An iterative algorithm for homology computation on simplicial shapes

Dobrina Boltcheva, David Canino, Sara Merino Aceituno, Jean-Claude Léon, de Floriani Leila, Franck Hétroy

An iterative algorithm for homology computation on simplicial shapes

Dobrina Boltcheva[a,*], David Canino[b], Sara Merino Aceituno[a], Jean-Claude Léon[a], Leila De Floriani[b], Franck Hétroy[a]

[a] *Grenoble University, Laboratoire Jean Kuntzmann, INRIA, 655, av. de l'Europe, 38334, Montbonnot, France*
[b] *University of Genova, Department of Computer Science, Via Dodecaneso, 35, 16146, Genova, Italy*

Abstract

We propose a new iterative algorithm for computing the homology of arbitrary shapes discretized through simplicial complexes, We demonstrate how the simplicial homology of a shape can be effectively expressed in terms of the homology of its sub-components. The proposed algorithm retrieves the complete homological information of an input shape including the Betti numbers, the torsion coefficients and the representative homology generators.

To the best of our knowledge, this is the first algorithm based on the *constructive* Mayer-Vietoris sequence, which relates the homology of a topological space to the homologies of its sub-spaces, i.e. the sub-components of the input shape and their intersections. We demonstrate the validity of our approach through a specific shape decomposition, based only on topological properties, which minimizes the size of the intersections between the sub-components and increases the efficiency of the algorithm.

Keywords: computational topology; simplicial complexes; shape decomposition; $\mathbb{Z}$-homology; Mayer-Vietoris sequence; generators

1. Introduction

Recently, the problem of computing the topological features of a shape has drawn much attention because of its applications in several disciplines, including shape analysis and understanding, shape retrieval, and finite element analysis [15, 20, 38]. Unlike geometric features (such as curvature) which are only invariant under rigid transformations, topological features are invariant under continuous deformations. Thus, they provide global quantitative and qualitative information about a shape, such as the number of its connected components, the number of holes and tunnels. Topological features are the core descriptors to extend geometric modelers with non-manifold shapes processing. For instance, the generation of simulation models still lacks capabilities for processing non-manifold shapes, like idealized representations [36]. Homological information on arbitrary shapes can strongly support new modeling capabilities, because constructive modeling techniques are often used. Also, topological features are especially important in high dimensional data analysis, where pure geometric tools are usually not sufficient.

The most common way to discretize a shape is through a simplicial complex. Simplicial complexes are easy to construct and manipulate, and compact data structures have been developed to encode them efficiently [7]. Simplicial homology is one of the most useful and algorithmically computable topological invariants. It characterizes a simplicial complex of dimension n through the notion of *homological descriptor*. Homological descriptors are defined in any dimension $k \leq n$ and are related to the non-trivial k-cycles in the complex which have intuitive geometrical interpretations up to dimension 2. In dimension 0, they are related to the connected components of the complex, in dimension 1, to the tunnels and the holes, and in dimension 2, to the shells surrounding voids or cavities.

Here, we propose a new algorithmic approach for homology computation on arbitrary shapes represented by finite simplicial complexes. Our framework is based on the *constructive homology theory* discussed in [32, 34, 33]. It provides a tool, the *constructive Mayer-Vietoris sequence*, which offers an elegant way for computing the homology of a simplicial complex from the homology of its sub-complexes and of their intersections. This leads to a *modular* algorithm for homology computation, that we call *Mayer-Vietoris (MV) algorithm*. Here, we show that our algorithm is more efficient than the classical one based on the reduction of the incidence matrices to a canonical form, known as the *Smith Normal Form (SNF)* [1, 30].

We demonstrate the validity of our approach through a decomposition of an n-dimensional simplicial complex, called the *Manifold-Connected (MC) decomposition* and proposed in [26] for 2-dimensional simplicial complexes. In our experiments, we demonstrate that the MC decomposition minimizes the size of the intersection between sub-complexes, and, thus, it is especially useful for the homol-

*Corresponding author
Email addresses: dobrina.boltcheva@inrialpes.fr (Dobrina Boltcheva), canino@disi.unige.it (David Canino), sm851@cam.ac.uk (Sara Merino Aceituno), jean-claude.leon@grenoble-inp.fr (Jean-Claude Léon), deflo@disi.unige.it (Leila De Floriani), franck.hetroy@grenoble-inp.fr (Franck Hétroy)

ogy computation in our constructive approach.

The remainder of the paper is organized as follows. In Section 2 we review some background notions on simplicial homology. In Section 3, we discuss related work on homology computation. In Section 4, we describe the MC decomposition, a graph-based data structure for encoding it and an algorithm for computing it. In Section 5, we introduce the main concepts from constructive homology, and, in Section 6, we describe the Homological Smith Reduction, the key tool for our MV algorithm. In Section 7, we provide a detailed description of the MV algorithm. In Section 8, we present experimental results based on our implementation of the homology computation algorithm. Finally, in Section 9, we draw some concluding remarks.

2. Background Notions on Simplicial Homology

Simplicial homology exploits the combinatorial structure of simplicial complexes and reformulates the homological problem into an algebraic one. In this section, we introduce some basic notions on simplicial homology needed throughout the paper. See [1, 30, 25] for more details.

Simplicial complexes. A simplex $\sigma = [v_0, \ldots, v_k]$ is the convex hull of a set V of affinely independent points in $\mathbb{R}^N$: here, k is the *dimension* of σ, which is called a $k-simplex$. For every non-empty subset $T \subseteq V$, the simplex σ' spanned by T is called a *face* of σ, and σ' is a *proper face* of σ if T is a proper subset of V. A *simplicial complex* X is a collection of simplices such that all the faces of any simplex in X are also in X and the intersection of two simplices is either empty or a face of both. The largest dimension of any simplex in X is the *dimension* of X, denoted as $dim(X)$. A subset Y of a simplicial complex X is a *subcomplex* of X if Y is itself a simplicial complex. Each k-simplex of a simplicial complex X can be *oriented* by assigning a *linear ordering* to its vertices. The *boundary* of an oriented k-simplex is defined as the alternate sum of its incident (k-1)-simplices: $d_k([v_0, \ldots, v_k]) = \sum_{i=0}^{k}(-1)^i[v_0, \ldots, v_{i-1}, v_{i+1}, \ldots, v_k]$. A fundamental property of the boundary operator is that the boundary of every boundary is null, $d_{k-1}d_k = 0$, for all $k > 0$.

Chain-complexes. Given an *oriented simplicial* complex X, simplicial homology builds an algebraic object $C_*(X)$, called *chain-complex*, on which the homological problem for X is resolved using linear algebra.

Let $n = dim(X)$. A *k-chain* is defined for each dimension $k \in [0, \ldots, n]$ as $a^k = \sum \lambda_i \sigma_i^k$, where $\lambda_i \in \mathbb{Z}$ are the coefficient assigned to each k-simplex σ_i^k. The k^{th} *chain group*, denoted as $C_k(X)$, is formed by the set of k-chains together with the addition operation, defined by adding the coefficients simplex by simplex. There is a chain group for every integer k, but for a complex in $\mathbb{R}^n$, only the groups for $0 \leq k \leq n$ may not be trivial. These chain groups are Abelian and finitely generated, thus, the set of *oriented* k-simplices forms a basis for $C_k(X)$. In the following, we will refer to this basis as the *canonical* basis.

Each boundary operator d_k can be linearly extended to k-chains, $a^k = \sum \lambda_i \sigma_i^k$, as sum of the boundaries of the simplices in the chain: $d_k(a^k) = \sum \lambda_i d_k(\sigma_i^k)$. The *chain-complex*, denoted as $C_*(X) = (C_k, d_k)_{k \in \mathbb{N}}$, is the sequence of the chain groups $C_k(X)$ connected by the boundary operator d_k:

$$(C_*, d_*) : 0 \xleftarrow{0} C_0 \xleftarrow{d_1} C_1 \xleftarrow{d_2} \ldots \xleftarrow{d_{n-1}} C_{n-1} \xleftarrow{d_n} C_n \xleftarrow{0} 0$$

The chain-complex $C_*(X)$, associated with a simplicial complex of finite dimension n, can be encoded as a pair (B_k, D_k) for each $0 \leq k \leq n$, where $B_k = [\sigma_0^k, \ldots, \sigma_l^k]$ is the canonical basis of C_k and $D_k = [\eta_{j,i}^k]$ is an integer matrix, called the *incidence matrix*, which expresses the boundary operator with respect to B_{k-1} and B_k:

$$\eta_{j,i}^k = \begin{cases} 0 & \text{if } \sigma_j^{k-1} \text{ is not in the boundary of } \sigma_i^k; \\ 1 & \text{if } \sigma_j^{k-1} \text{ is in the boundary of } \sigma_i^k; \\ -1 & \text{if } -\sigma_j^{k-1} \text{ is in the boundary of } \sigma_i^k. \end{cases}$$

Homology groups. Given a chain-complex $C_*(X)$, homology groups are derived from two specific subgroups of the chain groups defined by the boundary operators:

$$Z_k = \ker d_k = \{c \in C_k \mid d_k(c) = 0\}$$

$$B_k = img\ d_{k+1} = \{c \in C_k \mid \exists a \in C_{k+1} : c = d_{k+1}(a)\}$$

We say that a k-cycle in Z_k *bounds* if it is also in B_k. Two cycles are *homologous* if they differ by a cycle that bounds. The collections Z_k of k-cycles and B_k of k-boundaries form subgroups in C_k: $B_k \subseteq Z_k \subseteq C_k$. For each $k \in [0, n]$, the k^{th} *homology group* of X is defined as the quotient of the cycle group over the boundary group, i.e., $H_k = Z_k/B_k$. Thus, the elements of the homology group are equivalence classes of k-cycles which are not k-boundaries. Since $C_k(X)$ is an Abelian group, then it is isomorphic to:

$$H_k(X) = \overbrace{\mathbb{Z} \oplus \ldots \oplus \mathbb{Z}}^{\text{free group}} \oplus \overbrace{\mathbb{Z}/\lambda_1\mathbb{Z} \oplus \ldots \oplus \mathbb{Z}/\lambda_p\mathbb{Z}}^{\text{torsion group}}$$

The number of occurrences of $\mathbb{Z}$ in the free part is the number of elements of H_k with infinite order and is called the k^{th} *Betti number*, β_k. It can also be seen as the maximal number of independent k-cycles that do not bound. The values $\lambda_1, \ldots, \lambda_p$ satisfy two conditions: (i) $\lambda_i \geq 2$ and (ii) λ_i divides λ_{i+1}, for each $i \in [1, p)$. They correspond to the *torsion coefficients*. A set of homologous k-cycles can be associated with each group $\mathbb{Z}/\lambda_i\mathbb{Z}$ of H_k. These k-cycles are not the boundary of any $(k+1)$-chain, but if taken λ_i times, they become the boundary of any $(k+1)$-chain. We will call these cycles *weak-boundaries*.

For all k, there exists a finite number of elements of H_k from which we can deduce all elements of H_k. Let H_k be a homology group generated by q independent equivalence classes $C_1 \ldots C_q$, then any set $\{g_1, \ldots, g_q | g_1 \in$

$C_1, \ldots, g_q \in C_q\}$ is called a set of *generators* for H_k. We can denote a homology group in terms of its generators as $H_k = [g_1, \ldots, g_q]$. We refer the complete homology information of a simplicial complex X (generators, Betti numbers and torsion generators) as the $\mathbb{Z}$-*homology* of X.

Mayer-Vietoris sequence. The Mayer-Vietoris sequence is an algebraic tool which allows us to study the homology of a space X by splitting it into two subspaces A and B such that $A \cap B \neq \emptyset$, for which the homology groups are easier to compute. This sequence relates the chain-complex of the union $(A \cup B)_*$ to the chain-complexes of the disjoint sum $A_* \oplus B_*$ and the intersection $(A \cap B)_*$:

$$0 \xleftarrow{0} (A \cup B)_* \xleftarrow{j} (A \oplus B)_* \xleftarrow{i} (A \cap B)_* \xleftarrow{0} 0$$

This sequence is *exact*, i.e., the kernel of each homomorphism is equal to the image of the previous one, $Img(i) = Ker(j)$. Therefore, we have the following relations: $(A \cap B)_* \cong Ker(j)$ and $(A \cup B)_* \cong (A \oplus B)_* / Img(i)$.

As demonstrated in [30], we can build a long exact sequence of their homology groups, starting from the short exact sequence of the chain-complexes:

$$\ldots \longleftarrow H_{k-1}((A \cap B)_*) \xleftarrow{\partial} H_k((A \cup B)_*) \xleftarrow{j} H_k((A \oplus B)_*) \xleftarrow{i} H_k((A \cap B)_*) \xleftarrow{\partial} H_{k+1}((A \cap B)_*) \longleftarrow \ldots$$

In some cases, the homology of the union can be deduced from this long exact sequence of homology groups, but it is not always possible to decide. This problem is known as the *extension problem*. Moreover, there is no way to give the generators of the homology group, because this method is *non-constructive* [34]. Thus, *classical* Mayer-Vietoris sequence is known as a purely theoretical tool and is useful only for computations by hand.

Smith Normal Form (SNF) algorithm. For each k such that $1 \leq k \leq n$, the SNF algorithm transforms the incidence matrix D_k into its Smith normal form N_k [30]:

$$N_k = \begin{array}{c} \\ s_0^{k-1} \\ \vdots \\ s_p^{k-1} \end{array} \begin{array}{c} \begin{array}{ccc} s_0^k & \ldots & s_l^k \end{array} \\ \begin{pmatrix} 0 & \boldsymbol{\lambda} & 0 \\ 0 & 0 & Id \\ 0 & 0 & 0 \end{pmatrix} \end{array}$$

where $\boldsymbol{\lambda}$ is a diagonal matrix with $\lambda_r, \ldots, \lambda_1 \in \mathbb{Z}$ in the diagonal such as $\lambda_i > 1$ and λ_i divides λ_{i+1}, $\forall i \in [1, r)$. Each incidence matrix D_k is initially expressed into the *canonical* basis β_c^{k-1} and β_c^k of the chain groups C_{k-1} and C_k. At the end of the algorithm, matrix N_k is expressed into different basis β_s^{k-1} and β_s^k, called the *Smith basis*.

The input incidence matrix can be expressed as $D_k = P_{k-1} N_k P_k^{-1}$, where matrix P_k encodes the basis changes $P_k : C_k[\beta_s] \rightarrow C_k[\beta_c]$. Initially, $P_k = Id$, but during the transformation every elementary operation on the rows and the columns of the boundary matrix D_k is translated into an operation on matrix P_k which encodes the change of the basis. Thus, P_k tell us how to express an element of the *Smith basis* in terms of the *canonical basis* of C_k.

The homology is computed using two consecutive incidence matrices in Smith Normal forms, denoted as N_k, and N_{k+1}. The rank of the sub-group $Z_k = \ker N_k$ is equal to the number of zero-columns of N_k, which correspond to the k-cycles. The rank of $B_k = \text{img } N_{k+1}$ is equal to the number of non-zero rows of N_{k+1}. The *generators*, expressed in the canonical basis, are obtained by computing the image of each generator γ_i from the *Smith basis* through the matrix P_k.

3. Related Work

The classical approach to compute the $\mathbb{Z}$-homology of a simplicial complex of finite dimension is based on the Smith Normal Form (SNF) algorithm [1, 30]. Although this method is theoretically valid in any dimension and for any kind of simplicial complex, it has some inherent limitations due to the size of the incidence matrices and to the high complexity of the reduction algorithm. The best available reduction algorithms have super-cubical complexity [35, 14] and they are suitable only for small simplicial complexes. Another well-known problem is the appearance of huge integers during the reduction [24].

In the literature several optimizations of the SNF algorithm have been developed. *Stochastic methods* [21] are efficient on sparse integer matrices, but they do not provide the homological generators. *Deterministic methods* [28, 35] perform the computations modulo an integer chosen by a determined criterion, but the information about the torsion coefficients is lost with this strategy. Another way to improve the computation time is to *reduce* the input complex without changing its topology by applying iterative simplifications, and by computing the homology when no more simplifications are possible. This *reduction approach* has been mainly investigated in the context of homology computation from 3D voxel images [27, 6, 29, 31]. Other reduction approaches apply discrete Morse theory [18] to homology computation since in many applied situations one expects the Morse complex built on the original simplicial complex to be much smaller than this latter.

Another approach for homology computation is based on *persistent homology* [15]. In this framework, the input simplicial complex is filtered (according to any real function) in order to study which homological attributes appear, disappear and are maintained through nesting. The pertinent information is encapsulated by a pairing of the critical values of the function which are visualized by points forming a diagram in the plane. Since the filtration is done by adding only one simplex at a time, it can be interpreted as a special case of the *Mayer-Vietoris sequence*. These methods are usually designed for simplicial complexes with dimensional restrictions in most cases. The original algorithm in [16] computes the pairs from an ordering of the simplices in a triangle mesh and exhibits a cubic worst-case time in the size of the complex. In [5], an algorithm that maintains the pairing in worst-case linear time per transposition in the ordering has been

presented. A nearly linear algorithm for computing only the Betti numbers (the ranks of the homology groups) for simplicial 3-complexes is proposed in [9]. In [10], an algorithm is proposed for computing the homological generators of manifold simplicial complexes, embedded in the 3D Euclidean space, which is then extended to arbitrary simplicial complexes through a thickening process. The algorithm presented in [11] extracts two types of possible 1-cycles which identify handles and tunnels on 2-manifold surfaces. In [23], another method has been proposed for computing the non-contractible 1-cycles on smooth compact 2-manifolds. The shape of the computed generators has been addressed in [39, 4]. However, the persistence of a feature depends highly on the chosen filtering function and it is still an open problem to find geometrically meaningful functions on non-manifold simplicial complexes. In [17], a first attempt for a Mayer-Vietoris formula for persistent homology has been proposed with an application to shape recognition in the presence of occlusions. However, this work is based on the classical version of the Mayer-Vietoris sequence, and the proposed formula cannot be used in practice since it does not lead to an algorithm.

Finally, there exist also a few methods based on *constructive homology*, introduced in [33], which provides an original *algorithmic* approach for computing homology. Concepts borrowed from constructive homology have been used in [2, 22] for homology computation on images. To the best of our knowledge, none of the existing algorithms uses the *constructive* Mayer-Vietoris sequence which provides an effective strategy for computing the homology generators of an arbitrary simplicial complex from the homology of its sub-complexes.

4. The Manifold-Connected Decomposition

In this section, we describe a decomposition of a simplicial complex, called the *Manifold-Connected (MC)* decomposition, which we use as the basis for performing homology computation. The MC decomposition has been introduced in [26] for two-dimensional simplicial complexes, but it can be defined in arbitrary dimensions.

Let us consider a d-dimensional regular simplicial complex X. Recall that a *regular (or dimensionally homogeneous)* simplicial complex is a complex in which all top simplices are d-dimensional, where a *top simplex* is a simplex which does not belong to the boundary of any other simplex in X. We introduce some definitions and concepts needed for the definition of manifold-connected complex and component.

A $(d-1)$-simplex τ in a regular simplicial d-complex X is said to be a *manifold* simplex if and only if there exists at most two d-simplices in X incident in τ. Two d-simplices σ and σ' in X are said to be *manifold-connected* if and only if there exists a path P joining σ and σ' formed by d-simplices such that any two consecutive d-simplices in P are adjacent through a manifold $(d-1)$-simplex. A regular d-complex in which every pair of n-simplices are manifold-connected is a *manifold-connected* complex.

Any combinatorial manifold, i.e., any simplicial complex with a manifold domain, is clearly manifold-connected, but the reverse is not true. Thus, the class of manifold-connected complexes is a decidable superset of the class of combinatorial manifolds. Note that the class of d-manifolds is not decidable for $d \geq 6$ [8].

The manifold-connectivity relation between the top d-simplices in a regular d-complex X defines an equivalence relation. The *manifold-connected* components of X are the equivalence classes of the top d-simplices of X with respect to the manifold-connectivity relation. The collection of all manifold connected components in X forms the *Manifold-Connected (MC)* decomposition. Any two or more components in the MC decomposition of a simplicial d-complex X may have a common intersection which is a sub-complex of X of dimension lower than d.

Any arbitrary (non-regular) simplicial n-complex Y is uniquely decomposed into a collection of maximal regular complexes Y_d formed by top simplices of the same dimension $d \leq n$. Figure 1 shows an example of the decomposition of an arbitrary simplicial 3-complex into maximal regular sub-complexes Y_2 and Y_3.

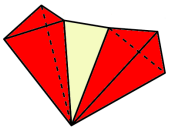


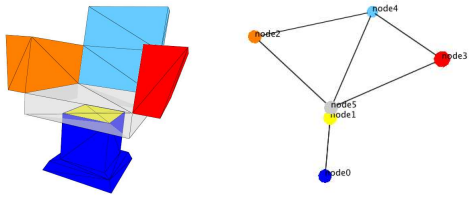
Figure 1: Decomposition of a 3-complex into maximal regular sub-complexes Y_2 (in yellow) and Y_3 (in red).

Thus, the MC decomposition of an n-complex Y is the collection of the MC decompositions of the maximal regular sub-complexes of Y. As a consequence, the MC decomposition of a complex Y is unique. Figure 2(a) shows an example of the MC decomposition of a regular simplicial 2-complex. The six MC-components of dimension 2 are connected through chains of non-manifold edges.

4.1. Encoding the MC decomposition

We have developed a representation for the MC decomposition suitable for homology computation. For this purpose, we need to efficiently access the intersection of pairs of MC-components, and to have a unique vertex ordering for all MC-components to be able compute the chain-complexes. Thus, we propose a graph-based data structure which encodes the MC decomposition as a graph, called the *Homology MC-graph (Homo-MC graph)*, denoted as $\mathcal{H} = (\mathcal{N}, \mathcal{A})$. A node in $\mathcal{N}$ corresponds to an MC-component, while an arc in $\mathcal{A}$ describes the intersection of *two* MC-components. Figure 2(b) shows the Homo-MC graph describing the MC decomposition in Figure 2(a).

The data structure based on the Homo MC-graph consists of two layers: the top layer is the encoding of the Homo MC-graph, while the second layer describes the simplicial complex Y and is currently implemented through the *Incidence Simplicial (IS)* data structure [7]. The IS data structure encodes all simplices in Y and allows a di-



(a) (b)

Figure 2: An example of a non-manifold shape (a) and a graphical representation of its Homo-MC graph. Each node of the Homo-MC graph is identified by a color and it is graphically represented by its center of gravity.

rect access to each simplex of Y in constant time. Moreover, it encodes the relations among a k-simplex and its bounding simplices of dimension $k-1$ and among a k-simplex and the simplexes of dimension $k+1$ in its coboundary. Each node C in the Homo-MC graph has a list of references to the top simplices of the corresponding complex in C. In this way, we ensure that all the MC-components have the same vertex ordering provided by Y. Similarly, each arc a of the Homo-MC graph contains a reference for each simplex belonging to the intersection described by a.

4.2. Building the Homology MC-Graph

The computation of the Homo-MC graph for an arbitrary simplicial n-complex Y consists of two steps: first, the MC-components are identified, and then the arcs of the Homo-MC graph are computed.

The detection of the MC-components is performed according to the following steps:

1. retrieve the maximal regular sub-complexes Y_k (with $k \leq n$) of Y formed by the top k-simplices in Y which are adjacent along $(k-1)$-simplices;
2. for each sub-complex Y_k, perform a traversal starting from an unvisited top k-simplex σ and retrieve all top k-simplices which are reachable from σ by visiting manifold $(k-1)$-simplices and their incident top k-simplices. All top k-simplices visited by starting from σ belong to the same k-dimensional MC-component, identified by an integer label C. Mark with C all sub-simplices of each top simplex in the MC-component;
3. create a node of the Homo-MC Graph for each MC-component.

At the end, each simplex σ in Y is marked with a list of integer labels $l_\sigma = \{C_1, \ldots, C_s\}$, which denote the MC-components containing σ. A simplex $\bar{\sigma}$ marked with several labels is a *singularity*. The arcs of the Homo-MC graph are then retrieved as follows:

1. for each non-manifold singularity $\bar{\sigma}$, generate all pairs of integer labels (C_i, C_j) in $l_{\bar{\sigma}}$ and store the tuples $(\bar{\sigma}, C_i, C_j)$ in an array A;
2. sort the tuples in A by using the lexicographic order of labels: tuples related to the same pair of labels are stored in consecutive locations in A;
3. generate an arc of the Homo-MC Graph for each unique pair of nodes identified at step 2.

5. Constructive Homology

Constructive Homology theory has been developed in order to solve the *non-constructiveness* of classical homology from its roots [32]. Within this framework, based on *constructive mathematics* [37], the homological concepts are reformulated into concepts with a computational nature, thus yielding to effective implementable algorithms. The theory has been developed to handle homology computations over chain-groups of infinite dimension [33, 34] and its validity has been proven using functional programming [13]. The Mayer Vietoris algorithm we present here is an application of constructive homology. In this section, we introduce the main concepts and theorems which are used in our MV algorithm. For more details, see also [3].

Our method is based on two key concepts: the *reduction*, which is a relation between two chain-complexes with equivalent homologies, and the *cone of a morphism*, which is a particular way to represent the morphism relating two chain-complexes as a new chain-complex. We use also two main constructive theorems: the *Short Exact Sequence (SES) theorem*, which provides the *constructive* version of the Mayer-Vietoris sequence, and the *Cone Reduction theorem*, which provides an efficient way to access the homology of the union of two simplicial complexes only from their *reduced* chain-complexes.

The reduction relates two chain-complexes with equivalent homologies in such a way that, if the homology of one of them is known, then the homology of the other can be found through reduction. Intuitively, it relates a *large* chain-complex $\widehat{C}_*$ to a *small* one C_*, which contains the same homological information in the most compact way.

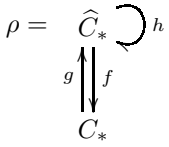


Figure 3: A reduction.

Definition 5.1 (Reduction). *A reduction $\rho : \widehat{C}_* \Rightarrow C_*$ is a diagram as shown in Figure 3, where $\widehat{C}_*$ and C_* are two chain-complexes; f and g are chain-complex morphisms satisfying $fg = id_{C_*}$; $h : \widehat{C}_{n-1} \to \widehat{C}_n$ is an homotopy operator satisfying the relations $gf + dh + hd = id_{\widehat{C}_*}$ and $fh = hg = hh = 0$.*

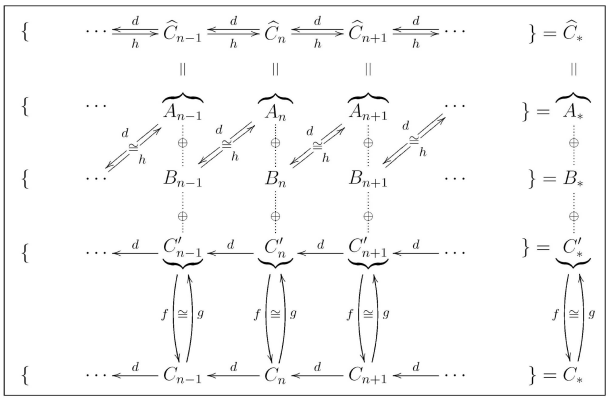


Figure 4: Reduction diagram (courtesy of [34]).

The reduction is a compact and convenient form for the diagram presented in Figure 4. It is equivalent to a decomposition, where every chain group $\widehat{C}_k$ is decomposed into three components: $\widehat{C}_k = A_k \oplus B_k \oplus C'_k$. Note that there exists a bijection between A_{k+1} and B_k through the boundary operator d and the homotopy operator h for every finite dimension k. Therefore, component A_{k+1} is a collection of $(k+1)$-cycles such that their boundaries are the elements in B_{k-1}. We call these cycles *pre-boundaries*. Component B_k is a collection of k-cycles known as k-boundaries. Component C'_k is a copy of C_k and, thus, $C'_* \cong \widehat{C}_*$. In summary, the large chain-complex $\widehat{C}_*$ is the direct sum of one small chain-complex C'_* and $A_* \oplus B_*$, where the last component does not play any role from a homological point of view. Given a chain-complex C_*, we call *trivial reduction*, the reduction where the small chain-complex is C_* itself, the morphism f and g are the identity morphisms, and the homotopy operators h are 0 morphisms.

Definition 5.2 (**Reduction equivalence**). *A reduction equivalence $C_* \Longleftrightarrow D_*$ between two chain-complexes C_*, D_*, is a pair of reductions connecting C_* and D_* through a third chain-complex $\widehat{C}_*$, as shown in Figure 5.*

The concept of *reduction equivalence* is used to relate three chain-complexes: the object of interest C_*, whose homology has to be computed, a very small homologically equivalent object D_* and a large object $\widehat{C}_*$, also equivalent. The homology information of object C_* is contained in the very small object D_*, while the big object $\widehat{C}_*$ is required to link C_* and D_*. Such an equivalence implies that the homology groups of the D_* and C_* are isomorphic.

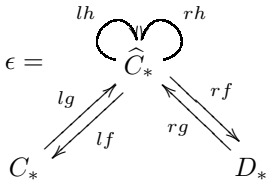


Figure 5: A reduction equivalence.

A cone of a morphism can simply be seen as a way to represent a morphism relating two chain-complexes as a new chain-complex. Informally, such a representation makes it possible to build an homologically equivalent object from the morphisms used to relate them.

Definition 5.3 (**Cone of a morphism**). *Let $f : X_* \to Y_*$ be a chain-complex morphism between two chain-complexes X_* and Y_*. The cone of the morphism f is a chain-complex, denoted $Cone(f)_*$. For each dimension $Cone(f)_k := Y_k \oplus X_{k-1}$ and the boundary operator is given by the matrix:*

$$D_{Cone(f)_k} := \begin{bmatrix} D_{Y_k} & f_{k-1} \\ 0 & -D_{X_{k-1}} \end{bmatrix}.$$

The matrices D_{Y_k} and $D_{X_{k-1}}$ are the incidence matrices of the chain-complexes Y_* and X_*. The groups Y_k and X_{k-1} are considered as disjoint and a basis of $Cone(f)_k$ is formed by a basis of Y_k and a basis of X_{k-1}.

Definition 5.4 (**Constructive exact short sequence of Mayer-Vietoris**). *Let A, B be two simplicial complexes with non-empty intersection $A \cap B$, then the following diagram defines a short exact sequence of their chain-complexes.*

$$0 \longleftarrow (A \cup B)_* \underset{j_A \ominus j_B}{\overset{\nu}{\rightleftarrows}} A_* \oplus B_* \underset{i_A \oplus i_B}{\overset{\rho}{\rightleftarrows}} (A \cap B)_* \longleftarrow 0$$

$$\begin{array}{rccc} i = i_A \oplus i_B : & (A \cap B)_* & \longrightarrow & A_* \oplus B_* \\ & \sigma & \longmapsto & (\sigma, \sigma) \\ j = j_A \ominus j_B : & A_* \oplus B_* & \longrightarrow & (A \cup B)_* \\ & (\sigma, \tilde{\sigma}) & \longmapsto & \sigma - \tilde{\sigma} \\ \nu : & (A \cup B)_* & \longrightarrow & A_* \oplus B_* \\ & \sigma & \longmapsto & (\sigma|_A, -\sigma|_B + \sigma|_{A \cap B}) \\ \rho : & A_* \oplus B_* & \longrightarrow & (A \cap B)_* \\ & (\sigma, \tilde{\sigma}) & \longmapsto & \tilde{\sigma}|_{A \cap B} \end{array}$$

The following theorem is the basic result on which our algorithm is based. It allows us to establish an homological equivalence between the cone of the morphism inclusion $i : (A \cap B)_* \to A_* \oplus B_*$ and the chain-complex of the union $(A \cup B)_*$.

Theorem 5.5 (**Short Exact Sequence (SES) theorem**). *The constructive short exact sequence of Mayer-Vietoris provides the reduction shown in Figure 6.*

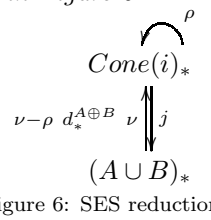


Figure 6: SES reduction.

Thus, if we know the homology of the cone of the morphism inclusion i, then we can retrieve the homology of $(A \cup B)_*$ by computation of the image of each element of the homology of $Cone(i)_*$ by morphism $f = j_A \ominus j_B$. However, chain-complex $Cone(i)_*$ is much larger than the chain-complex of the union $(A \cup B)_*$ and it would be extremely inefficient to compute the homology directly on this huge object.

The *Cone Reduction Theorem* gives us another reduction of chain-complex $Cone(i)_*$ and allows us to build a *reduction equivalence* between the chain-complex of the union $(A \cup B)_*$ and one very small chain-complex which is homologically equivalent to $Cone(i)_*$.

Theorem 5.6 (**Cone Reduction Theorem**). *Let $i : (A \cap B)_* \to (A \oplus B)_*$ be a chain-complex morphism and two reductions $(A \oplus B)_* \Rrightarrow EA_* \oplus EB_*$ and $(A \cap B)_* \Rrightarrow E(A \cap B)_*$. Then, we can define a reduction $\rho = (f_c, g_c, h_c) : Cone(i)_* \Rrightarrow Cone(Ei)_*$, as shown in Figure 7, where:*

$$f_c = \begin{bmatrix} f_{A \oplus B} & (-f_{A \oplus B})(i)(h_{A \cap B}) \\ 0 & f_{A \cap B} \end{bmatrix}$$

$$g_c = \begin{bmatrix} g_{A \oplus B} & (-h_{A \oplus B})(i)(g_{A \cap B}) \\ 0 & g_{A \cap B} \end{bmatrix}$$

$$h_c = \begin{bmatrix} h_{A \oplus B} & (h_{A \oplus B})(i)(h_{A \cap B}) \\ 0 & -h_{A \cap B} \end{bmatrix}.$$

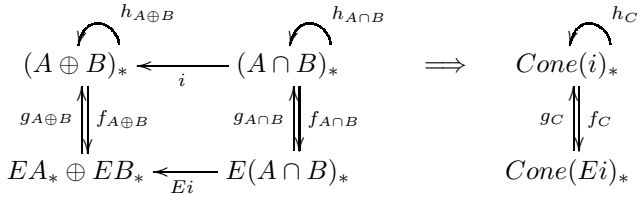


Figure 7: Cone Reduction theorem.

Note that the reduction $(A \oplus B)_* \Rrightarrow EA_* \oplus EB_*$ is simply defined as the formal sum of the reductions of the sub-complexes A and B.

By definition, chain-complex $Cone(Ei)_*$ is given by: $Cone((f_{A\oplus B})\circ(i)\circ(g_{A\cap B}))_* := EA_* \oplus EB_* \oplus E(A\cap B)_{*-1}$, where the chain-complexes EA_*, EB_* and $E(A \cap B)_*$ are the *reduced* chain-complexes of, respectively, A_*, B_* and $(A \cap B)_*$, and contain only their homological information. Therefore, we can efficiently compute the homology on this small chain-complex, by using the SNF algorithm.

As a consequence, we obtain the reduction equivalence shown in Figure 8, which demonstrates that the chain-complex $(A \cup B)_*$ has the same homology as the chain-complex $Cone(Ei)_*$. Therefore, the Betti numbers and the torsion coefficients of the union $(A \cup B)_*$ are provided directly from the homology of the chain-complex $Cone(Ei)_*$. The homological generators of $H_k((A \cup B)_*)$ can be obtained by computing the image of each cycle $c \in H_k(Cone(Ei)_*)$ by the morphisms $(j \circ g_c)(c)$.

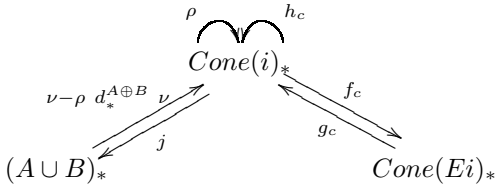


Figure 8: Cone reduction equivalence

Finally, we need to introduce the *Cone Equivalence theorem*, which will be useful for the MV algorithm, described in Section 7.

Theorem 5.7 (**Cone Equivalence Theorem**). *Let $i : (A \cap B)_* \to (A \oplus B)_*$ be a chain-complex morphism between two chain-complexes and two reduction equivalences, as shown in Figure 9. Thus, we can define the reduction equivalence:*

$$Cone(i) \overset{\rho_l}{\Leftarrow} Cone(\widehat{i}) \overset{\rho_r}{\Rrightarrow} Cone(Ei)$$

with $\widehat{i} = (lg')\ i\ (lf)$ *and* $Ei = (rf')\ (lg')\ i\ (lf)\ (rg)$

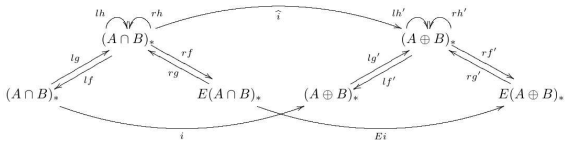


Figure 9: Cone equivalence theorem.

6. Homological Smith Reduction

In this section, we introduce a specific kind of reduction, which we call the *Homological Smith Reduction*. It will be used to encode the homology of each sub-complex of the input complex in our Mayer-Vietoris algorithm.

Given a simplicial complex X of finite dimension n, this reduction relates its chain-complex, X_*, with a very small chain-complex, EX_*, which contains only the homological information of X_*. This information is computed through the SNF algorithm, which transforms each incidence matrix D_k into its Smith Normal Form N_k. To describe chain-complex EX_*, we need a basis for each dimension and a boundary matrix. The basis is defined as a subset of Smith basis of X_*, while the boundary matrix is a sub-matrix of matrix N_k. The morphisms f, g and h relating the chain-complexes are defined from the matrices of the basis changes P_k, which is also restricted. Thus, we need first to classify the elements of the *Smith basis* provided by the SNF algorithm in order to find the basis of the small chain-complex EX_*.

Basis classification. Here, we illustrate the basis classification algorithm through the example shown in Figure 10. Let N_k and N_{k+1} be two consecutive incidence matrices in Smith Normal Form. We assume that $\beta_s^k = \{\gamma_1, \ldots, \gamma_l\}$ is the Smith basis, in which the columns of N_k and the rows of N_{k+1} are expressed.

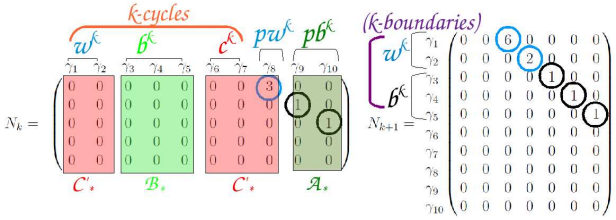


Figure 10: Smith basis classification $\beta_s^k = \{w^k, b^k, c^k, pw^k, pb^k\}$.

Consider now the sub-basis of the k-cycles $\ker d_k = [\gamma_1, \ldots, \gamma_7]$, which corresponds to the zero columns of N_k. This basis is formed by the union of three sub-basis, defined as follows. First, the sub-basis $\boldsymbol{w^k} = \{\gamma_1, \gamma_2\}$ is composed of the elements corresponding to the *weak-boundaries* which are associated with the torsion coefficients. They correspond to the rows of N_{k+1} with coefficient $\lambda_i > 1$. Second, the sub-basis $\boldsymbol{b^k} = \{\gamma_3, \gamma_4, \gamma_5\}$ is composed of the elements corresponding to the *boundaries* and can be retrieved through the rows of N_{k+1} with coefficient equal to 1. Finally, the remaining kernel basis corresponds to the non-trivial k-cycles $\boldsymbol{c^k} = \{\gamma_6, \gamma_7\}$.

We complete the basis classification with the k-chains which are not k-cycles. The elements corresponding to the columns of N_k with coefficients equal to 1 are called *pre-boundaries*, $\boldsymbol{pb^k} = \{\gamma_9, \gamma_{10}\}$. These chains do not carry any homological information. The elements corresponding to the columns of N_k with coefficients $\lambda_i > 1$ are called *pre-weak boundaries*, $\boldsymbol{pw^k} = \{\gamma_8\}$ and they are related to the torsion coefficients.

The basis classification for vertices and for simplices of dimension n must be treated as special cases, since the boundary morphisms at dimension 0 and $n+1$ are zero morphisms. Therefore, in the basis of dimension n there

are only cycles, pre-boundaries, and possibly pre-weak-boundaries but not weak-boundaries nor boundaries. In the vertex basis, there are only cycles and boundaries.

Reduced chain-complexes. The basis classification allows us to construct the *reduced* chain-complex EX_* from the original one X_*. Notice that the basis classification is equivalent to the decomposition of each original chain-group X_k into three sub-groups: $X_k = A_k \oplus B_k \oplus C'_k$, as shown in Figure 10.

Component $A_k = [\boldsymbol{pb}^k]$ is generated by the k-chains which do not play any role in homology computation. The chain-group $B_k = [\boldsymbol{b}^k]$ is generated by the k-cycles which are known to be boundaries. Note that, the subgroup generated by the pre-boundaries $[\boldsymbol{pb}^k]$ is isomorphic to the subgroup generated by the boundaries $[\boldsymbol{b}^{k-1}]$, since the identity sub-matrix relates them. In summary, the homology of X_k is given by the *reduced* chain-complex $EX_k = C'_k = [\boldsymbol{w}^k, \boldsymbol{c}^k, \boldsymbol{pw}^k]$. For each dimension $1 \leq k \leq n$, the boundary matrix EN_k of EX_k is:

$$EN_k := \begin{array}{c} \\ \boldsymbol{w}^{k-1} \\ \boldsymbol{c}^{k-1} \\ \boldsymbol{pw}^{k-1} \end{array} \begin{array}{c} \begin{array}{ccc} \boldsymbol{w}^k & \boldsymbol{c}^k & \boldsymbol{pw}^k \end{array} \\ \begin{pmatrix} 0 & 0 & \boldsymbol{\lambda} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}.$$

It is immediate to prove that $EN_{k-1}EN_k = 0 \quad \forall k$, so EC_* is effectively a chain-complex.

Definition 6.1 (Homological Smith Reduction)**.** *Let X_* be a chain-complex X_*, then its Homological Smith Reduction is*

$$\rho = \quad X_* \circlearrowright h \quad g \uparrow \downarrow f \quad EX_* = [\boldsymbol{w}^*, \boldsymbol{c}^*, \boldsymbol{pw}^*]$$

with:
$f : X_* \to EX_*, f_k = (P_k)^{-1}|_{\boldsymbol{w}^k, \boldsymbol{c}^k, \boldsymbol{pw}^k}\}$
$g : EC_* \to X_*, g_k = P_k|_{\boldsymbol{w}^k, \boldsymbol{c}^k, \boldsymbol{pw}^k}\}$
$h : X_* \to X_{*+1}, h_k = (P_k)|_{pb^k} * (P_{k-1})^{-1}|_{b^{k-1}}$

The chain-complex morphisms f and g are inverse isomorphisms between EX_* and a subchain of X_* that contains all the homological information of X_*.

The restriction of the homotopy operator $h_k : B_k \to A_{k+1}$ and the restriction of the boundary operator $d_k : A_{k+1} \to B_k$ are isomorphisms between boundaries and pre-boundaries. This means that, given a boundary $\sigma^k \in B_k$, h_k gives us the $(k+1)$-chain of A_{k+1} for which σ is the boundary. Intuitively, the homotopy operator h captures only the information about the boundaries and their pre-boundaries. It can be seen as the *constructive* version of the definition of boundary. The algorithm for computing the *Homological Smith Reduction* of a chain-complex X_* is summarized in Algorithm 1.

Algorithm 1 Building the Homological Smith Reduction

Input: A chain-complex X_*.
Output: The reduction $X_* \Rrightarrow EX_*$.
1: **For all** $1 \leq k \leq dim(X)$ **do**:
2: Compute the SNF of the incidence matrix $D_k = P_{k-1}N_kP_k^{-1}$.
3: Classify the Smith basis : $[w^k, b^k, c^k, pw^k, pb^k]$
4: Build the reduction by cutting the matrices:
$EX_k := N_k|_{w^k, c^k, pw^k}$
$f_k := (P_k)^{-1}|_{w^k, c^k, pw^k}$
$g_k := (P_k)|_{w^k, c^k, pw^k}$
$h_k := (P_k)|_{pb^k} * (P_{k-1})^{-1}|_{b^{k-1}}$
5: **End for**

7. The Mayer-Vietoris Algorithm

In this section, we first introduce the algorithm based on the constructive Mayer-Vietoris sequence, which computes the homology of the union of two simplicial complexes. Then, we explain how this algorithm can be applied on a Homo-MC graph thus resulting in the iterative Mayer-Vietoris homology computation algorithm.

7.1. Homology computation of the union of two complexes

Here, we explain how the constructive version of the Mayer-Vietoris sequence, introduced in Section 5, yields to an effective algorithm for homology computation. We illustrate this algorithm through the example in Figure 11, where a simplicial complex X is decomposed onto two MC-components A and B with non-empty intersection $A \cap B$, formed by two isolated vertices.

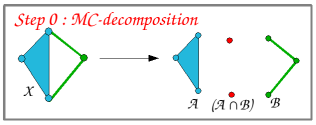


Figure 11: The MC-decomposition of a simplicial complex X into two MC-components A and B.

The first step of the algorithm consists in computing the Homological Smith Reductions of the three input complexes A, B and $A \cap B$, as explained in Section 6. We obtain reductions $A_* \Rrightarrow EA_*$, $B_* \Rrightarrow EB_*$ and $(A \cap B)_* \Rrightarrow E(A \cap B)_*$, as shown in Figure 12. Recall that the *reduced* chain-complexes EA_*, EB_* and $E(A \cap B)_*$ are homology equivalent to the large chain-complexes A_*, B_* and $(A \cap B)_*$ but contain *only* the homological information.

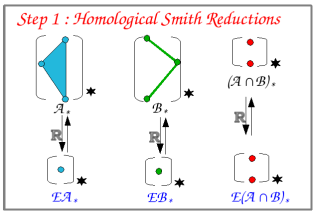


Figure 12: Homological Smith Reductions of A_*, B_* and $(A \cap B)_*$, associated to the example in Figure 11.

The second step of the algorithm builds the *constructive* Mayer-Vietoris sequence, as shown in Figure 13(a), and computes morphisms i, j, ρ and ν from the input simplicial complexes, following definition 5.4.

At this point, we can apply the SES theorem (theorem 5.5) which builds a reduction between the cone of

the inclusion morphism i and the chain-complex associated with the union of the sub-complexes A and B: $Cone(i)_* \Rrightarrow (A \cup B)_*$, as illustrated in Figure 13(b). This means that the chain-complex of the cone has the same homology as the chain-complex of the union. However, as indicated by the direction of the reduction, the size of the former is larger (in terms of number of simplices) than that of the latter, and computing the homology of the union through this large chain-complex would be extremely inefficient.

The fourth step of the algorithm applies the cone reduction theorem (theorem 5.6) in order to build a new reduction of the chain-complex of cone $Cone(i)_* \Rrightarrow Cone(Ei)_*$. This reduction establishes a homological equivalence between the large chain-complex $Cone(i)_*$ and the chain-complex associated with the cone of the inclusion morphism Ei, which relates the *reduced* chain-complexes $EA_* \oplus EB_*$ and $E(A \cap B)_*$, as shown in Figure 13(c). Note that chain-complex $Cone(Ei)_*$ can be efficiently computed from the reduced chains $EA_* \oplus EB_*$ and $E(A \cap B)_*$, following definition 5.3.

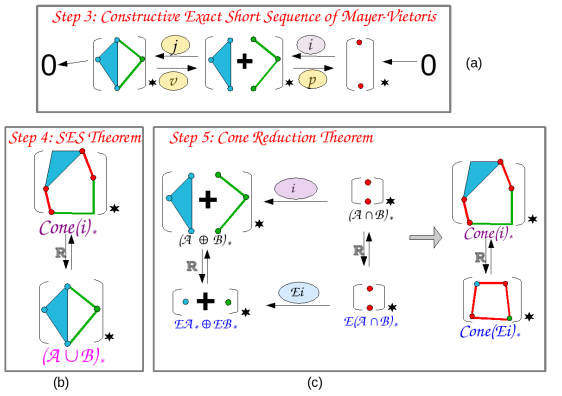


Figure 13: Main steps of our algorithm, applied to the example in Figure 11. We use (a) the Constructive Mayer-Vietoris sequence, (b) the SES theorem, and (c) the Cone Reduction theorem.

At this point, we establish the *reduction equivalence* $(A \cup B)_* \Longleftrightarrow Cone(Ei)_*$ from the two last reductions. This means that chain-complex $Cone(Ei)_*$ has the same homology as the chain-complex of the union $(A \cup B)_*$, since they are related through the large chain-complex $Cone(i)_*$, as shown in Figure 14(a). However, chain-complex $Cone(Ei)_*$ is much smaller (in terms of number of simplices) than the chain-complex of the union, since it contains only the homological information of sub-complexes A, B and $A \cap B$.

The next step of the algorithm computes the homology of the small chain-complex $Cone(Ei)_*$ through the SNF algorithm, obtaining the Homological Smith Reduction $Cone(Ei)_* \Rrightarrow ECone(Ei)_*$, as shown in Figure 14(b).

Finally, the algorithm composes the last two reductions. It outputs a *reduction equivalence* between the chain-complex of the union $(A \cup B)_*$ and chain-complex $ECone(Ei)_*$, as shown in Figure 14(c). From the reduction equivalence we can extract the required homological information. The Betti numbers and the torsion coefficients of the union can be directly accessed in $ECone(Ei)_*$, while the generators of the union are obtained by computing the image of the generators of $ECone(Ei)_*$ through the morphisms of the reduction equivalence.

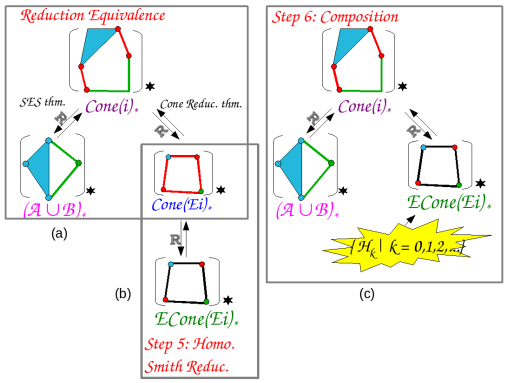


Figure 14: Last steps of our algorithm for computing homologies of the union of two complexes A and B.

The main steps of our algorithm for computing the homology of the union of two complexes are summarized in Algorithm 2.

Algorithm 2 Homology of the union of two complexes

Input: three simplicial complexes A, B and $A \cap B \neq \emptyset$.
Output: the reduction equivalence:
$(A \cup B)_* \Lleftarrow Cone(i) \Rrightarrow ECone(Ei)_*$

1: Compute the morphisms i, j, ρ and ν of the Constructive Mayer-Vietoris sequence for the chain-complexes $(A \oplus B)_*$, $(A \cap B)_*$ and $(A \cup B)_*$.
2: Build the reduction of the morphism inclusion i, provided by the SES Theorem: $Cone(i)_* \Rrightarrow (A \cup B)_*$.
3: Build the reduction of the morphism inclusion i, provided by the Cone Reduction Theorem: $Cone(i)_* \Rrightarrow Cone(Ei)_*$.
4: Compute the Homological Smith Reduction of the reduced chain-complex: $Cone(Ei)_* \Rrightarrow E(Cone(Ei))_*$.
5: Compose the last two reductions: $Cone(i)_* \Rrightarrow E(Cone(Ei))_*$

7.2. Mayer-Vietoris Iterative Algorithm

In this section, we introduce our Mayer-Vietoris iterative algorithm for computing the homology of a simplicial complex X, starting from its Homo-MC graph G_X. This algorithm iteratively computes the homology of the union of two MC-components A and B connected through an arc in the Homo-MC graph and merges the two components. It terminates when the graph consists of a single node. We will use Algorithm 2, introduced in subsection 7.1, to compute the homology of the sub-complex $A \cup B$.

However, before proceeding we need to find a way to reuse the *reduction equivalence* provided as output by Algorithm 2. Thus, we slightly modify the second step of this algorithm and we associate a *reduction equivalence* with each sub-complex. For each MC-component N, the algorithm computes a reduction equivalence $N_* \Lleftarrow N_* \Rrightarrow E(N)_*$, where the right reduction is the Homological Smith Reduction, which is computed as previously through Algorithm 1. The left reduction is simply the *trivial* reduction of N, at the beginning. Note that it is pos-

sible to compute the reduction equivalence for each node in G_X as a pre-processing step and in parallel. As a consequence, the modified Algorithm 2 should use, at step 4, the cone equivalence theorem (theorem 5.7), instead of the cone reduction one.

At each step, the algorithm collapses the arc between two components A and B in the Homo-MC graph and generates a new component AB (node in the graph) by merging the lists of the top simplices. It also updates the arcs incident in A and B, which become incident in AB. Then, it associates the reduction equivalence, computed by Algorithm 2, with the new component AB in order to make it reused in the subsequent steps. The algorithm repeats this operation until there is only one node in the graph. When the algorithm stops, the last node corresponds to the input simplicial complex X and its $\mathbb{Z}$-homology is retrieved from the reduction equivalence associated with this node, as performed in the final step of Algorithm 2.

Algorithm 3 summarizes the main steps of our Mayer-Vietoris iterative algorithm for homology computation.

Algorithm 3 Mayer-Vietoris iterative computation

Input: the Homo-MC graph G of a simplicial complex X.
Output: homology information for X

1: Initialize the reduction equivalence for all nodes in G_X
2: **while** there is more than one arc in G_X **do:**
3: Select a random arc $a = (n_A, n_B)$ in G_X.
4: Apply Algorithm 2 to n_A, n_B, and $n_A \cap n_B$
5: Create a new node $n_{AB} = n_A \cup n_B$
6: Associate the new reduction equivalence with n_{AB}
7: **End while**
8: Extract the $\mathbb{Z}$-homology of X from the unique node in G_X

Note that any sub-complex AB resulting the union of two MC-components A and B is not manifold-connected. Thus, the decomposition we obtain at any intermediate step is not an MC-decomposition. However, the intersections of the components is still composed by a limited number of non-manifold simplices.

8. Experimental Results

In this section, we present qualitative and quantitative results about the MC-decomposition and our Mayer-Vietoris (MV) algorithm. We have tested our algorithms on some datasets freely available from [19], on a computer with a $3,2$ Ghz Intel i7 processor and 16 Gb of RAM.

We first demonstrate one of the most important properties of the MC-decomposition, which is critical for the efficiency of the homology computation. Recall that our MV algorithm operates on decomposed shapes and computes the homology of the input model from the homology of the components and the homology of their intersection complexes through Mayer-Vietoris sequences. Thus, in order to reduce the redundancies during the homological computation, it is mandatory to use a decomposition which minimizes the size of the intersections (in terms of number of simplices). As shown in Table 1, the size of the intersection complexes (which correspond to the singularities shared by two components) is very small in comparison with the size of the input complex, and it never exceeds 5% of this size. This fact makes the MC-decomposition perfectly suitable as a basis for the MV algorithm.

Shape	s_0	s_1	s_2	S	N	A	MS	MA
armchair	43	125	88	256	6	8	32%	3%
twist	$1K$	$4K$	$2K$	$7K$	4	5	65%	0.8%
two-twist	$1K$	$5K$	$3K$	$9K$	8	13	45%	0.9%
carter	$4K$	$12K$	$8K$	$24K$	28	40	45%	0.6%

Table 1: Statistics about the Homo MC-Graph. Here, we analyze non-manifold shapes formed by s_0 vertices, s_1 edges and s_2 triangles: their corresponding Homo MC Graphs has N nodes and A arcs. It is interesting to compare the size MS of the largest MC-component and the size MA of the largest intersection between two MC-components, both expressed as a percentage of the total number of simplices $S = s_0 + s_1 + s_2$ in the input complex.

Our MV algorithm has been designed for computing the *complete* homological information for an arbitrary shape, including not only the Betti numbers, but also the generators, and the torsion coefficients, if there are any.

Figure 15 shows the MC decomposition of three non-manifold 2-simplicial complexes and some of the generators for the homology groups H_1 and H_2, computed by the MV algorithm. Note that the *twist* model, Figure 15(a), is isomorphic to a torus (in wired grey) in which there is an other embedded 2-cycle (in blue). The *two-twist* model, Figure 15(d), is equivalent to two intersecting tori, corresponding to the yellow and the wired grey 2-cycles, with one embedded shell, corresponding to the blue 2-cycle. The *carter* model has a very complicated topology. Some of its 1-cycles and 2-cycles are shown in Figure 15(f).

We have decided to compare our MV algorithm to the classical SNF algorithm, which is the most general method for computing the $\mathbb{Z}$-homology. Recall that the SNF algorithm operates on the entire model and computes the incidence matrices from the entire shape, while our MV algorithm uses the SNF algorithm to compute the homology on the MC-components. Our current implementation encodes the classical SNF algorithm, without any optimization: in any case, it is possible to use any other version of this algorithm, provided it keeps track of the basis changes during the reduction of the matrices.

Our experimental results, summarized in Table 2, tend to prove that the MV algorithm is a reasonable tool for computing the $\mathbb{Z}$-homology on simplicial shapes, requiring less space than the SNF algorithm, and providing a relevant speed-up to the computation.

The key point in our storage analysis is the size of the incidence matrices, which we have to be reduced. Recall that an incidence matrix D_k of dimension k relates the chain-groups C_{k-1} and C_k and it requires $I_k = s_k \times s_{k-1}$ integer values (encoded on 4 bytes), where s_j denotes the number of j-simplices in the input simplicial complex. The SNF algorithm needs D_1 and D_2, and thus its storage cost

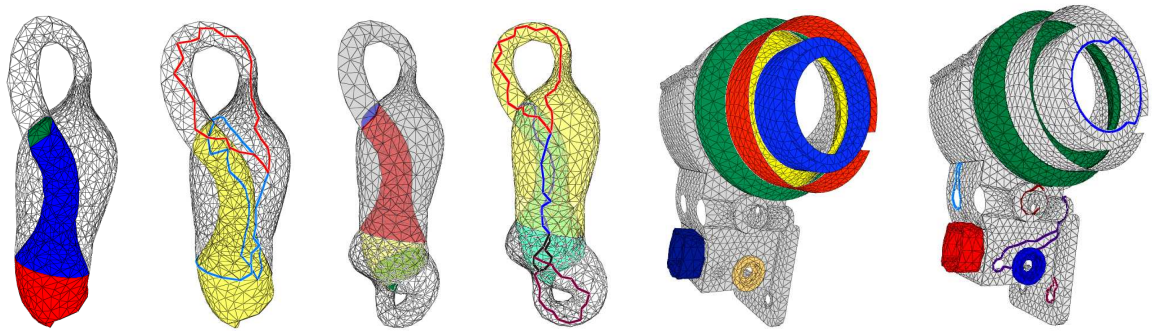


Figure 15: Examples of MC-decomposition and homology generators computed with our MV algorithm on some non-manifold 2-simplicial complexes. (a) The MC decomposition, (b) two 1-cycles (in red and blue), and two 2-cycles (in grey and yellow) for the *twist* model. (c) The MC decomposition, (d) four 1-cycles (in red, blue, black and purple), and three 2-cycles (in yellow, blue and grey) for the *two-twist* model. (e) The MC decomposition and (f) some free generators of H_1 and H_2 for the *carter* model.

Shape	$(\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2)$	$\mathcal{I}_1$	$\mathcal{I}_2$	$\mathcal{SNF}_s$	$\mathcal{SNF}_t$	$\mathcal{MI}_1$	$\mathcal{MI}_2$	$\mathcal{MV}_s$	$\mathcal{MV}_t$	$\mathcal{R}_s$	$\mathcal{R}_t$
armchair	$(\mathbb{Z}, 0, \mathbb{Z}^5)$	0.2 Mb	0.4 Mb	0.6 Mb	60 ms	0.03 Mb	0.04 Mb	0.07 Mb	19 ms	88%	3.2
twist	$(\mathbb{Z}, \mathbb{Z}^2, \mathbb{Z}^2)$	16 Mb	34 Mb	50 Mb	2.2×10^6 ms	7 Mb	14 Mb	21 Mb	1.4×10^6 ms	55%	1.6
two-twist	$(\mathbb{Z}, \mathbb{Z}^4, \mathbb{Z}^3)$	26 Mb	54 Mb	80 Mb	1.2×10^7 ms	7 Mb	14 Mb	21 Mb	3×10^6 ms	73%	3.8
carter	$(\mathbb{Z}, \mathbb{Z}^{27}, \mathbb{Z}^5)$	190 Mb	377 Mb	567 Mb	7.7×10^7 ms	41 Mb	75 Mb	116 Mb	1.7×10^7 ms	79%	4.5

Table 2: Comparisons in terms of timings and storage cost between the SNF and the MV algorithms, which compute the $\mathbb{Z}$-homology $(\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2)$ for some non-manifold simplicial complexes. Columns $\mathcal{I}_1$ and $\mathcal{I}_2$ indicate the size (in Mb) of the incidence matrices for the entire shape. Columns $\mathcal{SNF}_s$ and $\mathcal{SNF}_t$ show respectively the storage cost (in Mb) and the timing (in ms) required by the SNF algorithm. Columns $\mathcal{MI}_1$ and $\mathcal{MI}_2$ show the size (in Mb) of the incidence matrices for the largest MC component. Columns $\mathcal{MV}_s$ and $\mathcal{MV}_t$ show respectively the storage cost (in Mb) and the timing (in ms) required by the MV algorithm. We also provide the reduction of storage cost $\mathcal{R}_s$ (expressed as a percentage of $\mathcal{SNF}_s$), and the ratio $\mathcal{R}_t$ between $\mathcal{SNF}_t$ and $\mathcal{MV}_t$.

$\mathcal{SNF}_s$ is $\mathcal{O}(\mathcal{I}_1 + \mathcal{I}_2)$. Conversely, our MV algorithm computes, at each step, the homology of the union of only two sub-complexes $\mathcal{A}$, $\mathcal{B}$ and their intersection $\mathcal{A} \cap \mathcal{B}$. Since the size of the intersection complex is usually very small, we can ignore it. Thus, we operate only on the incidence matrices for the components $\mathcal{A}$ and $\mathcal{B}$, namely $\mathcal{D}_k^{\mathcal{A}}$ and $\mathcal{D}_k^{\mathcal{B}}$, with $k = 1, 2$. In the following, we respectively indicate their size as $\mathcal{I}_k^{\mathcal{A}}$ and $\mathcal{I}_k^{\mathcal{B}}$. Thus, the storage cost of each step is $\mathcal{O}(\mathcal{I}_1^{\mathcal{A}} + \mathcal{I}_2^{\mathcal{A}} + \mathcal{I}_1^{\mathcal{B}} + \mathcal{I}_2^{\mathcal{B}})$. If we consider the size of incidence matrices for the largest MC-component, respectively indicated as $\mathcal{MI}_1$ and $\mathcal{MI}_2$, then the storage cost of the MV algorithm $\mathcal{MV}_s$ becomes $\mathcal{O}(\mathcal{MI}_1 + \mathcal{MI}_2)$. This fact demonstrates that the MV algorithm requires much less space than the SNF algorithm: in our tests, we obtained a reduction of at least 55% of $\mathcal{SNF}_s$ (see column $\mathcal{R}_s$ in Table 2). We also provide timing comparisons between our MV algorithm and the SNF one, demonstrating that we obtain a relevant speed-up with our approach. In our tests, the MV algorithm is at least 1.6 times faster than the SNF algorithm (see column $\mathcal{R}_t$ in Table 2).

However, advantages introduced by the MV algorithm can be slightly reduced in some cases, as shown in Table 2 for the *twist* model. Since the Homo-MC Decomposition does not impose any limitation on the components size, it is possible to obtain a large MC-component. Thus, computing the homology of this MC-component through the SNF algorithm is time consuming.

This issue could be overcome in different ways. For instance, we can reduce the size of the MC-components through a special simplification algorithm, which preserves its topology and handles its singularities (i.e., non-manifold simplices). In this way, the homology generators of the component can be computed only on the simplified version. However, if this component has to be merged with another MC-component by the MV algorithm in a later iteration, the generators have to be expressed in the original (non-simplified) MC-component, in order to ensure the consistency of the computation duharing the union. It is also possible to gain in efficiency by using one of the existing optimizations of the SNF algorithm for sparse integer matrices. As noted before, our framework does not depend on the SNF algorithm selected and it can work with any optimized version of this latter.

9. Concluding Remarks

We have introduced a new algorithmic approach for homology computation on arbitrary non-manifold shapes discretized through simplicial complexes. Our algorithm is based on the constructive version of the Mayer-Vietoris sequence, which allows us to compute the $\mathbb{Z}$-homology of a simplicial complex from the homologies of its subcomplexes. We call this algorithm as *Mayer-Vietoris (MV)*

algorithm. The starting point of the MV algorithm is the MC-decomposition of the shape, which minimizes the intersections between the manifold-connected components. Combined with this decomposition, the MV algorithm has been proven to be more efficient (in terms of storage and timings) than the classical SNF algorithm, which operates on the entire input model.

In the future, we are planning to improve our current implementation of the SNF algorithm, which will allow us to increase the efficiency of the homology computation on each MC-component and, thus, process very large models. We are also planning to investigate how to combine our algorithm with a different approach for computing the homology of the MC-components, since these latter can be viewed as almost manifold complexes, and thus efficient geometric algorithms for homology computation on manifold shapes could be applied.

We are also planning to investigate different strategies for improving the geometric properties of the generators in order to provide the shortest set of loops that generate the homology groups. One possible solution is to minimize their length, by associating a metric to the homology basis, as recently introduced in [12].

Acknowledgements. We thank Professor Francis Sergeraert for the many helpful discussions on Constructive Homology. We thank the anonymous reviewers for their helpful suggestions.

References

[1] Agoston, M.K., 1976. Algebraic topology: a first course. M. Dekker Publisher, New York, USA.

[2] Alayrangues, S., Damiand, G., Fuchs, L., Lienhardt, P., Peltier, S., 2009. Homology computation on cellular structures in image context, in: CTIC, Austria. pp. 19–28.

[3] Boltcheva, D., Merino, S., Léon, J.C., Hétroy, F., 2010. Constructive Mayer-Vietoris algorithm: computing the homology of unions of simplicial complexes. Technical Report. INRIA-7471.

[4] Chen, C., Freedman, D., 2010. Measuring and computing natural generators for homology groups. Computational Geometry: Theory & Applications 43, 169–181.

[5] Cohen-Steiner, D., Edelsbrunner, H., Morozov, D., 2006. Vines and vineyards by updating persistence in linear time, in: Symposium on Computational geometry, SCG '06, pp. 119–126.

[6] Damiand, G., Peltier, S., Fuchs, L., 2006. Computing homology for surfaces with generalized maps: Application to 3d images., in: ISVC'06, Nevada, USA. pp. 235–244.

[7] De Floriani, L., Hui, A., Panozzo, D., Canino, D., 2010. A dimension-independent data structure for simplicial complexes, in: IMR, Chattanoga, TN, USA. pp. 403–420.

[8] De Floriani, L., Mesmoudi, M.M., Morando, F., Puppo, E., 2003. Decomposing non-manifold objects in arbitrary dimensions. Graph. Models 65, 2–22.

[9] Delfinado, C.J.A., Edelsbrunner, H., 1993. An incremental algorithm for betti numbers of simplicial complexes, in: SoCG'93, ACM, New York, NY, USA. pp. 232–239.

[10] Dey, T.K., Guha, S., 1996. Computing homology groups of simplicial complexes in $\mathbb{R}^3$, in: Proc. 28th ACM Symp. Comp. Theory (STOC 1996).

[11] Dey, T.K., Li, K., Sun, J., Cohen-Steiner, D., 2008. Computing geometry-aware handle and tunnel loops in 3d models. ACM Transactions on Graphics 27, No. 45.

[12] Dey, T.K., Sun, J., Wang, Y., 2010. Approximating loops in a shortest homology basis from point data, in: Proc. 26th Annu. Sympos. Comput. Geom. (SOCG 2010), pp. 166–175.

[13] Dousson, X., Rubio, J., Sergeraert, F., Siret, Y., 2008. The kenzo program. `http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/`.

[14] Dumas, J.G., Heckenbach, F., Saunders, B.D., Welker, V., 2003. Computing simplicial homology based on efficient smith normal form algorithms. Alg., Geom., and Soft. Syst. , 177–207.

[15] Edelsbrunner, H., Harer, J., 2010. Computational topology, an introduction. AMS, Providence, Rhode Island.

[16] Edelsbrunner, H., Letscher, D., Zomorodian, A., 2002. Topological persistence and simplification. Discrete & Computational Geometry 28, 511–533.

[17] Fabio, B., Landi, C., Medri, F., 2009. Recognition of occluded shapes using size functions, in: ICIAP'09, Springer-Verlag, Berlin, Heidelberg. pp. 642–651.

[18] Forman, R., 1998. Morse theory for cell complexes. Advances in Mathematics 134, 90–145.

[19] Geometry and Graphics Group, 2009. Non-manifold Meshes Repository. `http://indy.disi.unige.it/nmcollection/`.

[20] Ghrist, R., Muhammad, A., 2005. Coverage and hole-detection in sensor networks via homology, in: IPSN'05, IEEE Press, Piscataway, NJ, USA. p. No. 34.

[21] Giesbrecht, M., 1996. Probabilistic computation of the smith normal form of a sparse integer matrix. Algorithmic Number Theory. Lecture Notes in Computer Science 1122, 173–186.

[22] González-Díaz, R., Jiménez, M.J., Medrano, B., Real, P., 2009. Chain homotopies for object topological representations. Discrete Applied Mathematics 157, 490–499.

[23] Gotsman, C., Kaligosi, K., Mehlhorn, K., Michail, D., Pyrga, E., 2007. Cycle bases of graphs and sampled manifolds. Computer Aided Geometric Design 24, 464–480.

[24] Hafner, J.L., McCurley, K.S., 1991. Asymptotically fast triangularization of matrices over rings. SIAM Journal on Computing 20, 1068–1083.

[25] Hatcher, A., 2002. Algebraic Topology. Cambr. Univ. Press.

[26] Hui, A., De Floriani, L., 2007. A two-level topological decomposition for non-manifold simplicial shapes, in: SPM, Beijing, China. pp. 355–260.

[27] Kaczynski, T., Mrozek, M., Slusarek, M., 1998. Homology computation by reduction of chain complexes. Computers & Mathematics with Applications 35-4, 59–70.

[28] Kannan, R., A., B., 1979. Polinomial algorithms for computing the smith and hermite normal forms of an integer matrix. SIAM Journal of Computing 8, 499–507.

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

[30] Munkres, J., 1999. Algebraic Topology. Prentice Hall.

[31] Peltier, S., Ion, A., Kropatsch, W.G., Damiand, G., Haxhimusa, Y., 2009. Directly computing the generators of image homology using graph pyramids. Image & Vision Computing 27, 846–853.

[32] Sergeraert, F., 1994. The computability problem in algebraic topology. Advances in Mathematics 104, 139–155.

[33] Sergeraert, F., 1999. Constructive algebraic topology. SIGSAM Bull. 33, 13–25.

[34] Sergeraert, F., Rubio, J., 2006. Constructive homological algebra and applications. `http://www-fourier.ujf-grenoble.fr/~sergerar/Papers/`.

[35] Storjohann, A., 1996. Near optimal algorithms for computing smith normal forms of integer matrices, in: ISSAC'96, ACM, New York, NY, USA. pp. 267–274.

[36] Thakur, A., Banerjee, A.G., Gupta, S.K., 2009. A survey of cad model simplification techniques for physics-based simulation applications. Comput. Aided Des. 41, 65–80.

[37] Troelstra, A.S., van Dalen, D., 1988. Constructivism in Mathematics, an Introduction. Studies in Logic and the Foundations of Mathematics, North-Holland.

[38] Zhu, X., Sarkar, R., Gao, J., 2009. Topological data processing for distributed sensor networks with morse-smale decomposition, in: IEEE Infocom, pp. 2911–2915.

[39] Zomorodian, A., Carlsson, G., 2008. Localized homology. Computational Geometry: Theory & Applications 41, 126–148.