithms bound the time between the computation of consecutive maximal cliques [34]. The best current algorithm enumerates all maximal cliques using n^2 space with time delay $O(M(n))$, where $M(n)$ is the cost of multiplying two $n \times n$ matrices.

Greedy algorithms cannot be compared to output sensitive algorithms as no output-sensitive complexity is known for the former. While the latter are provably optimal, the former outperform them in practice. We use the algorithm IK-GX which is based on a variation of Koch's pivoting strategy [7].

In GENERATE-COMBINATIONS, we need to generate all combinations of every maximal clique, viewed as a set. Each r-combination of a maximal clique σ is an $(r-1)$-face of σ as a simplex. Generating combinations is a classic problem in discrete mathematics [31]. We generate the combinations lexicographically, but there are also algorithms for other orderings, such as the *Gray encoding*.

Having detailed the two subroutines, it is clear now that our third algorithm is correct.

Theorem 3 MAXIMAL-VR(G, k) *computes the* k*-skeleton of* $\mathcal{V}(G)$.

4.4 Weight Function

We end by extending the graph weight function to the simplices of the VR complex. We assume the simplices are in a filtration ordering, with faces preceding cofaces. INDUCTIVE-VR automatically provides this ordering as it constructs the simplices by dimension. We may ensure a filtration ordering by careful implementation of the other two algorithms. Given the VR complex $\mathcal{V}$ and the graph weight w, the function below returns the weight function $\omega \colon \mathcal{V} \to \mathbb{R}$.

COMPUTE-WEIGHTS$(\mathcal{V}, w)$

```
1  foreach vertex v ∈ V
2       do ω(v) ← 0
3  foreach edge e ∈ V
4       do ω(e) ← w(e)
5  foreach simplex σ ∈ V
6       do WEIGHT(σ, ω)
7  return ω
```

WEIGHT(σ, ω)

```
1  if ω(σ) is defined
2     then return ω(σ)
3     else
4          return ω(σ) ← max_{τ⊂σ} WEIGHT(τ, ω)
```

The following theorem states that the computed weight is correct.

Theorem 4 COMPUTE-WEIGHTS$(\mathcal{V}, w)$ *computes* $\omega\colon \mathcal{V} \to \mathbb{R}$.

Proof: The recursive algorithm directly follows the inductive definition (Definition 4). The algorithm WEIGHT both computes and assigns the weight, short-circuiting the computation. □

5 Experiments

In this section, we describe implementations of our algorithms and show their performance on both real and synthetic data. We also compare our performance with existing software. We end by demonstrating the utility of our software by empirically verifying recent theoretical results on the topology of Erdős-Rényi graphs.

Our implementations are in generic C++ and part of a library we are developing for computational topology. In particular, we use data structures from this library for creating simplicial complexes and filtrations. We use the *Boost Graph Library (BGL)* to store neighborhood graphs [33]. For nearest neighbors, we use the ANN library [28]. As stated earlier, all our timings are done on a 64-bit GNU/Linux machine with two dual-core 3 GHz Xeon processors, although our software is not threaded and uses only one core. We measured all timings with `clock()` from the Standard C library, and zero means that measured time was below the granularity of this function.

5.1 Data

Our datasets are listed in in Table 1. For each dataset S, we have a maximum scale $\hat{\epsilon}$ and we set $k = \dim S$, that is, we compute up to the embedding dimension. G is Gramicidin A, a small protein. M is a portion of the *van Hateren-Mumford dataset* derived from natural images with parameters $k = 30$ and cut $= 20\%$ [5]. B and D are points sampled from the surface of the *Stanford bunny* and *dragon*, respectively. To create S, we use Muller's method to pick points from a uniform distribution on unit 3-sphere [29]. We then use the 2-fold *diagonal map* $x \to (x, x)$ to embed the points in $\mathbb{R}^8$ [20]. We use our final dataset to demonstrate our ability to compute arbitrary clique complexes, not just VR complexes. The dataset E, is a dense Erdős-Rényi graph $G(n, p)$, where $n = 100$ and each edge is inserted independently with probability $p = 0.6$. We construct this graph as follows. For each graph edge, we choose a weight w uniformly at random from $[0, 1]$. If $w \leq 0.6$, we insert the edge with that weight w into E.

5.2 Phase I: Neighborhood Graphs

Recall from Section 3.2 that in this phase, our input is a set of points $S \subseteq \mathbb{R}^d$ and a maximum scale $\hat{\epsilon} \in \mathbb{R}$, and our goal is to construct the VR neighborhood graph $(G_\epsilon(S), w)$. We have a choice of exact, approximate, or randomized methods for this construction. Figure 4 displays the performance of four nearest neighbor algorithms for constructing the neighborhood graph on dataset S. At $\epsilon = 0.5$, we construct an 11.6 million edge graph in less than 20 seconds on average over 20 independent runs.

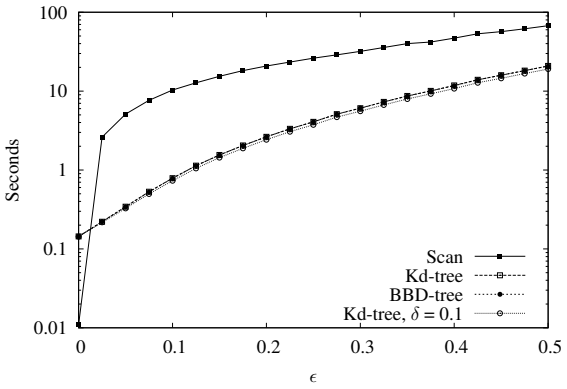


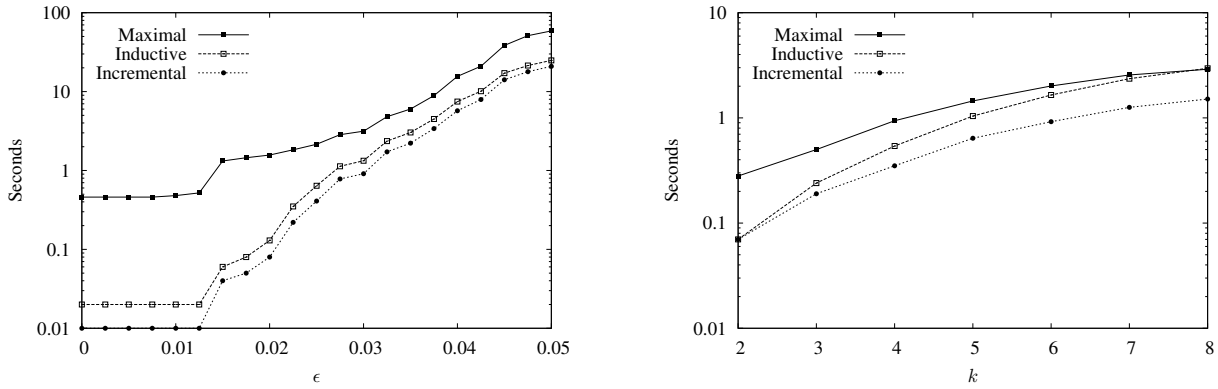
Figure 4: Phase I: construction time for neighborhood graph (G, w) for dataset S. The methods compared are scanning, kd-tree, BBD-tree, and kd-tree with error $\delta = 0.1$. The first three methods are exact and compute the VR neighborhood graph. The last method computes an approximate graph.

We also tested E^{2}LSH which is an implementation of the exact randomized hashing method for L_2 spaces [1]. The software was significantly slower due to its long preprocessing time, e.g. over 622 seconds for a set of 10,000 points on a 2-sphere. Theoretically, this is to be expected, as the method is designed for manifolds with high-intrinsic dimension embedded in even higher dimensions. Practically, this is an early implementation (version 0.1), so we expect the performance to improve in time.

From our experiments, we conclude that exact methods are sufficient for constructing VR neighborhood graphs.

	Input		Phase I			Phase II				T_{J}
S	$\lvert S\rvert$	$\hat{\epsilon}$	T_G	$\lvert E\rvert$	k	$\lvert\mathcal{V}_{\hat{\epsilon}}(S)\rvert$	T_{D}	T_{C}	T_{M}	
G	318	5.00	0.00	3,960	3	71,032	0.12	0.14	0.22	0.13
M	10,000	0.11	0.17	16,210	8	539,627	2.96	1.51	2.90	6.68
B	34,837	0.05	0.37	489,876	3	9,714,912	24.84	20.90	58.81	313.38
S	50,000	0.18	2.17	546,388	8	19,134,612	58.89	121.31	99.70	–
D	88,571	0.0014	1.08	543,996	3	45,995,489	197.23	123.50	–	–
E	100	0.60	0.00	2,978	6	1,683,151	5.61	11.88	9.98	–

Table 1: Data, Timing (in seconds), and Statistics



(a) B (b) M

Figure 5: VR expansion. (a) 3-dimensional complex for B. (b) k-dimensional complex for M at scale $\epsilon = 0.11$.

We may be able to tolerate small errors in data analysis, such as 10%, in order to use approximate methods. But these methods are not sufficiently faster at this error rate to justify their use. Finally, randomized methods are not competitive at this time. For Phase I, Table 1 lists the time T_G for computing the neighborhood graph using exact methods with the ANN library, and the number of edges $|E|$ in the computed graph.

5.3 Phase II: VR Expansion

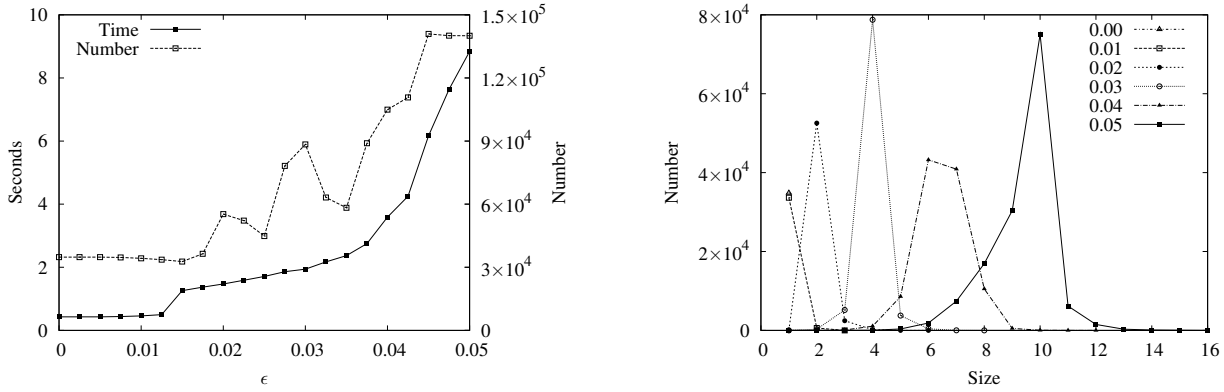
Recall from Section 4 that in this phase, our input is a neighborhood graph (G, w) and a maximum dimension $k \in \mathbb{Z}^{>1}$, and our goal is to compute the k-skeleton of the VR complex $\mathcal{V}(G)$ and its weight function ω. Figure 5 displays the running times for our algorithms on datasets B and M. The times include computing the weight function via COMPUTE-WEIGHTS in Section 4.4. The fastest algorithm is INCREMENTAL-VR and the slowest is MAXIMAL-VR, with INDUCTIVE-VR usually somewhere in between, becoming less competitive in higher dimensions. For Phase II, Table 1 lists the size $|\mathcal{V}_{\hat{\epsilon}}(S)|$ of the computed VR complex as well as times T_{X} for expansion, where X is D for INDUCTIVE-VR, C for INCREMENTAL-VR, and M for MAXIMAL-VR. Since our current implementation for combinations uses bit packing, we are limited to 63-cliques and cannot handle dataset D, which contains a 73-clique at $\hat{\epsilon}$. It is easy to remove this limitation, however.

5.4 Enumerating Maximal Cliques

Our third algorithm, MAXIMAL-VR, initially enumerates all maximal cliques. Figure 6 gives enumeration time, number of maximal cliques, and histograms of the sizes of maximal cliques for dataset B. At $\epsilon = 0$, we have 34,837 1-cliques, corresponding to the points in S. At $\epsilon = \infty$, we would have one 34,837-clique. The number of maximal cliques is nonmonotonic with increasing ϵ since small cliques merge into larger cliques. The histograms are usually leptokurtic (have positive kurtosis and are pointy) and shift right with increasing ϵ. As the figures demonstrates, we have a large number of cliques, e.g. 75,011 10-cliques at $\hat{\epsilon} = 0.05$. Table 2 lists the time in seconds to enumerate maximal cliques, their number, and average and maximum size at the maximum scale. For dataset B at $\hat{\epsilon} = 0.05$, the neighborhood graph $G_{\hat{\epsilon}}(S)$ has 489,876 edges or roughly 0.08% of all edges, so we are in a sparse regime from a graph-theoretic point of view. Dataset E, in contrast, is a dense graph, containing approximately 60% of the edges, and the resulting number and size of cliques reflect its density.

5.5 Comparison with JPlex

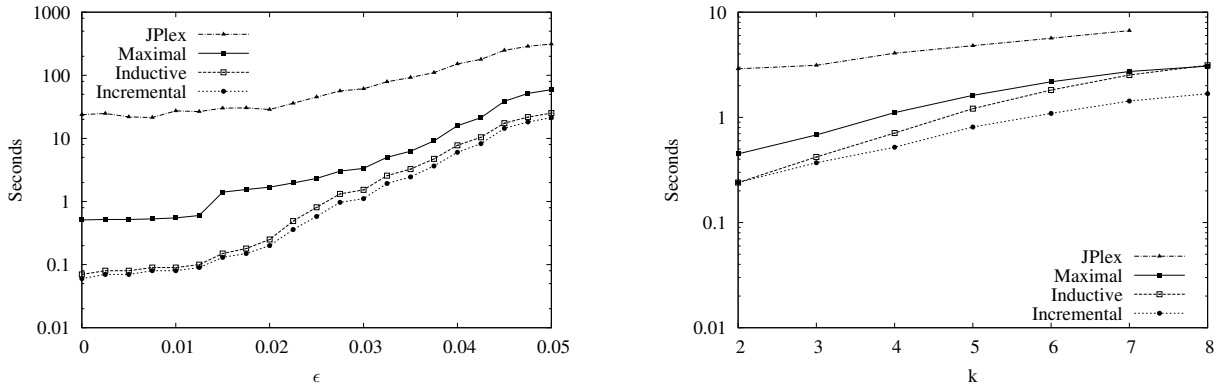
We next compare the performance of our algorithms with JPlex [32] for the full construction of the VR complex.



(a) Time and Number

(b) Overlaid Histograms

Figure 6: Enumerating maximal cliques with IK-GX on B. (a) Enumeration time and number of maximal cliques. (b) Histogram of size of maximal cliques at the specified ϵ. For better visibility, we do not use boxes.



(a) B

(b) M

Figure 7: Comparison with JPlex for the full construction of the VR complex. (a) On B dataset with increasing ϵ. (b) On M for $\epsilon = 0.11$ with increasing dimension k. JPlex can compute up to 7 dimensions currently.

S	time (s)	number	ave	max
G	0.03	870	8.97	13
M	0.16	10,419	2.93	17
B	8.78	140,052	9.39	16
S	8.96	167,742	6.56	18
D	39.29	123,091	6.91	73
E	0.71	64,055	7.65	11

Table 2: Maximal Clique Enumeration

Currently, JPlex can compute complexes up to 7 dimensions only. It also does not compute the full weight function but allows the user to discretize it using a parameter that we set to 0.0001 in our experiments. We compare the algorithms along two axes: increasing maximum scale or dimension. In Figure 7(a), we compute on dataset B with increasing scale. Our fastest algorithm is about 15 times faster than JPlex at the maximum scale, much faster on lower scales, and uses about 5 times less memory (1.1 GB vs. 5.3 GB). In Figure 7(b), we compute on dataset M with increasing dimension. On this dataset, we are about three times faster than JPlex even though we also compute the 8-dimensional simplices. In Table 1, T_{J} is the time for computing with JPlex. For dataset M, JPlex computes the 7-skeleton with 472,165 simplices in the time shown. JPlex is not able to process datasets S and D due to difficulties with memory allocation. It also cannot compute clique complexes, so we have no time measurement for dataset E.

5.6 Homology of Erdős-Rényi Graphs

We end this paper with an application of our software to the computation of homology of Erdős-Rényi graphs, a model of random graphs. We say that $G(n,p)$ *almost always (a.a.)* has property $\mathcal{P}$ if $\Pr(G(n,p) \in \mathcal{P}) \to 1$ as $n \to \infty$. The seminal *Erdős-Rényi Theorem* sets thresholds for connectivity: If $p = (\log n + \omega(n))/n$ and $\omega(n) \to \infty$ as $n \to \infty$, then $G(n,p)$ is a.a. connected; if $\omega(n) \to -\infty$, then $G(n,p)$ is a.a. disconnected [16].

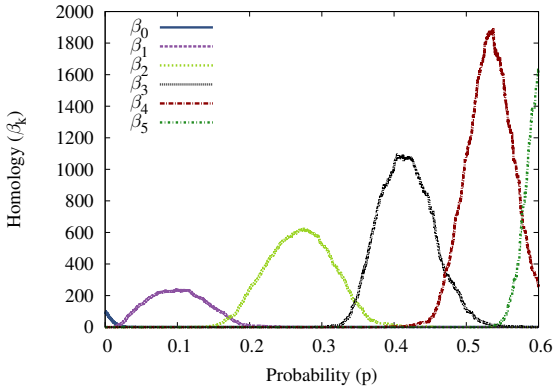


Figure 8: Filtered homology of dataset E, an Erdős-Rényi graph. We give the ranks β_i, $0 \leq i \leq 5$, of the homology groups of the clique complex for $0 \leq p \leq 0.6$.

In other words, below some threshold for p, the graph is usually disconnected, and above some threshold, it is usually connected. Connectivity may be viewed as zero-dimensional homology. Recently, Kahle [22] gives higher-dimensional analogues to the Erdős-Rényi theorem by studying homology groups of clique complexes of these graphs. Specifically, he shows that for $k > 0$, if $p = n^\alpha$, with $\alpha < -1/k$ or $\alpha > -1/(2k+1)$, then the kth homology group of the clique complex is a.a. vanishing, and if $-1/k < \alpha < -1/(k+1)$, then it is a.a. nonvanishing. In other words, there is a unimodality for each fixed homology group as $n \to \infty$. We verify Kahle's results empirically by computing the homology of E. Figure 8 shows the ranks of the homology groups, the *Betti numbers* β_i, $0 \leq i \leq 5$, that we can compute since we constructed the 6-skeleton for E, an instance of $G(100, 0.6)$. What is striking about the graph is that while Kahle's results are asymptotic, the homology has already converged to unimodality at $n = 100$.

6 Conclusion

In this paper, we present a two-phase approach to computing the Vietoris-Rips complex, examine each phase in detail, implement all algorithms, and present experimental results on their performance. Our work represents the first systematic examination of all the stages of the construction of the Vietoris-Rips complex. Our software is currently the fastest available, constructing complexes with millions of simplices in arbitrary dimensions in seconds. The algorithms may be used in constructing other clique complexes, such as the weak witness complex. We demonstrate their generality and utility by computing homology of the clique complex of an Erdős-Rényi graph.

From the three algorithms for VR expansion, we find the maximal one most intriguing. While it is not the fastest, the algorithm is naturally parallelizable. Moreover, it gives a *minimum* description of the complete complex as a set of maximal simplices, unlike the bottom-up algorithms. For these reasons, this algorithm is the basis of our current work in computing smaller representations for topological data analysis [38].

ACKNOWLEDGMENTS

The author would like to thank Frédéric Cazals for discussion on maximal cliques, Doug Moore for his implementation of lexicographic generation of combinations, Harlan Sexton for JPlex, and Gunnar Carlsson for initial discussions and for his support. William Chen ran timing experiments with ANN and E^{2}LSH, and Gabriel Weaver performed initial experiments with clique enumeration. Research by the author was partially supported by DARPA HR 0011-06-1-0038, ONR N 00014-08-1-0908, and NSF CAREER CCF-0845716.

References

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

[3] J.-D. Boissonnat, O. Devillers, and S. Hornus. Incremental construction of the Delaunay triangulation and the Delaunay graph in medium dimension. In *Proc. ACM Symposium on Computational Geometry*, pages 208–216, 2009.

[4] C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques in an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

[5] G. Carlsson, T. Ishkhanov, V. de Silva, and A. Zomorodian. On the local behavior of spaces of natural images. *International Journal of Computer Vision*, 76(1):1–12, 2008.

[6] F. Cazals, J. Giesen, M. Pauly, and A. Zomorodian. The conformal alpha shape filtration. *The Visual Computer*, 22(8):531–540, 2006.

[7] F. Cazals and C. Karande. Reporting maximal cliques: new insights into an old problem. Research Report 5642, INRIA, 2005.

[8] A. Collins, A. Zomorodian, G. Carlsson, and L. Guibas. A barcode shape descriptor for curve point cloud data. *Computers and Graphics*, 28(6):881–894, 2004.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, third edition, 2009.

[10] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, New York, second edition, 2000.

[11] D. de Caen. An upper bound on the sum of squares of degrees in a graph. *Discrete Mathematics*, 185:245–248, 1998.

[12] V. de Silva and G. Carlsson. Topological estimation using witness complexes. In *Proc. IEEE/Eurographics Symposium on Point-Based Graphics*, pages 157–166, 2004.

[13] R. O. Duda and P. E. Hart. *Pattern Classification*. John Wiley & Sons, Inc., New York, NY, second edition, 2000.

[14] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002.

[15] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13:43–72, 1994.

[16] P. Erdős and A. Rényi. On random graphs, I. *Pulicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[17] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, pages 209–226, 1977.

[18] J. Giesen and M. John. The flow complex: A data structure for geometric modeling. In *Proc. Symposium on Discrete Algorithms*, pages 285–294, 2003.

[19] M. Gromov. Hyperbolic groups. In S. Gersten, editor, *Essays in Group Theory*, pages 75–263. Springer-Verlag, New York, NY, 1987.

[20] A. Hatcher. *Algebraic Topology*. Cambridge University Press, New York, NY, 2002.

[21] ISO/IEC. Digital compression and coding of continuous-tone still images: Requirements and guidelines (10918-1), 1994.

[22] M. Kahle. Topology of random clique complexes. *Discrete Mathematics*, 309:1658–1671, 2009.

[23] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations (Symposium Proceedings)*. Plenum Press, 1972.

[24] I. Koch. Enumerating all connected maximal common subgraphs in two graphs. *Theoretical Computer Science*, 250(1–2):1–30, 2001.

[25] D. Kozlov. *Combinatorial Algebraic Topology*. Springer-Verlag, New York, NY, 2008.

[26] Y. Lifshits. The homepage of nearest neighbors and similarity search, 2009. `http://simsearch.yury.name/`.

[27] J. W. Moon and L. Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3(1):23–28, 1965.

[28] D. M. Mount and S. Arya. ANN: A library for approximate nearest neighbor searching version 1.1.1, 2006. `http://www.cs.umd.edu/~mount/ANN/`.

[29] M. E. Muller. A note on a method for generating points uniformly on n-dimensional spheres. *Communications of the ACM*, 2(4):19–20, 1959.

[30] P. Perry and V. de Silva. PLEX 2.5: Simplicial complexes in MATLAB, 2006. `http://math.stanford.edu/comptop/programs/plex.html`.

[31] K. H. Rosen. *Discrete Mathematics and Its Applications*. McGraw-Hill, New York, NY, sixth edition, 2006.

[32] H. Sexton and M. V. Johansson. JPlex, 2009. `http://comptop.stanford.edu/programs/jplex/`.

[33] J. Siek, L.-Q. Lee, and A. Lumsdaine. *The Boost Graph Library: user guide and reference manual*. Pearson Education, Inc., Upper Saddle River, NJ, 2002.

[34] S. Tsukiama, M. Ide, H. Ariyoshi, and I. Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM Journal on Computing*, 6(3):505–517, 1977.

[35] L. Vietoris. Über den höheren zusammenhang kompakter räume und eine klasse von zusammenhangstreuen abbildungen. *Mathematische Annalen*, 97(1):454–472, 1927.

[36] Y. S. Yoon and J. K. Kim. A relationship between bounds on the sum of squares of degrees of a graph. *Journal of Applied Mathematics and Computing*, 21(1–2), 2006.

[37] A. Zomorodian. Computational topology. In M. Atallah and M. Blanton, editors, *Algorithms and Theory of Computation Handbook*, volume 2, chapter 3. Chapman & Hall/CRC Press, Boca Raton, FL, second edition, 2010.

[38] A. Zomorodian. The tidy set: A minimal simplicial set for computing homology of clique complexes. In *Proc. ACM Symposium of Computational Geometry*, 2010. (To appear).

[39] A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, 2005.