Computing Homology for Surfaces with Generalized Maps: Application to 3D Images

Guillaume Damiand[1], Samuel Peltier[2⋆], and Laurent Fuchs[1]

[1] SIC - bât. SP2MI, Bvd M. et P. Curie
BP 30179, 86962 Futuroscope Chasseneuil Cedex - France
{damiand,fuchs}@sic.univ-poitiers.fr
[2] PRIP - Vienna University of Technology
Favoritenstr. 9/1832, A-1040 Vienna - Austria
sam@prip.tuwien.ac.at

Abstract. In this paper, we present an algorithm which allows to compute efficiently generators of the first homology group of a closed surface, orientable or not. Starting with an initial subdivision of a surface, we simplify it to its minimal form (minimal refers to the number of cells), while preserving its homology. Homology generators can thus be directly deduced from the minimal representation of the initial surface. Finally, we show how this algorithm can be used in a 3D labelled image in order to compute homology of each region described by its boundary.

Keywords. topological features, homology generators, generalized maps, minimal subdivision.

1 Introduction

In many domains of computer graphics (geometric modeling, computational geometry, image structuring,...), combinatorial structures are used to describe objects subdivided into cells of different dimensions (vertices, edges, faces, volumes,...). A common problem in each domain is to characterize structural (topological) properties of handled objects. Homology is a topological invariant, classically studied in algebraic topology, which characterizes an object by its "holes" in each dimension, which corresponds to its connected components in dimension 0, its tunnels in dimension 1, its cavities in dimension 2 ; this notion of hole can be generalized in any dimension. For each dimension d, the number of $d-$dimensional holes of a given object is called its d^{th} Betti number.

Generalized maps [1, 2], are a combinatorial cellular structure which can be used to build and handle quasi-manifolds. In this work, generalized maps are used to compute a minimal cell decomposition (called *minimal map*) of a closed surface.

Kaczynski et al. proposed a general algorithm [3] for reducing the number of cells of a given object, without changing its homology. If the object is orientable, this general algorithm provides its Betti numbers.

⋆ Partially supported by the Austrian Science Fund under grants S9103-N04.

In this paper, we adapt this general algorithm to the model of generalized maps, and we extend it (by adding an operation, called shifting edge operation) in order to compute a minimal map for any surface, orientable or not. Moreover, we show that a set of generators of the first homology group can directly be deduced from the minimal map.

In Section 2, basic notions related to generalized maps and homology groups are recalled. The removal operations, which are used to compute a minimal map are introduced. In Section 3, the algorithm for computing a minimal map is detailed, and its complexity is discussed. Finally, we prove that our algorithm provides a minimal object with the same homology that the initial one. In Section 4, we show how to use the minimal map algorithm for characterizing each region in a 3D labeled image.

2 Recalls

2.1 Generalized Maps

A subdivision of a 3D topological space is a partition of the space into 4 subsets whose elements are *cells* of dimension 0, 1, 2 and 3 (respectively called vertices, edges, faces and volumes). In the following, an i-cell will denote a cell of dimension i. Border relations are defined between these cells and the border of an i-cell is a set of *(j<i)*-cells . Two cells are *incident* if one belongs to the border of the other, and two i-cells are *adjacent* if they are both incident to a common *(j<i)*-cell.

For 3D subdivisions, incidence and adjacency relations can be represented using 3-dimensional generalized maps (3-*G-maps*) [2]. Intuitively, a 3D generalized map can be obtained by successive decompositions of a 3D object. We first decompose the volumes of this object, then the faces of these volumes, the edges of these faces, and then the vertices of these edges. The elements obtained from the last decomposition are called *darts* and are the basic elements of the generalized map definition.

To obtain the map, adjacency relations between i-cells (denoted α_i) are reported onto darts. *Involution*[3] α_0 connects two vertices incident to a same edge, face and volume, α_1 connects two edges incident to a same vertex, face and volume, and so on for all dimensions. These α_i have to verify some particular properties in order to ensure the validity of the represented subdivision (see Fig. 1 for an example and [2] for generalized map definition).

Within the generalized map framework, all cells are implicitly represented through the notion of *orbit*. Given, $\{p_1, \ldots, p_j\}$, a set of (darts) involutions, and a dart d, an orbit $< p_1, \ldots, p_j > (d)$ is the set of darts that can be reached with a breadth-first search algorithm, starting with d, and using all combinations of p_i $\forall k, 1 \leq k \leq j$. Based on the cells definition, we can retrieve the classical *cell degree* notion. The degree of an i-cell c is the number of distinct *(i+1)*-cells incident to c.

[3] An involution f on S is a one to one mapping from S onto S such that $f = f^{-1}$.

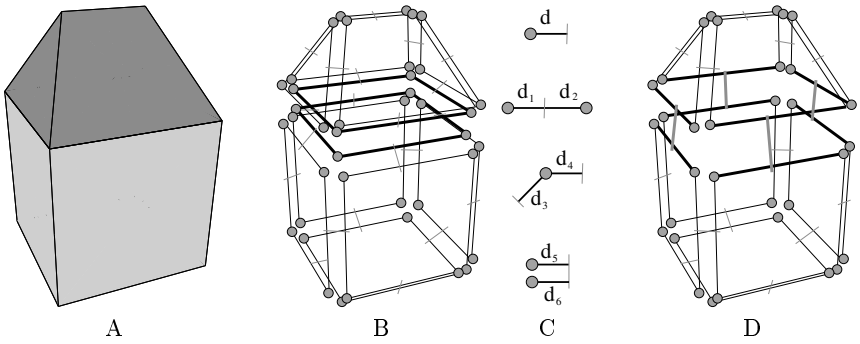

Fig. 1. Example of a 3D generalized map. (A) A 3D object. (B) Implicit representation of the corresponding generalized map. (C) Explanations of the graphical representation. d: representation of a dart. d_1 and d_2: two darts in relation by α_0. d_3 and d_4: two darts in relation by α_1. d_5 and d_6: two darts in relation by α_2. α_3 is not represented on figures but can be deduced from the shape of objects. (D) Map obtained from (B) by removing the black face. Adjacent faces of the initial removed face are joined by modifying α_2 relations.

2.2 Removal Operations

Removal operations are the basic operations used during the construction of a minimal map. The removal of an i-cell c (called i-removal of c) leads to the merging of the two *(i+1)*-cells incident to c (see an example of *2*-removal in Fig. 1). For 3D subdivisions, i-removal operations are defined for $i = 0, 1, 2$. A more complete description of the removal operation can be found in [4].

Any face of a 3-G-map can be removed, since the degree of a face in a 3D subdivision is always equal to one or two. The face removal operation consists mainly in locally modify the α_2 relation for each dart that belongs to the neighborhood of the removed face (all removal operations are based on a similar principle).

The 1-removal (edge removal) can only be applied for edges whose degree is one or two. Otherwise it is not possible to automatically decide how to connect the faces incident to the removed edge. This operation is achieved in a similar way than for face removal. In this case, α_1 relations are modified.

The 0-removal (vertex removal) is similar to 1-removal, but α_0 is modified.

2.3 Homology

In this part, basic homology notions are recalled; interested readers can find more details in [5]. The notion of homology can be defined in an algebraic way using the sets of $i-$cells used to describe the subdivision of the surface. Within this context, a *p-chain* (i.e. a chain of dimension p) is defined as a formal sum of p-cells. From this, the group C_p of p-chains is defined. The boundary of a p-chain is defined as the sum of the boundaries of its p-cells. Note that the boundary of a p-chain must be a $p-1$-chain. The set of p-chains which have a null boundary

(i.e. *p-cycles*) is a subgroup of C_p (denoted Z_p). The set of p-chains which are boundary of a $p+1$-chain (i.e. *p-boundaries*) form a subgroup of C_p (denoted B_p).

An essential property is that the boundary of any boundary is null. Hence, every boundary is a cycle and B_p is contained in Z_p. Two p-cycles z_1 and z_2 are *homologous* if their difference is a boundary, i.e. there is a $p+1$-chain f such that $z_1 = z_2 + \partial f$. From this, an equivalence relation can be defined and the homology class of z is the set $\{z+b \mid b \in B_p\}$. The *homology group of dimension* p, denoted H_p, is defined as the quotient group Z_p/B_p, and its elements are the homology classes. For a group G, a *set of generators* is a maximal subset S of elements of G, such that every element of G can uniquely be defined as a linear combination of elements of S.

3 Minimal Map Preserving Homology

3.1 Simplification Algorithm

The goal of Algorithm 1 is to simplify a generalized map M representing a closed surface into its minimal representation (minimal refers to the number of cells) while preserving its homology. Moreover, the set of edges obtained in the minimal map of M is a set of generators of the first homology group of initial surface described by M.

The principle of the algorithm is to remove as much cells as possible while preserving the homology of the initial object. For that, we use a similar approach than the algorithm described in [3]. As mentioned in the previous reference, this general algorithm may stop without a minimal object (i.e. the same object can be represented with less cells). We solve this problem by adding an operation (fictive edge shifting) that ensures that the obtained object is minimal.

This algorithm proceeds in two successive steps. The first one (line 1) removes useless edges. For that, each edge e is successively processed in any order. If e is a degree two edge, then it can be removed without modifying the homology (see the proof in Section 3.3). Indeed this operation consists in merging two distinct adjacent faces in a unique one.

If e is not a degree two edge, there are two distinct cases depending if e is a degree one edge dangling or not dangling. In the first case, e can be removed without modifying the homology. In the second case, e can not be removed. Indeed, its removal will disconnect its incident face in two connected components and this will involve a modification of the homology of the object.

When e is a dangling edge, e needs to be removed (like for degree two edges), but we eventually need to re-consider the edge adjacent to e. Indeed, if some edges form a path, when we consider an edge in the middle of the path, this edge is a not dangling degree one edge and thus it is not removed. But when the current edge is at the end of the path, it is a dangling edge and thus it is removed. But now, the previous edge in the path became a dangling edge and the process can be iteratively applied for each edge of the path. To process this

Algorithm 1: Simplification of a surface in its minimal representation.

Input: A generalized map M representing a closed surface
Output: The minimal map corresponding to M

```
1 foreach edge e of the map do
    if the degree of e is 2 then
      Remove e;
    else
      while e is a dangling edge do
        e' ← one edge adjacent to e;
        Remove e;     e ← e';
2 foreach vertex v of the map do
    if the degree of v is 2 then
      Remove v;
    else if it exists an edge e non-loop incident to v then
      Shift all edges (except e) incident to v along e;
      Remove v;
```

case, when a dangling edge e is removed, we keep the unique edge e' adjacent to e, and e is removed. Now, if e' is a dangling edge, we reiterate the process. Otherwise, we are at the end of the path and we can go back to the first loop and consider another edge.

The second step of the simplification algorithm (line 2) removes useless vertices. For that, it runs through each vertex of the surface. If the current vertex is a degree two vertex, it is removed. Indeed, such removal involves to merge its two incident edges. If the degree of the vertex is different from two, a special processing must be applied before its removing. We first take an edge e incident to the current vertex which is not a loop (an edge is a non-loop edge iff it is incident to two distinct vertices). If such an edge does not exist, the subdivision is thus composed uniquely by loops and thus the current vertex is the unique vertex of the subdivision. This case occurs only for the last vertex of the map when the process is finished and the minimal map is obtained. Otherwise, all the edges incident to the current vertex are shifted along e. The edge shifting operation consists in pushing an edge along another one (see Fig. 4). After this operation, since all edges incident to the current vertex are pushed, the vertex degree is now equal to one and the vertex can be removed without modifying the homology of the object.

We can see in Fig. 2 and Fig. 3 two examples: the first one with an initial surface representing a double-torus and the second one with an initial surface representing a Klein bottle. For these examples, we have represented in (A) the initial subdivision, in (B) the map obtained after the first simplification step and in (C) the final map which is minimal.

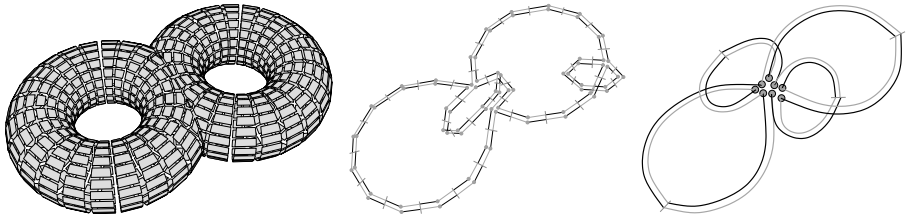

Fig. 2. An example of minimal map corresponding to a double torus. (A) The initial surface. (B) After edge removals. (C) After vertex removals.

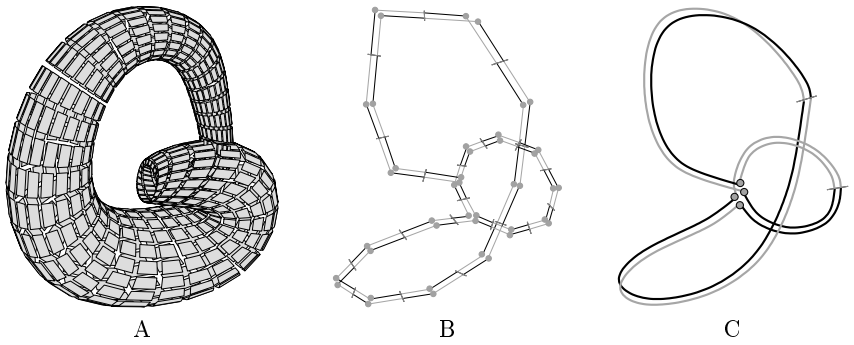

Fig. 3. An example of minimal map corresponding to a Klein bottle. (A) The initial surface. (B) After edge removals. (C) After vertex removals.

3.2 Complexity

The complexity of Algorithm 1 is equal to $O((3-\chi)\times n)$ where n is the number of cells and χ is the Euler characteristic of the surface.

Each edge is processed exactly once during the first loop of the algorithm. The test on the edge degree can be achieved in two steps. First locally, in order to exclude all the edges with degree strictly greater than two (by testing if $\alpha_{23}(d) \neq \alpha_{32}(d)$). Then, there are two distinct cases to distinguish depending if the edge is a degree one or two edge. Indeed, the edge of the first case has to be removed while the edge of the second case does not. But these two cases can not be characterized locally since we need to know if both side of the edge belong to the same face.

In order to solve this problem, we use union-find trees [6] which allow to represent efficiently disjointed sets. This structure is handled by two operations: *find* which returns, given an element, the representative of the set, and *union* which allows to merge two sets. The amortized cost of a series of m union-find operations on n elements can be done in time $O(n.\alpha(m,n))$ with $\alpha(m,n)$ being the inverse Ackermann function which grows extremely very slowly, and

which is less than 5 in practical cases (see [6] for the demonstration about the complexities).

We link each dart of a same face with an union-find tree representing the face. When we remove an edge, we merge both corresponding trees by using the *union* operation. The test if d and $\alpha_2(d)$ belong to the same face is simply achieved by testing if *find(d)* is equal to *find*($\alpha_2(d)$).

To summarize the first step, the cost of the test on the edge degree can be bounded by 5, the cost of the test if the edge is a dangling edge is in constant time and the edge removal is also achieved in $O(1)$.

For the second loop of Algorithm 1 "*While e is a dangling edge*", in the worst case, we can run through all the edges of the map. This case case occurs when the map is a unique path, and if the first loop start with a dart in the middle of the path. In this case, each edge is considered only twice. The first time is during the first loop, without removing it since each edge in the middle of the path is a degree one edge and not a dangling one. The second time is during the second loop, after we have processed an edge at the extremities of the path. Since each edge of the second loop is removed, it will never be considered once again.

The second step of the simplification algorithm considers each vertex exactly once. The test on the degree of v is achieved by running through all the darts of the vertex. Each dart is only considered once. Moreover, the vertex removal operation is also achieved in linear time according to the number of darts of the removed cell.

The edge shifting operation is achieved linearly in number of edges incident to the removed vertex. The subdivision in which we simplify vertices is the result of the edge simplification. Thus, this subdivision is always composed of one face and n_l loops, where n_l is equal to $2-\chi$ with χ being the Euler characteristic of the surface (indeed, the minimal map has 1 vertex, 1 face and n_l edges and we can compute n_l by using the Euler formula).

Thus, given a vertex v, there are at most $2-\chi$ edges incident to v and thus the complexity of the second step of Algorithm 1 is in $O((2-\chi)\times n)$. By adding the cost of the first step $O(n)$, we obtain the final complexity $O((3-\chi)\times n)$ (with $\chi \leq 2$).

3.3 Proof of the Algorithm

In order to ensure the validity of our algorithm, we prove that the obtained map is minimal and the homology is preserved during the simplification process.

It can be proven that the resulting map is minimal by contradiction. The initial map represents a closed surface. The map obtained after the first step of our algorithm is made of an unique face. Indeed, otherwise there are at least two faces and the edge between these two faces is a degree two edge. This contradicts the simplification algorithm which removes each degree two edge. Moreover, this map does not contain dangling edge since they are all removed during the simplification.

The map obtained after the second step of our algorithm is made of an unique vertex. Indeed, otherwise, there are at least two vertices, and since the map is

connected, one edge non-loop between these two vertices. This contradicts the fact that in our algorithm, we remove each vertex after to have shifted all edges incident to it.

This proves that the resulting map can is minimal in number of faces and vertices. This map is thus obviously minimal in number of edges since, given a number of faces and vertices, the number of edges is fixed depending on the Euler characteristic of the surface.

In order to prove that the simplification process preserves the homology, we use the works of [3] and make a parallel between their elementary reductions and our simplification algorithm. In this work, the authors present their algorithm starting with a simplicial complex, and simplify it by using *interior face reduction*. Our removal operation is equivalent to this reduction which consists in taking a common i-face c of exactly two $(i+1)$-simplices a and b, and which deletes c and replaces b and c by a cell which represents their union. It is proven that interior face reduction preserves homology (see [3]) and thus we can conclude that removal operations too.

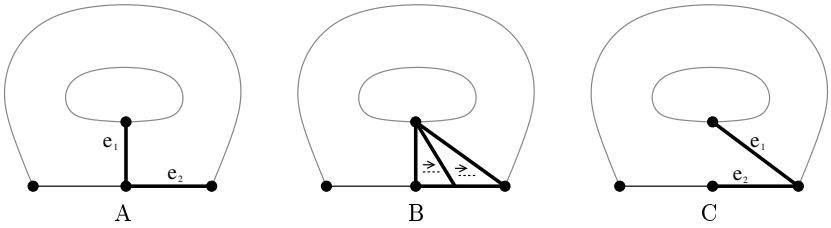

Fig. 4. An example of edge shifting. (A) The initial configuration. We are going to shift edge e_1 along edge e_2. (B) Edge shifting seen as a continuous mapping. (C) Final configuration.

In this work, we use an additional operation, edge shifting, in order to obtain a minimal representation. Proving that this operation preserves homology can easily be done by considering this operation as a continuous mapping between the initial configuration (e.g. Fig. 4A) and the configuration obtain after the edge shifting (e.g. Fig. 4C).

The edges of the resulting minimal map, are homology groups generators: they are cycles because they are self-loops and then their border is zero, and each edge can not be bordered because in this case, the map would have at least two faces. Moreover, these edges are not two by two homologous: if two edges e_1 and e_2 are homologous, then their exist a border ∂f of (at least) a face such that $e_1 = e_2 + \partial f$. This is not possible as we only consider minimal map of closed surfaces.

Moreover, in our case, the combinatorial structure used (i.e. generalized maps) allows us to distinguish between free generators (generators of free groups of the homology type) and torsion generators (generators of torsion groups of

the homology type). This is due to the manner of the darts are proceeded when we run through the $< \alpha_0 \circ \alpha_1 >$ orbit (black darts on figures 2 and 3). Let d be a dart of a given edge, if $\alpha_2(d)$ is in the same orbit then the edge is a torsion generator, if not then the edge is a free generator. This way to recognize torsion generators is valid because we are in the case of quasi-manifolds and then torsion coefficients cannot be more than 2. Fig. 3C shows example of free and torsion generators. It can also be noted that if we count free generators in each dimension, we obtain the Betti numbers of the surface.

4 Application for 3D Images

Given a 3D labeled image, it is possible to compute the minimal map which represents all the regions contained in the image and all the incidence and adjacency relations between these regions [7, 8]. It has been shown in many works (for example [9–13]) that using structures based on combinatorial (or generalized) maps gives an efficient framework for image processing algorithms. In these algorithms, it is often necessary to compute some features on regions, either to characterize them for example in an recognition process, or to use these features during a segmentation process.

We can use our simplification algorithm in order to compute homology generators of each region contained in a 3D labeled image. This is particularly interesting in order to characterize these regions by their boundaries. For that, we start from the minimal map which represents all the regions of the image (called topological map, see [14, 12] for a more detailed description). Since our algorithm works only for one surface (for the moment), we can not compute directly the generators on the topological map since it represents several surfaces.

In order to characterize a given region R, we first extract its surfaces. This is achieved easily by running through the topological map and extracting only surfaces belonging to R. This can be achieved with a breadth-first search algorithm, so linearly in number of darts of the surface. Region R can have one or several surfaces. Indeed, each region is always represented by one external boundary, but it can also be represented by some internal boundaries if a region has some included regions. But since these surfaces are disconnected, we can process each of them by our simplification algorithm and thus obtain generators for each surface.

5 Conclusion

In this paper we have presented an algorithm for computing the minimal map of a generalized map representing a subdivision of a closed surface. We have shown that a set of homology generators (free or torsion ones) can directly be deduced from the minimal map.

We have proven that the minimal map algorithm, based on the general algorithm proposed by Kaczynski et al. preserves the homology of an object represented by a G-map. Moreover, the complexity of the minimal map algorithm is

$O((3-\chi)\times n)$, where n the number of cells and χ is the Euler characteristic of the surface. An application of these results have been proposed for 3D images: characterizing a region of voxels by a set of generators of the first homology group of its boundary.

In future works, we plan to extend the minimal map algorithm for G-maps with boundaries and for computing a minimal map that preserves homology for higher dimension objects.

References

1. Lienhardt, P.: Topological models for boundary representation: a comparison with n-dimensional generalized maps. Commputer Aided Design **23** (1991)
2. Lienhardt, P.: N-dimensional generalized combinatorial maps and cellular quasi-manifolds. International Journal of Computational Geometry and Applications **4** (1994) 275–324
3. Kaczynski, T., Mrozek, M., Slusarek, M.: Homology computation by reduction of chain complexes. Computers & Math. Appl. **34** (1998) 59–70
4. Damiand, G., Lienhardt, P.: Removal and contraction for n-dimensional generalized maps. In: Discrete Geometry for Computer Imagery. Number 2886 in Lecture Notes in Computer Science, Naples, Italy (2003) 408–419
5. Hatcher, A.: Algebraic Topology. Cambridge University Press (2002) available on http://www.math.cornell.edu/∼hatcher/AT/ATpage.html.
6. Tarjan, R.: Efficiency of a good but not linear set union algorithm. Journal of the ACM **22** (1975) 215–225
7. Braquelaire, J.P., Desbarats, P., Domenger, J.P., Wüthrich, C.: A topological structuring for aggregates of 3d discrete objects. In: Workshop on Graph-Based Representations in Pattern Recognition, Austria, IAPR-TC15 (1999) 193–202
8. Bertrand, Y., Damiand, G., Fiorio, C.: Topological map: Minimal encoding of 3d segmented images. In: Workshop on Graph-Based Representations in Pattern Recognition, Ischia, Italy, IAPR-TC15 (2001) 64–73
9. Domenger, J.: Conception et implémentation du noyeau graphique d'un environnement 2D1/2 d'édition d'images discrètes. Thèse de doctorat, Université Bordeaux I (1992)
10. Fiorio, C.: A topologically consistent representation for image analysis: the frontiers topological graph. In: Discrete Geometry for Computer Imagery. Number 1176 in Lecture Notes in Computer Science, Lyon, France (1996) 151–162
11. Braquelaire, J.P., Brun, L.: Image segmentation with topological maps and inter-pixel representation. Journal of Visual Communication and Image Representation **9** (1998) 62–79
12. Damiand, G., Resch, P.: Topological map based algorithms for 3d image segmentation. In: Discrete Geometry for Computer Imagery. Number 2301 in Lecture Notes in Computer Science, Bordeaux, France (2002) 220–231
13. Bourdon, P., Alata, O., Damiand, G., Olivier, C., Bertrand, Y.: Geometrical and topological informations for image segmentation with monte carlo markov chain implementation. In: Vision Interface, Calgary, Canada (2002) 413–420
14. Bertrand, Y., Damiand, G., Fiorio, C.: Topological encoding of 3d segmented images. In: Discrete Geometry for Computer Imagery. Number 1953 in Lecture Notes in Computer Science, Uppsala, Sweden (2000) 311–324