Removal Operations in nD Generalized Maps for Efficient Homology Computation
Guillaume Damiand, Rocio Gonzalez-Diaz, Samuel Peltier

To cite this version:
Removal Operations in $n$D Generalized Maps for Efficient Homology Computation*

Guillaume Damiand$^1$, Rocio Gonzalez-Diaz$^2$, and Samuel Peltier$^3$

$^1$ Université de Lyon, CNRS, LIRIS, UMR5205, F-69622 France
$^2$ Universidad de Sevilla, Dpto. de Matemática Aplicada I, S-41012, Spain
$^3$ Université de Poitiers, CNRS, XLIM-SIC, UMR6172, F-86962 France

Abstract. In this paper, we present an efficient way for computing homology generators of $n$D generalized maps. The algorithm proceeds in two steps: (1) cell removals reduces the number of cells while preserving homology; (2) homology generator computation is performed on the reduced object by reducing incidence matrices into their Smith-Agoston normal form. In this paper, we provide a definition of cells that can be removed while preserving homology. Some results on 2D and 3D homology generators computation are presented.

Keywords: $n$D Generalized Maps; Cellular Homology; Homology Generators; Removal Operations.

1 Introduction

In this paper, we propose a method for efficiently computing homology generators of subdivided cellular objects. The main idea is to simplify a subdivided object into a smaller one while preserving its homology. This principle is similar to the one used in [10] which is mainly algebraic (i.e. based on reduction of chain complexes), while our approach is mainly combinatorial.

In this work, we define a simplification algorithm based on the cell removal operations defined on generalized maps. Its principle is to simplify as much as possible the number of cells while preserving homology. Then we reduce incidence matrices (used for describing boundary operators) into their Smith-Agoston normal form for computing homology generators [3]. Moreover, generators computed in the reduced object can easily be projected into the original one.

The paper is structured as follows: in Sect. 2 all the necessary background regarding $n$-Gmaps is recalled. Section 3 presents the main result of the paper: the definition of the simplification algorithm based on the removal of two types of cells, and the proof of the homology preservation. Finally, some experiments are presented in Sect. 4 in order to illustrate that the simplification step widely reduces the number of cells, and also the homology generator computation.

* Paper published in Proceedings of 4th International Workshop on Computational Topology in Image Context, LNCS 7309, pp. 20-29, June 2012. Thanks to Springer Berlin Heidelberg. The original publication is available at http://dx.doi.org/10.1007/978-3-642-30238-1_3
2 Preliminary Works

An $n$-Gmap is a combinatorial structure devoted to the representation of cellular subdivision of orientable or not orientable $n$D quasi-manifolds, with or without boundaries (see [11, 12] for more details). Any polytopal complex can be described by an $n$-Gmap, while the converse is not true (an $i$-cell can be non homeomorphic to an $i$-disk). It is possible to associate a semi-simplicial set with any $n$-Gmap. An $n$-Gmap is not constructed directly from the cells of the subdivision but from more elementary objects: darts. The set of darts is structured through involutions that describe how they are linked to each other.

Definition 1 ($n$-Gmap). An $n$-dimensional generalized map, called $n$-Gmap, with $0 \leq n$, is a $(n+2)$-tuple $G = (D, \alpha_0, \ldots, \alpha_n)$ where:

1. $D$ is a finite set of darts;
2. $\forall i$, $0 \leq i \leq n$, $\alpha_i$ is an involution on $D$;
3. $\forall i$, $0 \leq i \leq n-2$, $\forall j$, $i + 2 \leq j \leq n$, $\alpha_i \circ \alpha_j$ is an involution.

The cells of the subdivision are defined implicitly as set of darts thanks to the orbit notion (see Def. 2). An orbit in an $n$-Gmap can be seen as the set of darts that we can reach from a given dart and using as many times as possible the given involutions.

Definition 2 (Orbit). Let $\Phi = \{\pi_0, \cdots, \pi_n\}$ be a set of permutations defined on a set $D$. $\langle \Phi \rangle$ is the permutation group of $D$ generated by $\Phi$. The orbit of an element $d \in D$ relatively to $\langle \Phi \rangle$, denoted $\langle \Phi \rangle (d)$ is the set $\{ \phi(d) \mid \phi \in \langle \Phi \rangle \}$.

As we can see in Def. 3, each $i$-dimensional cell is an $n$-Gmap is obtained by an orbit using all the involutions except $\alpha_i$.

Definition 3 ($i$-cell). Let $G$ be an $n$-Gmap, and $d \in D$ be a dart. Given $i$, $0 \leq i \leq n$, the $i$-dimensional cell containing $d$, called $i$-cell and denoted by $c^i(d)$, is $\langle \alpha_0, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n \rangle (d)$.

Due to the definition of cells as sets of darts, the incident and adjacency relations on cells can easily be tested. Two distinct cells $c_1$ and $c_2$ are incident if $c_1 \cap c_2 \neq \emptyset$, and two $i$-cells $c_1$ and $c_2$ are adjacent if there is two darts $d_1 \in c_1$ and $d_2 \in c_2$ satisfying $d_1 = \alpha_i(d_2)$. When a dart $d$ belongs to an $i$-dimensional border, we have $\alpha_i(d) = d$ and we say that $d$ is $i$-free.

In the example of Fig. 1, face $f_3$ is described by $\langle \alpha_0, \alpha_1 \rangle (1) = \{1, 2, 3, 4, 5, 6\}$, edge $e_1$ by $\langle \alpha_0, \alpha_2 \rangle (13) = \{13, 14, 15, 16\}$, and vertex $v_1$ by $\langle \alpha_1, \alpha_2 \rangle (2) = \{2, 3, 4, 7, 14, 15, 24\}$. $v_1$ and $e_1$ are incident since $\langle \alpha_1, \alpha_2 \rangle (2) \cap \langle \alpha_0, \alpha_2 \rangle (13) = \{14, 15\} \neq \emptyset$. $f_1$ and $f_3$ are adjacent since $23 \in f_1$, $1 \in f_3$, and $\alpha_2(1) = 23$.

In this paper, the main operations used to simplify an $n$-Gmap are the removal operations (see [7, 6] for the definitions). Intuitively, removing a removable cell $c$ merges the two $(i+1)$-cells incident to $c$, without modifying the other cells.

Definition 4 (Removable cell). Let $G$ be an $n$-Gmap, $c$ be an $i$-cell of $G$. $c$ is removable if one of the two conditions is satisfied:

- $i = n - 1$; or $0 \leq i < n - 1$ and $\forall d \in c$, $\alpha_{i+1} \circ \alpha_{i+2}(d) = \alpha_{i+2} \circ \alpha_{i+1}(d)$. 

Removal Operations in \( n \)-Gmaps for Efficient Homology Computation

Fig. 1. Example of a 2G-map \( G = (D, \alpha_0, \alpha_1, \alpha_2) \). (a) A 2D cellular complex containing 3 faces; 9 edges and 7 vertices. (b) The 2G-map describing this cellular complex, having 24 darts (represented by numbered black segments). Two darts linked by \( \alpha_0 \) are drawn consecutively and separated by a gray segment (for example \( \alpha_0(19) = 20 \)), two darts linked by \( \alpha_1 \) share a common point (for example \( \alpha_1(20) = 21 \)), and two darts linked by \( \alpha_2 \) are drawn parallel, the gray segment over these two darts (for example \( \alpha_2(13) = 16 \)).

The notion of removable cell \( c \) is strongly related to the number of its \((i+1)\) incident cells, called the degree of \( c \) and denoted \( \text{degree}(c) \). A direct consequence of Def. 4 is that an \((i-1)\)-cell \( c \) of degree \( > 2 \) is not removable.

In the example of Fig. 1, all the edges are removable (since an \((n-1)\)-cell is always removable in an \( n \)-Gmap), vertex \( v_1 \) is removable while vertex \( v_2 \) not.

Removing edge \( e_1 \) merges faces \( f_1 \) and \( f_2 \) in one face having as boundary the boundary of \( f_1 \) plus the boundary of \( f_2 \) minus edge \( e_1 \).

To be able to compute homology of an \( n \)-Gmap, we need to have a boundary operator (defined in [5, 4]). The boundary operator is defined for \( n \)-Gmaps having orientable cells. Note that it is possible to represent a non-orientable object (e.g. a Klein bottle) with a \( n \)-Gmap having only orientable cells.

In the following we detail the notions of orientable cell and signed cell (cf.Defs. 5 and 6).

**Definition 5 (Orientable \( i \)-cell).** An \( i \)-cell \( c \) is orientable if \( c = e_1 \cup e_2 \) such that: \( \forall d \in c, \forall j, 0 \leq j \leq n, j \neq i : d \) is not \( j \)-free \( \Rightarrow d \) and \( \alpha_j(d) \) do not belong to the same set \( e_1 \) or \( e_2 \). \( c \) is non-orientable otherwise.

If \( c \) is orientable, then it can be partitioned in two sets of darts representing its two orientations and we can associate a value \(-1\) or \(+1\) to each of its dart, called a sign. In the following, we only consider \( n \)-Gmap having all its cells signed.

**Definition 6 (Signed \( i \)-cell).** Let \( c \) be an orientable \( i \)-cell. The corresponding signed \( i \)-cell is \( c \) together with a sign for each of its dart \( d \), denoted \( \text{sg}^i(d) \):

- \( \text{sg}^i(d) = -\text{sg}^i(\alpha_j(d)) \forall j : 0 \leq j < i \) such that \( d \) is not \( j \)-free;
- \( \text{sg}^i(d) = \text{sg}^i(\alpha_j(d)) \forall j : i < j \leq n \).

For defining a boundary operator on \( n \)-Gmaps, we first define the signed incidence number between two cells \( c^i \) and \( c^{i-1} \) which describes the number of times that \( c^{i-1} \) appears in the boundary of \( c^i \).
Definition 7 (Signed incidence number). Let \( \{p_j\}_{j=1}^k \) be a set of darts s.t. the orbits \( \{ (\alpha_0, \cdots, \alpha_{i-2}) (p_j) \}_{j=1}^k \) make a partition of \( (\alpha_0, \cdots, \alpha_{i-1}) (d) \). The signed incidence number between \( c^i \) and \( c^{i-1} \) is defined by

\[
(c^i : c^{i-1}) = \sum_{p_j, j=1, \cdots, k} s_g^i(p_j)_s g^{i-1}(p_j).
\]

Note that this definition is equivalent to the one given in [5]. Now the boundary operator \( \partial_G \) of any i-cell \( c \) is defined as \( \partial_G(c) = \sum_{c} (c : c') c' \), where \( c' \) are \( (i-1) \)-cells incident to \( c \). The boundary operator \( \partial_G \) satisfies \( \partial_G \circ \partial_G = 0 \) when involutions \( \alpha_i \) are without fixed points for \( 0 \leq i \leq n-1 \). Moreover, we have proven in [4] that the homology defined on \( n \)-Gmaps by this boundary operator is equivalent to the simplicial homology of the associated quasi-manifolds when the homology of the canonical boundary of each \( i \)-cell is that of an \((i-1)\)-sphere, and when \( \forall d \in D, \forall i \in \{0, \ldots, n\} \), \( d \) is i-free or \( \alpha_i(d) \not\in \{\alpha_0, \ldots, \alpha_{i-2}, \alpha_{i+2}, \ldots, \alpha_n\} (d) \). In the following, all the considered \( n \)-Gmaps satisfied these conditions.

3 Removal Operations Preserving Homology

In this section, we prove that removing a degree two cell or a dangling cell preserves the homology of the \( n \)-Gmap.

3.1 Chain Complexes and Chain Contractions

Let \( S = \{S_q\}_q \) be a graded. A \( q \)-chain is a finite formal sum of elements of \( S_q \) with coefficients in \( \mathbb{Z} \). Let \( C_q(S) \) denote the group of \( q \)-chains of \( S \). The chain complex \((C_*(S), \partial)\) is the chain group \( C_*(S) = \{C_q(S)\}_q \) together with a boundary operator \( \partial \). Given an \( n \)-Gmap \( G \), let \( S_G \) be the set of all the cells of \( G \). \((C_*(S_G), \partial_G)\) is the chain complex associated to \( G \).

A chain contraction [13] of \((C_*(S), \partial)\) to \((C_*(S'), \partial')\) is a triple \((f = \{f_q : C_q(S) \to C_q(S')\}_q, g = \{g_q : C_q(S') \to C_q(S)\}_q, \phi = \{\phi_q : C_q(S) \to C_q(S')\}_q\) such that: (i) \( f \) and \( g \) are chain maps; i.e. \( f_q \circ \partial_q = \partial'_q \circ f_q \) and \( g_q \circ \partial_q = \partial'_q \circ g_q \) for all \( q \); (ii) \( \phi \) is a chain homotopy of \( \text{id}_{C_*(S)} = \{\text{id}_q : C_q(S) \to C_q(S)\}_q \) to \( g \circ f = \{g_q \circ f_q : C_q(S) \to C_q(S)\}_q \); i.e. \( \phi_{q-1} \circ \partial_q + \partial'_q \circ \phi_q = \text{id}_q - g_q \circ f_q \) for all \( q \); (iii) \( f \circ g = \text{id}_{C_*(S')} \). If a chain contraction of \((C_*(S_G), \partial_G)\) to \((C_*(S_{G'}), \partial_{G'})\) exists, then the \( n \)-Gmaps \( G \) and \( G' \) have isomorphic homology groups.

3.2 Degree two cells

Proposition 1. Let \( c \) be an i-cell in an \( n \)-Gmap. If \( c \) is removable and degree two cell, then there are two \((i+1)\)-cells \( a \) and \( b \) satisfying: \(|(a : c)| = |(b : c)| = 1 \) and for all other \((i+1)\)-cells \( c' \), \((c' : c) = 0 \).

Proof. Since \( c \) is degree two, there are two \((i+1)\)-cells \( a \) and \( b \) that are incident to \( c \). For these two cells, we have \( c \in \partial_G(a) \) and \( c \in \partial_G(b) \). So, \((a : c) \neq 0 \) and \((b : c) \neq 0 \). If \(|(a : c)| > 1 \), contradiction with removal property, thus \(|(a : c)| = 1 \) (and the same for \(|(b : c)| = 1 \)). For all other \((i+1)\)-cells \( c' \), \( c' \) is not incident to \( c \) otherwise the degree was greater than two. Thus \((c' : c) = 0 \). \( \square \)
Proposition 2. Let \( c \) be an \( i \)-cell in an \( n \)-Gmap. If \( c \) is a removable degree two cell, and if each \( j \)-cell \( e \) incident to \( c \), is after the removal of \( c \) a \( j \)-cell equal to \( e \setminus c \), then homology is preserved after the removal of \( c \).

Note that the removal of a cell may induce removal of other cells (for example, it is possible to build a sphere made of one vertex, one degree two edge and two faces. Removing the edge would suppress all the darts and so the vertex and the two faces). The second condition ensures that only one cell is removed.

Proof. Let \( (C_*(S_G), \partial_G) \) be the chain complex associated to \( G \). Since \( c \) is degree two, there are two \((i+1)\)-cells \( a \) and \( b \) that are incident to \( c \). The set \( S_G' \) of the cells of the \( n \)-Gmap \( G' \) obtained after removing the cell \( c \) consists in \( S_G \setminus \{a, b, c\} \cup \{a'\} \) where \( a' \) is the resulting \((i+1)\)-cell from merging the two cells \( a \) and \( b \). Since, by Prop. 1, \(|(a : c)| = |(b : c)| = 1\) and for all other \((i+1)\)-cells \( c', (c' : c) = 0 \), we can construct a chain contraction \((f, g, \phi)\) of \((C_*(S_G), \partial_G)\) to \((C_*(S_G'), \partial_G')\) as follows:

\[
f(x) = \begin{cases} 
  c - (b : c)\partial_G(b), & \text{if } x = c, \\
  a', & \text{if } x = a, \\
  0, & \text{if } x = b, \\
  x, & \text{otherwise}; 
\end{cases} \quad g(x) = \begin{cases} 
  a - (a : c)(b : c)b, & \text{if } x = a', \\
  x, & \text{otherwise}; 
\end{cases} \quad \phi(x) = \begin{cases} 
  (b : c)b, & \text{if } x = c, \\
  0, & \text{otherwise}. 
\end{cases}
\]

To check that \((f, g, \phi)\) is a chain contraction is left to the reader. Moreover, we know that each \( j \)-cell incident to \( c \) is preserved by the removal operation. Then \( G \) and \( G' \) have isomorphic homology groups. \( \square \)

3.3 Dangling cells

Let \((C_*(S), \partial)\) be a chain complex. Let \( s, t \in S \) such that \(|(s : t)| = 1\) and \(|(s' : t)| = 0\) for any \( s' \in S, s' \neq s \). If we remove \( s \) and \( t \) from \( S \) to get \( S' \), we obtain another chain complex \((C_*(S'), \partial')\) which is called an elementary collapse of \( S \). A chain contraction of \((C_*(S), \partial)\) to \((C_*(S'), \partial')\) is given by

\[
f(x) = \begin{cases} 
  0, & \text{if } x = s, \\
  t - (s : t)\partial(s), & \text{if } x = t, \\
  x, & \text{otherwise}; 
\end{cases} \quad g(x) = x; \quad \phi(x) = \begin{cases} 
  (s : t)t, & \text{if } x = t, \\
  0, & \text{otherwise}. 
\end{cases}
\]

Therefore an elementary collapse preserves homology. A subset of \( S \) is collapsible if they can all be removed from \( S \) in a sequence of elementary collapses.

Let \( c \) be a \( k \)-cell, the closure of \( c \), denoted \( \overline{c} \), is the set made of \( c \) plus all the \( j \)-cells, \( 0 \leq j < k \) that are incident to \( c \). The closure of a set \( S \) of cells, denoted \( \overline{S} \), is the union of the closures of all the cells of \( S \).

Definition 8 (Dangling cell). Let \( c \) be an \( i \)-cell. We denote \( C \) the set of \((i-1)\)-cells of \( \partial_G(c) \), and \( B = \{c' \in \partial_G(c) | \text{degree}(c') > 1\} \). \( c \) is dangling if \( c \) is orientable, its degree is 1, \( \{c\} \cup \overline{C} \setminus B \) is collapsible, and each \( j \)-cell \( e \in B \), is after the removal of \( c \) a \( j \)-cell equal to \( e \setminus c \).
**Proposition 3.** Let \( c \) be an \( i \)-cell in an \( n \)-Gmap. If \( c \) is removable and dangling cell, then its removal preserves the homology of the \( n \)-Gmap.

*Proof.* Removing \( c \) will remove also all the cells in \( C \setminus B \) because these cells are included in \( c \) (i.e. their set of darts is included in the set of darts of \( c \)). As \( \{c\} \cup C \setminus B \) is collapsible, and as all the other cells are preserved, the homology of the \( n \)-Gmap is preserved by the definition and property of collapsible. \( \square \)

### 3.4 Simplification Preserving Homology

The main principle of the simplification algorithm consists in removing successively all the degree two cells and all the dangling cells for all the dimensions starting from \((n-1)\)-cells to 0-cells. For that, we start to define Algo. 1 which simplifies all the \( i \)-cells of a given \( n \)-Gmap for a given dimension \( i \).

**Algorithm 1:** Simplification of \( i \)-cells.

- **Input:** An \( n \)-Gmap \( G \).
- **Output:** Simplify all the \( i \)-cells of \( G \) while preserving the same homology.

```plaintext
foreach i-cell c of G do
    if c is removable and the degree of c is 2 then
        Remove f;
    else if c is removable and c is a dangling cell then
        push(P, c);
        repeat
            c ← pop(P);
            push in P all the dangling i-cells adjacent to c;
            Remove c;
        until empty(P);
```

In this algorithm, we consider successively each \( i \)-cell \( c \), and there are three possible cases. First, if \( c \) is not removable, then we are sure that \( c \) cannot be removable in a future step of the algorithm. Indeed, we only remove \( i \)-cells and this does not modify the \((i+1)\)-cells incident to \( c \). Second, if \( c \) is removable and its degree is two, we remove \( c \). Third, if \( c \) is removable and dangling, we also remove \( c \), but now we have to reconsider all the \( i \)-cells adjacent to \( c \). Indeed, these cells can possibly become dangling due to the removal of \( c \). At the end of the loop, we have considered all the \( i \)-cells and removed all the degree 2 cells and the dangling cells that were removable.

Now the global simplification method consists only in simplifying all the \( i \)-cells of the \( n \)-Gmap for all the cells by decreasing dimensions. We have to work in decreasing dimensions because the removal of an \( i \)-cell modifies the degree of all the incident \((i-1)\)-cells. At the end of the global simplification algorithm, we have removed all the removable cells of degree 2 or dangling. By using Props. 2
Removal Operations in $n$-Gmaps for Efficient Homology Computation

and 3, we know that the final $n$-Gmap obtained after all the removals has the same homology than the initial $n$-Gmap.

4 Experiments

In order to illustrate the interest of our simplification algorithm, we show results on homology generator computation for the five objects shown in Fig. 2. Objects (a), (b) and (c) are described by 2-Gmaps; objects (d) and (e) are described by 3-Gmaps.

![Fig. 2. (a) 2-torus. (b) Klein bottle. (c) pinion. (d) tower. (e) Menger sponge.](image)

<table>
<thead>
<tr>
<th>Object</th>
<th>Initial</th>
<th>Simplified</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># cells</td>
<td>Homology computation</td>
</tr>
<tr>
<td></td>
<td>0 1 2 3</td>
<td>0 1 2 3</td>
</tr>
<tr>
<td>2-torus</td>
<td>404 802 396</td>
<td>14Mb 5.76s</td>
</tr>
<tr>
<td>Klein</td>
<td>900 1800 900</td>
<td>74Mb 128.47s</td>
</tr>
<tr>
<td>Pinion</td>
<td>470 701 231</td>
<td>11Mb 3.56s</td>
</tr>
<tr>
<td>Tower</td>
<td>906 1856 952 4</td>
<td>85Mb 140.97s</td>
</tr>
<tr>
<td>Menger</td>
<td>896 2304 1728 400</td>
<td>159Mb 372.50s</td>
</tr>
</tbody>
</table>

Table 1. Results of our experiments. We give the number of cells (columns # cells) for initial objects, and after the simplification algorithm. The last column gives the time of the simplification step. The two columns Homology computation give the memory space and the time of the homology generators computation (0s means less than $10^{-6}$s).

To compute the homology generators, we iterate through all the cells of the $n$-Gmap and we compute incidence matrices (which describes the boundary of the cells) using the incidence number definition. Then we reduce incidence matrices into their Smith-Agoston normal form for computing homology generators [3]. Compared to the classical Smith normal form, the specificity of the Agoston
reduced normal form is that for a given dimension $d$, the basis of the boundaries $B_p$ is a subset of the basis of cycles $Z_p$, thus the quotient group $H_p = Z_p / B_p$ can directly be obtained by simply removing from $Z_p$ the boundaries of infinite order. Note that several optimizations exist for the reduction of incidence matrices [15, 8]. Even if they can be used, we do not use them here as we focus on showing the improvement obtained with the simplification process.

The computation of homology generators was implemented in Moka [16], a 3D topological modeler based on 3-Gmap. For this reason, the computation of homology generators is limited to 2D and 3D cases, but all the functions are generic in any dimension. The results are presented in Table 1, where the simplification step widely reduces the number of cells. On the last column one can see that the simplification step is very fast. Memory space is also reduced as the size of incidence matrices are directly linked to the number of cells.

![Figure 3](image)

**Fig. 3.** The generators of $H_1$ (in red) computed on simplified objects, and projected on initial objects (drawn in grey). (a) Klein bottle. (b) Tower. (c) Menger sponge.

Lastly, we can see in Fig. 3 the different generators of $H_1$ obtained for some objects. By using the definition of removal operations, we are able to project the generators of the simplified object on the initial one (by using a similar technique as in [14, 9]).

We have made a second type of experiments in order to compare our approach with other existing methods. To our knowledge there is no other general method which compute homology generator of cellular objects. Thus we compare our solution with Chomp and RedHom [1, 2] which compute homology generators of cubical complexes. We chose these two methods since the two softwares are publicly available. However, it must be noticed that representing a cubical complex by a $n$-Gmaps is not efficient since a cube is described by 48 darts; the interest of cellular model is precisely to represent non regular subdivisions. In order to test the scale up property of the three methods, we chose three objects (see in Fig. 4(a), (b) and (c)), and multiply the size of each voxel by 4 to 9 for the first two objects, and by 2 to 7 for the last object which contains more voxels.
Removal Operations in n-Gmaps for Efficient Homology Computation

We can see in Fig. 4 the time required to compute homology generators of each object by the three compared methods. These results are really encouraging for our method which obtain the best computation time for the first two objects, with an important gain for the second one. For the third object, Chomp, and Moka performances are very similar (even if Chomp is a little bit quicker), while RedHom is really faster. In this last case, Chomp, and Moka have similar computation time than for the two first objects, while RedHom is extremely fast. We suppose there is an optimization allowing to remove directly some block of voxels. Indeed, the upper part of the last object is composed by a full block of voxels. This kind of improvement can also be made for our method.

These experiments show that our method is very competitive since it is not optimized for a specific type of subdivision but it is generic for any cellular complex. Thus its main interest is its genericity and we can conclude from this comparison that this is not to the detriment of the efficiency. Moreover, we can improve our results by adding a thinning pre-processing step that reduces the number of voxels while preserving the homology.

5 Conclusion

In this paper, we have presented an algorithm that simplifies an n-Gmap while preserving its homology. For that, it removes degree two cells and dangling cells.
Then we can compute homology on the reduced \( n \)-Gmap and project the generator on the original object. Some results show the interest of the simplification step, both in memory space and in computation time.

Some questions are still open. The first question is about the conditions on removed cells. Is it possible to remove some other type of cells while preserving the homology? The answer is no in 2D and 3D, but still open in higher dimension. This question is related to the definition of the minimal generalized map having the same homology. In 3D, to obtain this minimal map, we need to use another type of operation (fictive edge shifting). Thus we would like to study the extension of this operation in higher dimension to define the minimal \( n \)-Gmap.

References