The Tidy Set: A Minimal Simplicial Set for Computing Homology of Clique Complexes*

[Extended Abstract]

Afra Zomorodian
Department of Computer Science
Dartmouth College
Hanover, New Hampshire
afra@cs.dartmouth.edu

ABSTRACT

We introduce the tidy set, a minimal simplicial set that captures the topology of a simplicial complex. The tidy set is particularly effective for computing the homology of clique complexes. This family of complexes include the Vietoris-Rips complex and the weak witness complex, methods that are popular in topological data analysis. The key feature of our approach is that it skips constructing the clique complex. We give algorithms for constructing tidy sets, implement them, and present experiments. Our preliminary results show that tidy sets are orders of magnitude smaller than clique complexes, giving us a homology engine with small memory requirements.

Categories and Subject Descriptors

I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Curve, surface, solid, and object representations*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*

General Terms

Algorithms, Theory

Keywords

computational topology, Vietoris-Rips complex, witness complex, simplicial set

*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.

SCG'10, June 13–16, 2010, Snowbird, Utah, USA.

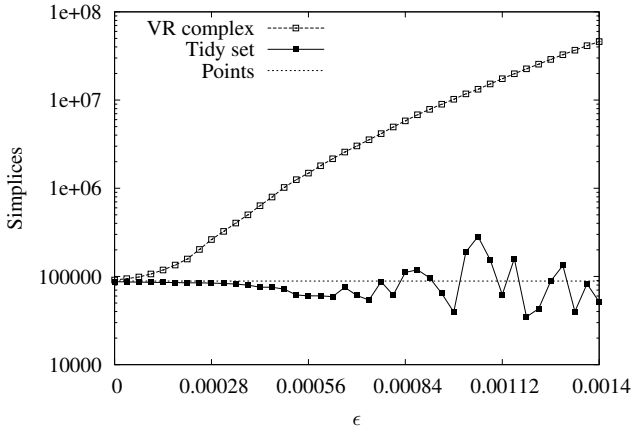


Figure 1: Size of the Vietoris-Rips complex versus the tidy set at scale ϵ for dataset D in Section 5.1, a set of 88,571 points in $\mathbb{R}^3$. Note: y-axis is log scale.

1. INTRODUCTION

In this paper, we introduce the tidy set, a minimal simplicial set that captures the topology of a simplicial complex. We give algorithms for computing the tidy set, implement the algorithms, and present preliminary experimental results. Our method is particularly effective computing the homology of clique complexes without first constructing them. To inspire the reader, Figure 1 compares computing homology of a point set with a clique complex, the Vietoris-Rips complex, versus the tidy set. At the highest scale ϵ, the standard method constructs a simplicial complex with 46M simplices. Our method produces a simplicial set that has the same homology but is nearly three orders of magnitude smaller. With only 51K simplices, this tidy set is even smaller than the input point set. Our method is also five times faster than the standard method.

1.1 Motivation

We are motivated by *topological data analysis*, where the goal is recovering the lost topology of sampled data [31]. The standard recovery process has two steps: First, we approximate the underlying space of data using a combinatorial structure; Second, we compute topological invariants of this structure, such as homology. There are a number

of methods for completing the first step, such as the Čech complex [15], the alpha complex [10], and the flow complex [12], to name a few. These methods, especially when geometric in nature, do not scale to large datasets or extend to high dimensions. For example, we currently do not have efficient software for computing the alpha complex in dimensions higher than three. There are currently two methods used in practice. The Vietoris-Rips complex [14] is popular due to its algorithmic simplicity, but yields massive non-embedded complexes even for small point sets. The (weak) witness complex [9] mitigates the problem of size by only approximating the topology over a subset of the data. Both of these complexes are clique complexes of certain graphs. Therefore, our goal is to find methods for computing homology of clique complexes of large datasets. We do so by computing a minimal simplicial set directly from the graph itself without first constructing the clique complex. We call this simplicial set the tidy set.

1.2 Prior Work

A common assumption in topological analysis is that step one – constructing a complex – is easy, while step two – computing homology – is hard. This assumption is somewhat justified since the classic *reduction algorithm* for computing homology over integers has supercubical complexity in the size of the complex [28]. Over field coefficients, reduction simplifies to Gaussian elimination, still having quadratic space and cubic time complexity. It is a reasonable strategy to search for heuristics that reduce the size of the complex while preserving its topology. There have been a number of reduction techniques proposed for different categories of structures. For simplicial complexes, the earliest and perhaps simplest method is *elementary contraction*, proposed by Whitehead in defining the simple homotopy type [29]. A stronger technique is the recent *LC-reduction* [5] that produces not only a homotopic, but isomorphic complex [19, 1]. Both these techniques produce simplicial complexes with fewer simplices. Due to the popularity of simplicial complexes in computational topology, the techniques have been widely used.

For *cubical complexes*, the CHomP project [4] has examined a large number of heuristics over the years, resulting in an array of homology engines [16]. Our methods in this paper are analogous to those by Mrozek et al. [22]. Our work was done independently from this group and applies to arbitrary-dimensional simplicial complexes, not three-dimensional cubical complexes. The primary characteristic that unifies both works is using cell collapses. This reduction moves us from the familiar simplicial complex category to the unfamiliar simplicial set category. The only instance of simplicial sets in computational topology that we know of is Perry [24] who describes an implementation of simplicial sets and an algorithm for computing their homology.

1.3 Our Approach

Our approach is based on our experience in applying topological techniques to problems in a number of areas, such as shape description [6], biophysics [17], and computer vision [2]. We find that the opposite of the traditional assumption is true: Constructing a complex is hard as non-embedded complexes, such as clique complexes, can get very large, very fast, requiring large construction time and storage. Also, computing homology over fields is easy using the *persistence algorithm* [33] as it represents matrices sparsely, resulting in linear time behavior. A significant repercussion of these two observations is that standard reduction techniques are not competitive for us. We spend most of time building a massive complex, not computing its homology.

Our approach, then, is to avoid constructing the full complex by reducing it *during* its construction. We make reduction techniques effective by applying them to high-dimensional simplices only as simplices have exponential complexity. We maintain a compact representation of the structure throughout computation and postpone enumerating the structure and computing its homology as long as possible.

The contributions of this paper are as follows. In Section 3, we define the tidy set, a minimal simplicial set that has the same homology as an input simplicial complex. In Section 4, we give algorithms for computing the tidy set as well as for computing homology of clique complexes directly. In Section 5, we give preliminary results of our implementation of the algorithms, including computing homology of point sets using the Vietoris-Rips complex. Our implementation can process large datasets in arbitrary dimensions, datasets that are currently not handled by any existing software. It is also fast, takes less memory, and produces tidy sets that are orders of magnitude smaller, often sublinear in input size.

2. BACKGROUND

We begin with a review of simplicial complexes and their generalization, simplicial sets. We are interested in simplicial sets that may be derived from simplicial complexes via collapses, which we describe next. We then describe homology for both categories. We end this section with clique complexes, the family of complexes whose homology we wish to compute. For an accessible introduction to computational topology, see [31].

2.1 Simplicial Complex

A *simplicial complex* is a set K of finite sets, closed under the subset relation: If $\sigma \in K$ and $\tau \subseteq \sigma$, then $\tau \in K$. We then say that τ is a *face* of σ, its *coface*. 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*, denoted $\dim \sigma = k$. K is d-dimensional if $d = \dim K = \max_{\sigma \in K} \dim \sigma$. Given a subset $L \subseteq K$, L is a *subcomplex* if it is a simplicial complex. Otherwise, the smallest subcomplex containing L is its *closure* under subset: $\mathrm{Cl}\, L = \{\tau \in K \mid \tau \subseteq \sigma \in L\}$. A key subcomplex is the *k-skeleton* consisting of simplices in K of dimension less than or equal to k.

2.2 Simplicial Set

A *simplicial set*, initially called the *complete semi-simplicial complex* or *c.s.s. complex*, generalizes a simplicial complex to model a well-behaved topological space [8, 11, 20]. Informally, a simplicial set is like a simplicial complex where simplices may be collapsed to a point, and vertices may be identified. Formally, a *simplicial set* X is a collection of sets $\{X_n\}_n$ together with maps

$$d_i \colon X_n \to X_{n-1},$$
$$s_i \colon X_n \to X_{n+1},$$

for $0 \le i \le n$, that satisfy the following identities:

$$d_i \circ d_j = d_{j-1} \circ d_i, \quad \text{if } i < j.$$
$$s_i \circ s_j = s_{j+1} \circ s_i, \quad \text{if } i \le j.$$
$$d_i \circ s_j = \begin{cases} s_{j-1} \circ d_i, & \text{if } i < j, \\ 1_{X_n}, & \text{if } i = j \text{ or } i = j+1, \\ s_j \circ d_{i-1}, & \text{if } i > j+1, \end{cases}$$

where 1_{X_n} is the identity on X_n. An element of X_n is an *n-simplex*, the map d_i is the *ith face operator*, and s_i is the *ith degeneracy operator* [8, 20]. Simplicial sets may be elegantly reinterpreted in the language of category theory [13].

In this paper, we are interested simplicial sets that are derived from simplicial complexes, such as the seven 2-simplices in Figure 2. Let K be a simplicial complex and denote its n-simplices by K_n. We assume that the vertices in each simplex are ordered. For instance, a total order on the vertices in the complex implies an order for each simplex. We define the simplicial set X that corresponds to simplicial complex K inductively:

$$X_0 = K_0,$$
$$X_n = K_n \cup \cup_i^n s_i(X_{n-1}), \quad n > 0.$$

The ith face operator d_i deletes the ith vertex, and the ith degeneracy operator s_i repeats it:

$$d_i([v_0, \ldots, v_n]) = [v_0, \ldots, \hat{v_i}, \ldots, v_n],$$
$$s_i([v_0, \ldots, v_n]) = [v_0, \ldots, v_i, v_i, \ldots, v_n].$$

It is easy to verify that these operators satisfy the simplicial identities. A simplex that may be written as $s_i(\sigma)$ for some σ is *degenerate* and is not in the simplicial complex. Otherwise, the simplex is *non-degenerate*.

Example 1 (triangle) Consider the triangle abc in Figure 2. As a simplicial complex K, abc has

$$K_0 = \{a, b, c\},$$
$$K_1 = \{ab, bc, ac\},$$
$$K_2 = \{abc\},$$

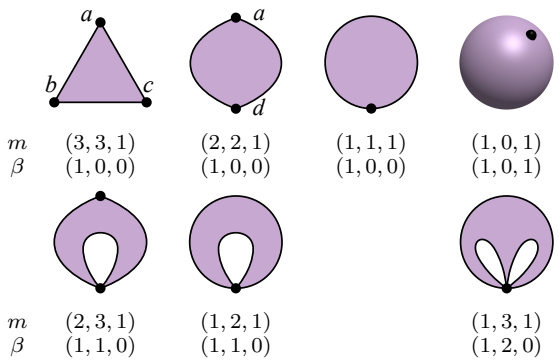


Figure 2: The 7 possible 2-simplices in a simplicial set. The triangle is the only one allowed in a complex. The rest have collapsed edges (top row), identified vertices (bottom row), or both. The vector m counts the non-degenerate simplices and the vector β holds the Betti numbers.

where we denote simplices as strings for brevity. As a simplicial set X, abc has

$$X_0 = \{a, b, c\},$$
$$X_1 = \{ab, bc, ac, aa, bb, cc\},$$
$$X_2 = \{abc, aab, abb, bbc, bcc, aac, acc, aaa, bbb, ccc\}.$$

Note the addition of degenerate simplices. For example, abb may be viewed as a collapsed triangle.

2.3 Simplicial Collapse

Simplicial sets may be constructed directly, but we are interested in sets that is derived from complexes by collapses only. Given a simplicial set X and an n-simplex $\sigma \in X$, the *collapse* of σ identifies it to a single point, giving us a new simplicial set $X' = X/\sigma$. To construct X', we introduce a new vertex v and replace σ, its faces, and its degeneracies, with the degeneracies of v. We first gather its non-degenerate k-faces inductively for $k \ge 0$:

$$\bar{F}_k(\sigma) = \begin{cases} \emptyset, & \text{if } k > n, \\ \{\sigma\}, & \text{if } k = n, \\ \cup_{i=0}^{k+1} d_i(\bar{F}_{k+1}(\sigma)), & \text{if } k < n. \end{cases}$$

By adding the degenerate faces, we get all the faces.

$$F_k(\sigma) = \begin{cases} \bar{F}_0(\sigma), & \text{if } k = 0, \\ \bar{F}_k(\sigma) \cup \bigcup_{i=0}^{k-1} s_i(F_{k-1}(\sigma)), & \text{if } k > 0. \end{cases}$$

We may now define the k-simplices for X' as

$$X'_k = (X_k - F_k(\sigma)) \cup \{s_0^k(v)\},$$

where s_0^k denotes applying the degeneracy operator k times. We next define the operators for the new set:

$$d'_i(\tau) = \begin{cases} d_i(\tau), & \text{if } d_i(\tau) \notin F_{i-1}(\sigma), \\ s_0^{i-1}(v), & \text{otherwise.} \end{cases}$$
$$s'_i(\tau) = \begin{cases} s_i(\tau), & \text{if } s_i(\tau) \notin F_{i+1}(\sigma), \\ s_0^{i+1}(v), & \text{otherwise.} \end{cases}$$

Example 2 (2-gon) In Example 1, we listed the n-simplices of the triangle abc in Figure 2 as a simplicial set X. We now collapse edge bc to a new vertex d to get the 2-gon $X' = ad$ in the figure. We have

$$\bar{F}_2(bc) = \emptyset,$$
$$\bar{F}_1(bc) = \{bc\},$$
$$\bar{F}_0(bc) = F_0(bc) = \{b, c\},$$
$$F_1(bc) = \{bc, bb, cc\},$$
$$F_2(bc) = \{bbc, bcc, bbb, ccc\},$$
$$X'_0 = \{a, d\},$$
$$X'_1 = \{ab, ac, aa, dd\},$$
$$X'_2 = \{abc, aab, abb, aac, acc, aaa, ddd\}.$$

The operators follow easily, e.g. $d'_0(abc) = dd$.

Generally, we may also identify vertices in constructing simplicial sets. While we do not use vertex identifications in this paper, vertices may be identified by collapsed neighbors.

2.4 Simplicial Homology

Simplicial homology extends naturally to simplicial sets. Let X be a simplicial set. The *nth chain group* $C_n(X)$ of X is the free Abelian group on K's set of oriented, non-degenerate, n-simplices. The *boundary homomorphism* $\partial_n \colon C_n(X) \to C_{n-1}(X)$ is the linear extension of

$$\partial_n = \sum_{i=0}^{n} (-1)^i d_i,$$

where d_i are the face operators and a degenerate face is treated as 0. The fundamental property of the boundary homomorphism is that $\partial_n \partial_{n+1} = 0$ for all n.

Example 3 (collapsed ∂) The face operators for our collapsed set in Example 2 give us the correct boundary. For instance, we have $d_0(abc) = dd$, $d_1(abc) = ac$, and $d_2(abc) = ab$, giving us $\partial_2(abc) = -ac + ab$, as dd is degenerate. Taking another boundary, we have

$$\partial_1 \partial_2 (abc) = \partial_1(-ac) + \partial_1(ab) = -(d-a) + (d-a) = 0.$$

The boundary homomorphism connects the chain groups into a *chain complex* C_*:

$$\cdots \to C_{n+1}(X) \xrightarrow{\partial_{n+1}} C_n(X) \xrightarrow{\partial_n} C_{n-1}(X) \to \cdots.$$

Given any chain complex, the *nth homology group* is

$$H_n(X) = \ker \partial_n / \operatorname{im} \partial_{n+1}.$$

Homology is a functor that we may apply to C_* to get another sequence H_*:

$$\cdots \to H_{n+1}(X) \xrightarrow{H(\partial_{n+1})} H_n(X) \xrightarrow{H(\partial_n)} H_{n-1}(X) \to \cdots.$$

Given a subset $A \subseteq X$ that is a simplicial set, we may also define the *relative homology groups* $H_n(X, A)$, where we view the subset A as collapsed onto a single point. A *contractible* space has the homotopy type of a point, and it is often convenient for it to have trivial homology in all dimensions, including zero. For this, we define *reduced homology groups* $\tilde{H}_n(X)$, so that $H_0(X) \cong \tilde{H}_0(X) \oplus \mathbb{Z}$ and $H_n(X) \cong \tilde{H}_n(X)$ for $n > 0$. We say that a space X is *acyclic* if it has trivial reduced homology, i.e. $\tilde{H}_n(X) = 0$ for all n. Contractible spaces, such as simplices in a simplicial complex, are acyclic.

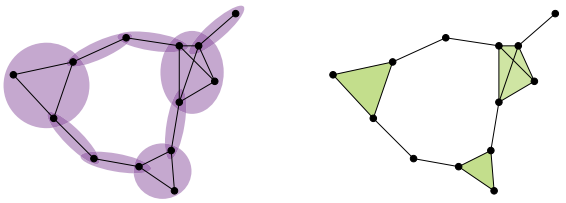
We end this section with an integer topological invariant that is coarser than homology, but may be computed by simple counting. Given a simplicial set X, the *Euler characteristic* $\chi(X)$ is

$$\chi(X) = \sum_n (-1)^n c_n = \sum_n (-1)^n \beta_n,$$

where c_n is the number of non-degenerate simplices in X and $\beta_n = \operatorname{rank} H_n(X)$ [15]. The second equality is the *Euler-Poincaré formula* that allows us to conclude that an acyclic space X has $\chi(X) = 1$, such as the three spaces on the left in Figure 2. The figure also lists m_n and β_n for each space and we may confirm the Euler-Poincaré formula readily.

2.5 Clique Complex

Suppose we are given a graph $G = (V, E)$, such as the graph in Figure 3(a). A *clique* is a set of vertices $Q \subseteq V$ that induces a complete subgraph in G. A clique is *maximal* if it cannot be made any larger. Figure 3(a) highlights



(a) $G = (V, E)$ (b) $\mathcal{C}(G)$

Figure 3: A graph G (a) with shaded ovals highlighting its nine maximal cliques that become maximal simplices in its clique complex $\mathcal{C}(G)$ (b).

the 9 maximal cliques of the graph with shaded ovals. The *clique complex* $\mathcal{C}(G)$ has the maximal cliques of G as its maximal simplices [18]. Since subsets of cliques are also cliques, the clique complex is a simplicial complex, as shown in Figure 3(b) for our example. Different graphs give rise to different families of clique complexes.

Example 4 (Vietoris-Rips) given a finite point set $S \subseteq \mathbb{R}^d$, the *Vietoris-Rips graph (VR graph)* at scale $\epsilon \in \mathbb{R}$ is the geometric graph $G_\epsilon(S) = (S, E_\epsilon(S))$, where $E_\epsilon(S) = \{\{u, v\} \mid \mathrm{d}(u, v) \leq \epsilon, u \neq v \in S\}$. That is, vertices within distance ϵ are connected. The graph in Figure 3(a) is a VR graph. The clique complex of the VR graph is the *Vietoris-Rips complex (VR complex)* [14].

Example 5 (witness) Given a finite point set $S \subseteq \mathbb{R}^d$, choose a subset Alternatively, we may also choose a subset $L \subseteq S$ of *landmarks* and let $W = S - L$. 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 clique complex of this graph is the *(weak) witness complex* [9].

Each method defines a one-parameter filtered family of complexes that describe the shape of the point set at different scales. Both methods are currently popular in topological data analysis and motivate our work on clique complexes.

3. TIDY SET

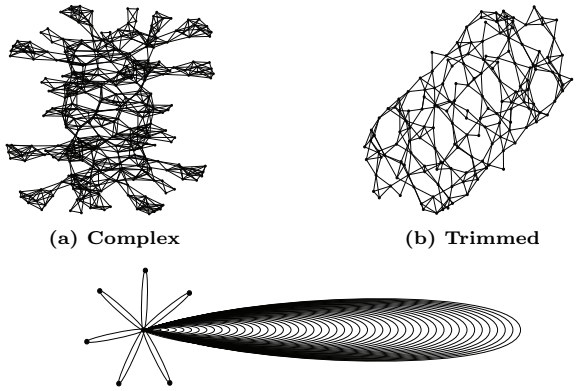
In this section, we define the tidy set, a minimal simplicial complex. We begin by describing two reductions that we use in deriving the tidy set, as previewed in Figure 3. We then describe a minimal representation for simplicial sets. Our representation both simplifies and increases the efficacy of the reductions. We end by showing that the tidy set is minimal with respect to both reductions.

3.1 Reductions

Our first reduction technique is trimming leaves. Intuitively, a leaf in a simplicial complex has an acyclic intersection with the rest of the complex, the intersection being its "stem". We generalize this notion for simplicial sets.

Definition 1 (leaf) *Let X be a simplicial set. A simplex $\sigma \in X$ is a* leaf *if for all n,*

$$H_n(Cl\,\sigma, Cl\,\sigma \cap Cl(X - Cl\,\sigma)) = 0.$$



(a) Complex (b) Trimmed

(c) Tidy Set

Figure 4: 1-skeletons of homologous complexes: Complex (a), trimmed complex (b), and tidy set (c) for dataset G in Section 5.1.

Here, $\mathrm{Cl}\,\sigma$ is σ as a simplicial complex, and $\mathrm{Cl}\,(X - \mathrm{Cl}\,\sigma)$ is the rest of the complex. In a simplicial complex, the definition simplifies to our earlier intuition.

Theorem 1 (complex leaf) *Let K be a simplicial complex and $\sigma \in K$ be a leaf. Then for all n,*

$$\tilde{H}_n(Cl\sigma \cap Cl(X - Cl\sigma)) = 0.$$

PROOF. From the definition of a leaf and an application of the Mayer-Vietoris sequence [15], we have

$$H_n(\mathrm{Cl}\,\sigma) \cong H_n(\mathrm{Cl}\,\sigma \cap \mathrm{Cl}\,(X - \mathrm{Cl}\,\sigma))$$

Simplices in simplicial complexes are contractible and therefore acyclic, so $\tilde{H}_n(\mathrm{Cl}\,(\sigma)) = 0$ for all n. □

Leaves may be deleted without any change in homology.

Theorem 2 *Let X be a simplicial set and $\sigma \in X$ be a leaf. Then for all n, $H_n(X) \cong H_n(Cl(X - Cl\sigma))$.*

PROOF. Let $A = \mathrm{Cl}\,(X - \mathrm{Cl}\,\sigma)$ and $B = \mathrm{Cl}\,\sigma$ and note that $X = A \cup B$. By the definition of a leaf, we have $H_n(B, A \cap B) = 0$ for all n. By excision [15, Corollary 2.24], we have $H_n(B, A \cap B) \cong H_n(X, A)$ for all n. Combining, we get $H_n(X, A) = 0$ giving us $H_n(X) \cong H_n(A)$ for all n through Mayer-Vietoris. □

The idea of removing leaves is not new. For instance, it is called *shaving* for full-dimensional cubes within cubical complexes [25]. Deleting is a favored technique as it reduces the size of a complex without changing its category.

Our second reduction technique is collapsing as defined in Section 2.2. Collapsing changes the category of the structure, from a simplicial complex to a simplicial set, but sometimes, does not change its homology:

Theorem 3 (collapse) *Let X be a simplicial set and $\sigma \in X$. If $Cl\sigma$ is acyclic, then for all n, $H_n(X) \cong H_n(X/\sigma)$.*

PROOF. The proof follows [15, Theorem 2.13]. □

3.2 Definition

A key feature of our approach is that we use a minimal description for representing simplicial sets. We narrow our focus to sets that are complexes with collapsed maximal simplices.

Definition 2 ($\mathcal{X}$) *Let Q and C be disjoint sets of maximal sets. Then $\mathcal{X}(Q, C)$ is the simplicial set having the sets in Q as maximal simplices and the sets in C as collapsed maximal simplices. We use the tuple (Q, C) to denote $\mathcal{X}(Q, C)$.*

Our representation is a natural extension of a minimum representation for simplicial complexes, with $\mathcal{X}$ extending closure as the face enumeration operator.

Definition 3 ($\mathcal{Q}$) *Let K be a simplicial complex. We define $\mathcal{Q}(K)$ to be the set of maximal simplices.*

Theorem 4 (representation) *$\mathcal{Q}(K)$ is a unique representation of minimum size for simplicial complex K.*

PROOF. $\mathcal{Q}(K)$ is a set of maximal sets, where maximality is with respect to subset. Now, $\mathrm{Cl}\,\mathcal{Q}(K) = K$, so $\mathcal{Q}(K)$ generates K. Uniqueness follows from the definition of maximal sets and axioms of specification and extension. For minimality, observe that a maximal set σ must be in any description of K as σ is not a subset of any other set. We may prove this formally using order theory. □

We may denote K with $(\mathcal{Q}(K), \emptyset)$ as a simplicial set. For general simplicial sets, the representation is not unique as we may build the same set through different operations. It also does not have minimum size necessarily, as we may collapse any acyclic set to a single point. It does, however, make reduction operations very easy.

Definition 4 (trim & thin) *Let (Q, C) denote a simplicial set. For $\sigma \in Q$, we have two homology-preserving reductions:*

trim:	$(Q, C) \mapsto (Q - \{\sigma\}, C)$	*σ, a leaf*
thin:	$(Q, C) \mapsto (Q - \{\sigma\}, C \cup \{\sigma\})$	*σ, acyclic*

Note that both reductions maintain the invariant that Q and C are disjoint. Our representation also enables us to postpone enumeration of simplicial sets until we require it for computing homology. We are now ready to define the tidy set.

Definition 5 (tidy set) *A* tidy set *is a trimmed, then thinned, simplicial complex.*

That is, we first delete all maximal leaves in the input simplicial complex. We then collapse all maximal acyclic simplices to get a simplicial set, as shown in Figure 3.

3.3 Minimality

The order in which we do reduction is important. Indeed, it may seem reasonable to try trimming a tidy set again, as we currently trim complex leaves only. The following theorem states, however, that collapses do not grow leaves.

Theorem 5 (minimal) *A tidy set has no leaves. That is, it is minimal with respect to trimming and thinning.*

PROOF. Recall Definition 1. For any $\sigma \in Q$, let $c \in H_n(\mathrm{Cl}\,\sigma)$ be a nontrivial homology class for some n. We know that this class exists as otherwise, σ would be acyclic and collapsed, $\sigma \in C$, but we know $Q \cap C = \emptyset$. Initially, however, σ was acyclic as a simplex within the input simplicial complex. So, c developed within $\mathrm{Cl}\,\sigma$ due to the collapse of a neighboring simplex. But this development was through σ's intersection with the neighbor and this intersection was also collapsed, so $c \notin H_n(\mathrm{Cl}\,\sigma \cap \mathrm{Cl}\,(X - \mathrm{Cl}\,\sigma))$. Therefore, σ is not a leaf. □

Because of the theorem above, a tidy set is a minimal model for representing a simplicial complex. It is not unique as it depends on the ordering of the sequence of deletions and collapses.

4. ALGORITHMS

In this section, we describe algorithms for computing homology by first constructing a tidy set. We begin with a quick look at data structures as well as algorithms for computing homology. We then describe algorithms for acquiring the minimal representations for simplicial complexes. For reduction, we give a recursive algorithm for trimming, and a two-phase algorithm for thinning.

4.1 Basic Blocks

We begin with a brief discussion on data structures for computing homology with simplicial sets. A crucial aspect of our approach is having two representations: For most of the computation, we represent a simplicial set $\mathcal{X}(Q, C)$ using the tuple (Q, C). But when required, we enumerate the simplices as a set. We represent each simplex or clique as a set of vertices, with the boundary operator being implicit. This implicit representation means that we need to maintain vertex identifications that may result from simplicial collapses. The vertices form *disjoint sets*, so we utilize a *union-find* data structure [7]. To determine whether a simplex σ is degenerate, we check to see if σ is a subset of any set in C, and we only keep non-degenerate faces during enumeration. Consequently, a simplicial set, unlike a simplicial complex, is not closed in our representation. However, this design decision allows our sets to remain small, a principle goal of our paper.

An important structure is the dual graph, which captures the intersection pattern of the maximal simplices.

Definition 6 (dual graph) *Let Q be a set of maximal sets. The* dual graph *is $G = (Q, E)$, where $E = \{\{\sigma, \tau\} \mid \sigma \cap \tau \neq \emptyset, \sigma \neq \tau \in Q\}$.*

For sparse complexes, such as clique complexes of sparse graphs, we represent the dual complex explicitly for quick access to neighbors of simplices. For denser complexes, however, explicit construction is computationally infeasible and we must construct the graph locally as needed. For the rest of the section, we assume the procedure DUAL-GRAPH(Q) returns the dual graph for a set of maximal sets Q.

For reduction, we need to compute homology, and we may do so by using the reduction algorithm [23]. However, placing any arbitrary filtration on a simplicial set turns it into a *based persistence complex* [34], allowing us to use the persistence algorithm over fields. Below, we assume that the procedure BETTI(X) returns the reduced Betti numbers of a given simplicial set X in a vector by first computing homology over $\mathbb{Z}_2$ coefficients using the generalized persistence algorithm [34]. We use the same implementation of the algorithm for computing homology of simplicial complexes.

While a clique complex may have high dimension, a space does not have non-trivial homology in dimensions higher than its embedding dimension [15]. So, for geometric simplicial sets embedded in $\mathbb{R}^d$, we only require the d-skeleton in computing homology. Similarly, in many applications, we only need the ith homology, which requires the $(i+1)$th skeleton. Therefore, in practice, we parameterize our algorithms with the maximum dimension of the skeleton we are interested in. In the discussion below, we hide this parameter to simplify presentation.

4.2 Maximal Simplices

Given a simplicial complex K, we need to compute its minimum representation $\mathcal{Q}(K)$, its maximal simplices, which is equivalent to maximal sets in K. Yellin [30] gives an output-sensitive iterative algorithm for this problem with complexity $O(mn)$, where m is the number of maximal sets and $n = \sum_{\sigma \in K} |\sigma|$ is the complexity of K. As described, the algorithm requires a prohibitive $\Theta(|K|^2)$ additional space for storing an array of counters. We observe, however, that these counters are used locally in each iteration of the algorithm. This observation allows us to reduce the space complexity to $\Theta(\dim K)$, making the algorithm feasible for large complexes. We define MAXIMAL-SETS(K) to be the function that returns the set of maximal simplices in K.

For clique complexes, we may compute our representation directly from the input graph G as maximal simplices are maximal cliques. This correspondence reduces our problem to enumerating maximal cliques of a graph. An n-vertex graph may have up to $3^{n/2}$ maximal cliques in the worst case [21], so this problem is hard in general. There have been a number of algorithms based on greedy or output sensitive approaches. We define MAXIMAL-CLIQUES(G) to be the function that computes the set of maximal cliques in G using the IK-GX algorithm [3]. We emphasize that we do not have to construct the complex $\mathcal{C}(G)$ to find its minimal representation $\mathcal{Q}(\mathcal{C}(G))$.

4.3 Greedy Reduction

We employ an iterative scheme for both trimming and thinning the complex. Our approach is greedy as a simplex has exponential number of faces, so eliminating large simplices has enormous payoff. Therefore, we always maintain sets of simplices in decreasing order of size. We define GREEDY($Q, C,$ *Is-Reducible*, *Reduce*) to be the procedure that reduces the simplicial set denoted (Q, C) by the reduction technique specified by predicate *Is-Reducible* and action *Reduce*.

```
GREEDY(Q, C, Is-Reducible, Reduce)
1  P ← Q
2  while P ≠ ∅                                   ▷ Potential
3        do R ← ∅                                ▷ Reduced
4           foreach σ ∈ P
5                 do if Is-Reducible(σ, Q, C)
6                       then (Q, C) ← Reduce(σ, Q, C)
7                            R ← R ∪ {σ}
8           P ← NEIGHBORS(R, Q)
9  return (Q, C)
```

The algorithm maintains a set P of potentially reducible simplices, initializing it to Q. In each round, the algorithm collects reduced simplices in set and uses their neighbors in Q as candidates for the next round. We use this scheme, along with another greedy thinning algorithm, DFS-THIN, to define the procedure TIDY-SET(Q), which reduces a simplicial complex with maximal simplices Q and returns the tidy set as a tuple (Q, C).

```
TIDY-SET(Q)
1  (Q, C) ← GREEDY(Q, ∅, IS-LEAF, TRIM)
2  (Q, C) ← DFS-THIN(Q, C)
3  (Q, C) ← GREEDY(Q, C, IS-ACYCLIC, THIN)
4  return (Q, C)
```

4.4 Recursive Trimming

To trim using the greedy scheme, we define the action TRIM and the predicate IS-LEAF. By Theorem 5, we only need to trim leaves in the simplicial complex, so we use the definition of complex leaves in Theorem 1. For this reason, TRIM follows Definition 4, and $C = \emptyset$ is not used below.

```
TRIM(σ, Q, C)
1  return (Q − {σ}, C)
```

```
IS-LEAF(σ, Q, C)
 1  I ← ⋃_{τ∈Q−{σ}} (σ ∩ τ)                          ▷ Stem
 2  M ← MAXIMAL-SETS(I)
 3  if |M| = 1                                       ▷ Single set
 4     then return TRUE
 5  elseif (max_{τ∈M} dim τ) > MAX-DIM
 6     then (Q_M, C_M) ← TIDY-SET(M)                 ▷ Recurse
 7          X_M ← X(Q_M, C_M)
 8          if χ(X_M) = 1
 9             then return BETTI(X_M) = 0
10             else return FALSE
11     else K_M ← Cl M                               ▷ Direct
12          if χ(K_M) = 1
13             then return BETTI(K_M) = 0
14             else return FALSE
```

To determine if σ is a leaf in the simplicial complex, the predicate IS-LEAF needs to determine if the intersection I of σ and the rest of the complex is acyclic. The procedure computes I and represents it with maximal simplices M by using MAXIMAL-SETS from Section 4.2. If the intersection has only one maximal set, it immediately returns true as a maximal set corresponds to a simplex which is acyclic in a complex. If the intersection has high dimension, it recurses by using the procedure TIDY-SET as it now has a smaller instance of the original problem. It then enumerates the resulting simplicial set and checks if its reduced Betti numbers are all zero. Since both TIDY-SET and IS-LEAF call each other, the two procedures become mutually recursive. Otherwise, when the intersection has low dimension, it directly computes homology by enumerating the simplicial complex. In both cases, it uses the Euler characteristic to skip homology computation whenever possible. Based on our experiments, we currently set MAX-DIM to 5. It is important to emphasize that the minimal representation is key to the efficiency of our trimming procedure: It allows both simple acyclicity testing as well as recursive evaluation, using direct homology testing of small complexes as a base case.

4.5 Two-Phase Thinning

We thin the trimmed complex in two phases, corresponding to complex and set thinning, respectively. Within a complex, all simplices are acyclic, but collapsing any simplex may cause a neighboring simplex to have non-trivial homology. For this reason, we attempt to find a large set of non-neighboring simplices that we may collapse at once. This idea may remind the reader of an *independent set*, a set of vertices in a graph that are pairwise non-adjacent. Indeed, vertices in an independent set of the dual graph correspond to non-intersecting maximal simplices. We may collect a larger set, however.

Theorem 6 *A simplex with one collapsed neighbor remains acyclic.*

PROOF. Two simplices intersect along a shared face. If we collapse one of the simplices, the shared face is also collapsed, and this does not cause the other simplex to develop non-trivial homology. □

Given this observation, we search the dual graph using *depth-first-search (DFS)* [7] to collect a set of acyclic simplices. DFS-THIN reduces a simplicial set denoted by (Q, C) by searching the dual graph and collecting acyclic simplices using THIN-DFS-VISITOR upon node discovery. We use the *algorithm visitor* paradigm [27] along with a generic version of DFS and define action THIN exactly as in Definition 4. Note that this is a fast and simple procedure as it does not require either face enumeration or homology computation.

```
THIN(σ, Q, C)
1  return (Q − {σ}, C ∪ {σ})
```

```
DFS-THIN(Q, C)
1  return DFS(DUAL-GRAPH(Q), THIN-DFS-VISITOR)
```

```
THIN-DFS-VISITOR(σ, Q, C)
1  N ← NEIGHBORS(σ, C)                  ▷ Collapsed neighbors
2  if |N| ≤ 1
3     then return THIN(σ, Q, C)
```

Having thinned the complex, we move into the category of simplicial sets. To thin using the greedy scheme, we define the predicate IS-ACYCLIC, having defined action THIN already. The procedure now needs to enumerate the full simplicial set. As with trimming, we attempt to avoid computing homology by using the Euler characteristic.

```
IS-ACYCLIC(σ, Q, C)
1  X_σ ← X({σ}, C)
2  if χ(X_σ) = 1
3     then return BETTI(X_σ) = 0
4     else return FALSE
```

5. EXPERIMENTS

In this section, we describe an implementation of our algorithms, examine its performance on real and synthetic data, and compare it with existing software. Our implementation is in generic C++ and part of a library we are currently developing for computational topology. All our timings are done on a 64-bit GNU/Linux machine with two dual-core 3

S	$\dim S$	$\lvert S\rvert$	ϵ	$G_\epsilon = (S, E)$ time	$\lvert E\rvert$	$\mathcal{C}(G_\epsilon)$ time	size	β time	total time
G	3	318	5.00	0.00	3,960	0.11	71,032	0.12	0.23
M	8	10,000	0.13	0.22	36,263	48.74	13,234,966	126.01	174.97
B	3	34,837	0.05	0.38	489,876	19.29	9,714,912	26.35	46.02
S	8	50,000	0.18	2.17	546,388	49.25	19,134,612	76.66	128.08
D	3	88,571	0.0014	1.08	543,996	168.40	45,995,489	297.71	467.19

Table 1: Data sets and statistics on computing the VR graph G_ϵ at scale ϵ, the complex $\mathcal{C}(G_\epsilon)$, and computing its homology. All times are in seconds.

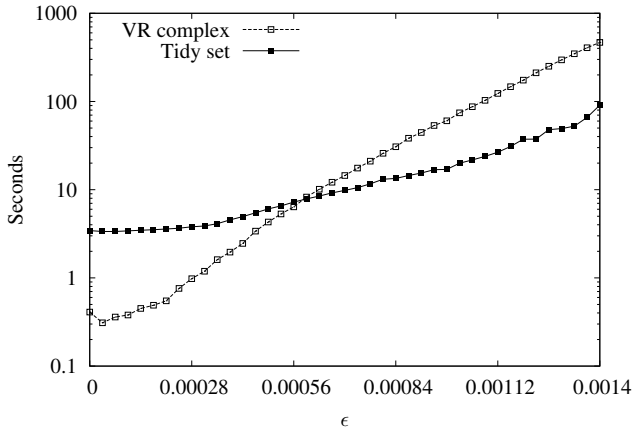


Figure 5: Homology time using the VR complex versus the tidy set at scale ϵ for dataset D.

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. G is Gramicidin A, a small protein. M is a portion of the *van Hateren-Mumford dataset* with parameters $k = 30$ and cut $= 20\%$ [2]. B and D are points sampled from the surface of the *Stanford bunny* and *dragon*, respectively. S is sampled from a unit 3-sphere, diagonally embedded in $\mathbb{R}^8$. The standard method for computing homology involves constructing the VR complex for which we give timing and statistics in Table 1. We use the Ann library to build the VR graph G_ϵ, and modify algorithm Incremental-VR [32] to build a non-filtered VR complex $\mathcal{C}(G_\epsilon)$ in the ambient dimension, e.g. an 8-dimensional complex for S. Finally, we compute Betti numbers β using the persistence algorithm [34].

5.2 Timing

We build the tidy set directly from the graph G_ϵ. Figure 5 shows the performance of our algorithm for dataset D across scale. We showed the size of the resulting tidy set in Figure 1. Figure 6 compares our method for the 8-dimensional dataset M. Note that the complex is just beginning to connect and develop topological attributes at the scales we are examining. Table 2 lists the times for all phases of our method at the maximum scale for each dataset. We observe that our algorithm is generally much faster than computing with the VR complex, and competitive for smaller sets.

S	$\mathcal{Q}$	dual	trim	dfs	thin	$\mathcal{X}$	β	total
G	0.03	0.01	0.13	0.00	0.00	0.00	0.00	0.17
M	0.34	0.60	3.81	0.00	1.75	0.14	0.20	7.06
B	8.45	3.95	25.22	0.01	0.01	0.00	0.00	38.02
S	8.03	3.45	32.43	0.08	17.39	3.19	4.16	70.09
D	37.63	15.36	33.12	0.04	3.53	0.11	0.11	90.98

Table 2: Time, in seconds, for computing cliques $\mathcal{Q}$ of the VR graph G_ϵ, the dual graph, trimming, thinning using DFS and greedy scheme, enumerating the tidy set, and computing homology. The total includes time for computing G_ϵ from Table 1.

Moreover, our method requires much less memory. For example, computing the VR complex and homology for D requires 5 GB and 10 GB of memory, respectively, while our method only needs 500 MB, a factor of 20 times less.

5.3 Statistics

Table 3 gives statistics on the number and size of cliques in the VR graph, the efficacy of our trimming and two thinning procedures, and the size of the resulting tidy set. We generally have a great number of cliques, including large ones, such as a 73-clique for D that generates $\binom{73}{4}$ or roughly 1M tetrahedra in the VR complex. Our methods are highly effective in reducing cliques. For G, we are left with a single 9-clique that generates 264 simplices, as compared to 71K simplices in the VR complex. For B, we have a tidy set that has a factor of 9,000 fewer simplices. The tidy set for S has nearly 21K cliques, although its topological complexity ($\sum_i |\beta_i|$) is 2,223. We hope to examine such hard cases to look for further reduction.

In general, the effectiveness of reduction techniques vary across scale, as illustrated by Figure 7 for G. The techniques, however, are complementary, removing nearly 90% of maximal cliques at all scales. Initially, the graph is just a set of points, which DFS "contracts" trivially. With increasing scale, as the graph becomes denser, the complex gains thickness, allowing us to trim from the boundary inward. What is intriguing is that the efficacy of the methods as displayed in in the figure reflects the underlying topology of the point set: The first local minimum for trimming is near 2 Angstroms, slightly larger than radii of atoms, when atoms connect and form a large number of 1-cycles.

We end by comparing our methods with *JPlex* [26] the only publicly available software that implements VR complexes and computes homology using the persistence algorithm. Figure 8 compares total computation time for B, our largest dataset that JPlex is able to process. At the highest scale, our methods run about 13 times faster and requires 5

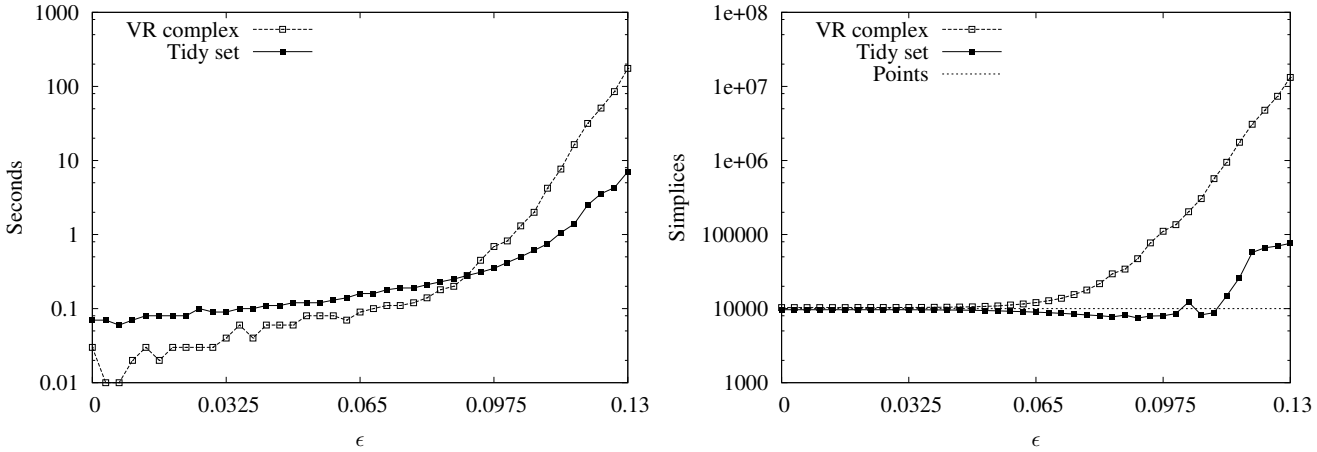


Figure 6: Homology times and sizes of the VR complex and tidy sets for dataset M.

S	cliques Ω			− trim	− dfs	− thin	= tidy	$\lvert\mathcal{X}\rvert$
	ave	max	num					
G	8.97	13	870	853	8	8	1	246
M	5.31	22	16,114	10,626	3,702	644	1,142	76,008
B	9.40	16	140,052	139,679	358	9	6	1,104
S	6.56	18	167,742	129,835	7,986	8,938	20,983	1,477,416
D	6.91	73	123,091	87,336	25,773	3,079	6,903	51,448

Table 3: Average and maximum clique size, and number of cliques, number removed by trimming using DFS and greedy scheme, remaining cliques in the tidy set, and its enumerated size.

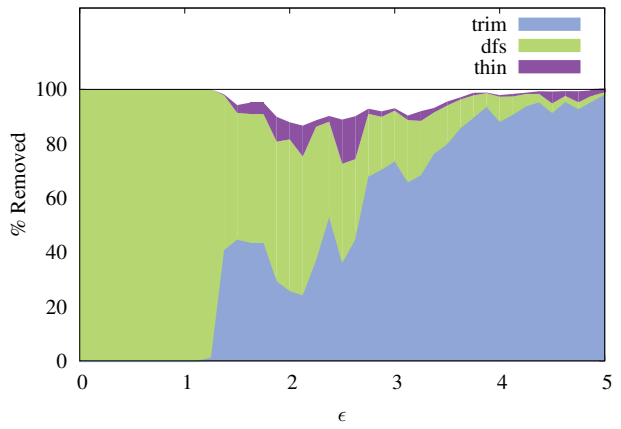


Figure 7: Cumulative graphs showing percentage of cliques removed by our three reduction techniques: trimming, DFS thinning, and greedy thinning for dataset G.

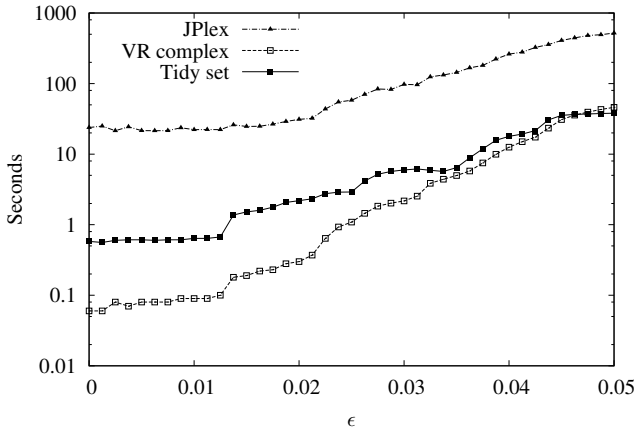


Figure 8: Comparison of our methods with JPlex for dataset B.

times less memory (4.7 GB vs. 958 MB) even though JPlex is optimized for simplicial complexes in low dimensions.

6. CONCLUSION

In this paper, we define the tidy set, a minimal simplicial set, for computing homology of clique complexes. We give recursive algorithms for computing the tidy set, implement them, and experimentally verify their effectiveness by computing homology of large datasets. Our software enables a number of interesting applications. For instance, we may compute geometric descriptions of homological cycles of tidy sets and expand them back onto the original point set to identify regions with nontrivial homology.

There are a number of rich avenues for research. While we focus on clique complexes in this paper, our work applies to arbitrary simplicial complexes, provided we compute maxi-

mal simplices efficiently. We also have not applied our methods toward computing witness complexes: We may now use large sets of landmarks for massive datasets. Dense graphs behave very differently from sparse graphs, and we are working on algorithms for computing their clique homology. We plan to extend our methods to filtered complexes to enable computation of persistence barcodes. Finally, almost every step of our method is parallelizable by design.

Acknowledgments

The author would like to thank Gunnar Carlsson for initial discussions and support.

7. REFERENCES

[1] Barmak, J., and Minian, E. Strong homotopy types, nerves and collapses, 2009. arXiv:0907.2954v1 [math.GT].

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

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

[4] CHomP: Computational Homology Project. `http://chomp.rutgers.edu`.

[5] Civan, Y., and Yalçin, E. Linear colorings of simplicial complexes and collapsing. *Journal of Combinatorial Theory Series A 114*, 7 (2007), 1315–1331.

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

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

[8] Curtis, E. B. Simplicial homotopy theory. *Advances in Mathematics 6* (1971), 107–209.

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

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

[11] Eilenberg, S., and Zilber, J. A. Semi-simplicial complexes and singular homology. *Annals of Mathematics 51*, 3 (1950), 499–513.

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

[13] Goerss, P. G., and Jardine, J. F. *Simplicial homotopy theory*, vol. 174 of *Progress in Mathematics*. Birkhäuser-Verlag, Basel, Switzerland, 1999.

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

[15] Hatcher, A. *Algebraic Topology*. Cambridge University Press, New York, NY, 2002. `http://www.math.cornell.edu/~hatcher/AT/ATpage.html`.

[16] Kaczynski, T., Mischaikow, K., and Mrozek, M. *Computational Homology*. Springer-Verlag, New York, NY, 2004.

[17] Kasson, P. M., Zomorodian, A., Park, S., Singhal, N., Guibas, L. J., and Pande, V. S. Persistent voids: a new structural metric for membrane fusion. *Bioinformatics 23*, 14 (2007), 1753–1759.

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

[19] Matoušek, J. LC reductions yield isomorphic simplicial complexes. *Contributions to Discrete Mathematics 3*, 2 (2008), 37–39.

[20] May, J. P. *Simplicial Objects in Algebraic Topology*. D. Van Nostrand Co., Inc., Princeton, NJ, 1967.

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

[22] Mrozek, M., Pilarczyk, P., and Żelazna, N. Homology algorithm based on acyclic subspace. *Computers and Mathematics with Applications 55*, 11 (2008), 2395–2412.

[23] Munkres, J. R. *Elements of Algebraic Topology*. Addison-Wesley, Reading, MA, 1984.

[24] Perry, P. Application of simplicial sets to computational topology. Senior thesis, Stanford University, 2003.

[25] Pilarczyk, P. Computer assisted method for proving existence of periodic orbits. *Topological Methods in Nonlinear Analysis 13* (1999), 365–377.

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

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

[28] Storjohann, A. Near optimal algorithms for computing Smith normal forms of integer matrices. In *Proc. International Conference on Symbolic and Algebraic Computation* (1996), pp. 267–274.

[29] Whitehead, J. H. C. Simplicial spaces, nuclei, and m-groups. *Proceedings of the London Mathematical Society s2-45*, 1 (1939), 243–327.

[30] Yellin, D. M. Algorithms for subset testing and finding maximal sets. In *Proc. ACM-SIAM Symposium on Discrete Algorithms* (1992), pp. 386–392.

[31] Zomorodian, A. Computational topology. In *Algorithms and Theory of Computation Handbook*, M. Atallah and M. Blanton, Eds., second ed., vol. 2. Chapman & Hall/CRC Press, Boca Raton, FL, 2010, ch. 3.

[32] Zomorodian, A. Fast construction of the Vietoris-Rips complex. *Computers & Graphics* (2010). (To appear).

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

[34] Zomorodian, A., and Carlsson, G. Localized homology. *Computational Geometry: Theory & Applications 41*, 3 (2008), 126–148.