Computer Vision and Image Understanding 121 (2014) 2–12

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Topological modifications and hierarchical representation of cell complexes in arbitrary dimensions

Lidija Čomić [a,*], Leila De Floriani [b], Federico Iuricich [b], Ulderico Fugacci [b]

[a] Faculty of Technical Sciences, University of Novi Sad, Novi Sad, Serbia
[b] Department of Computer Science, Bioengineering, Robotics and Systems Engineering, University of Genova, Genova, Italy

ARTICLE INFO

Article history:
Received 26 October 2012
Accepted 29 November 2013

Keywords:
Computational topology
Cell complexes
Hierarchical models
Homology generators

ABSTRACT

We propose a set of atomic modeling operators for simplifying and refining cell complexes in arbitrary dimensions. Such operators either preserve the homology of the cell complex, or they modify it in a controlled way. We show that such operators form a minimally complete basis for updating cell complexes, and we compare them with various operators previously proposed in the literature. Based on the new operators, we define a hierarchical model for cell complexes, that we call a *Hierarchical Cell Complex* (*HCC*), and we discuss its properties. An *HCC* implicitly encodes a virtually continuous set of complexes obtained from the original complex through the application of our operators. Then, we describe the implementation of a version of the *HCC* based on the subset of the proposed modeling operators which preserve homology. We apply the homology-preserving *HCC* to enhance the efficiency in extracting homology generators at different resolutions. To this aim, we propose an algorithm which computes homology generators on the coarsest representation of the original complex, and uses the hierarchical model to propagate them to complexes at any intermediate resolution, and we prove its correctness. Finally, we present experimental results showing the efficiency and effectiveness of the proposed approach.

1. Introduction

Cell complexes are used as a discretization and modeling tool in a wide range of application domains, such as solid modeling, computer graphics, computer aided design, finite element analysis, animation, scientific visualization, and geographic data processing. Cell complexes are used to discretize geometric shapes, such as static and dynamic 3D objects, or surfaces and hyper-surfaces describing the behavior of scalar or vector fields. A variety of topological operators have been designed for building and updating data structures representing two and three dimensional cell complexes, such as handle operators [1,2], Euler operators [3–10], or removal/insertion operators in *n*D [11–13]. Handle operators are based on the handlebody theory [14], stating that any *n*-manifold can be obtained from an *n*-ball by attaching handles to it. The main characteristic of Euler operators is that they maintain the Euler–Poincaré formula expressing a topological validity condition for a cell complex.

We describe here a set of modeling operators which form a minimally complete basis for simplifying and refining cell complexes in arbitrary dimensions in a topologically consistent manner. These operators have been originally proposed in [15]. We distinguish between operators that maintain homology of the complex, and operators that modify it in a controlled manner. *Homology-preserving* operators add (or remove) a pair of cells of consecutive dimension, but they do not change the Betti numbers of the complex. *Homology-modifying* operators add (or remove) an *i*-cell, and increase (or decrease) the *i*th Betti number. We compare our modeling operators with other operators on cell complexes proposed in the literature, and we show how these latter can be expressed in terms of the former.

Based on the proposed operators, we define a hierarchical model that we call a *Hierarchical Cell Complex (HCC)*. An *HCC* is generated by applying our simplification operators, and it is defined based on their inverse refinement operators and on a dependency relation among these latter that guides the extraction of topologically correct complexes at uniform or variable resolutions. Unlike the pyramidal model defined on quasi-manifolds represented as 2-maps [16] and *n*-G-maps [17], an *HCC* can represent arbitrary cell complexes, is based on both homology-preserving and homology-modifying operators, and allows extracting a large number of complexes, also adaptive, according to any user-selected approximation criterion.

In our work, we have also defined and implemented a version of the *HCC* based only on homology-preserving operators [18]. Here, we investigate the use of this model for homology computation.

* Corresponding author.
E-mail addresses: lidija.comic@gmail.com, comic@uns.ac.rs (L. Čomić).

1077-3142/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.cviu.2013.11.011

We show that the *HCC* based on the homology-preserving operators enables us to obtain the homology (with coefficients in $\mathbb{Z}_2$) of all the complexes encoded in the model by computing the homology of the complex at the coarsest resolution using standard techniques [19]. Moreover, we are able to construct homology generators of a complex at any intermediate resolution by computing generators on the coarsest complex and using the hierarchical model to propagate the computed generators to all the complexes at intermediate resolutions. Unlike approaches based on pyramids on n-maps and n-G-maps [12,20], the implementation of the *HCC* based on homology-preserving operators can be applied to general complexes, not only to quasi-manifolds and it supports the extraction of homology generators at variable resolutions.

The remainder of this paper is organized as follows. In Section 2, we review some background notions on cell complexes. In Section 3, we describe the new set of topological operators for simplifying and refining cell complexes in arbitrary dimensions. In Section 4, we show how various other update operators on cell complexes proposed in the literature, such as removal/insertion operators, Euler operators and handle operators, can be expressed in terms of our operators. In Section 5, we investigate the properties of an *HCC*, and we show how to extract from an *HCC* a large number of complexes at intermediate resolutions. In Section 6, we discuss how we retrieve homology and homology generators of complexes at intermediate resolutions, starting from the homology and the homology generators of the complex at the coarsest resolution. In Section 7, we present some experimental evaluation of the model. In Section 8, we draw some concluding remarks, and discuss possible research directions.

2. Background notions

In this section, we review some notions on cell complexes (see [19,21] for more details). A (topological) *k-cell* in the Euclidean space $\mathbb{E}^n$ is a homeomorphic image of a k-dimensional open ball. Intuitively, a *cell complex* in $\mathbb{E}^n$ is a finite set X of cells in $\mathbb{E}^n$ of dimension at most d, $0 \leqslant d \leqslant n$, such that

- the cells in X are pairwise disjoint;
- for each cell $x \in X$, the boundary of x is a disjoint union of cells in X.

If the maximum dimension of the cells in X is equal to d, then X is called a *d-complex*. The set of cells on the boundary of a cell x is called the *(combinatorial) boundary* of x. The *(combinatorial) co-boundary* (or *star*) of x consists of all the cells of X that have x on their combinatorial boundary. An h-cell x' on the boundary of a k-cell x, $0 \leqslant h \leqslant k$, is called an *h-face* of x, and x is called a *coface* of x'. Each cell x is a face of itself. If $x' \neq x$, then x' is called a *proper face* of x. The set of $(k-1)$-cells on the boundary of a k-cell x forms its *immediate boundary* ($1 \leqslant k \leqslant d$), and the set of $(k+1)$-cells in its co-boundary forms its *immediate co-boundary* ($0 \leqslant k \leqslant d-1$). A cell, that is not a proper face of any other cell in X, is a *top cell*. The *domain*, or *carrier*, of a cell d-complex X embedded in $\mathbb{E}^n$, with $0 \leqslant d \leqslant n$, is the subset of $\mathbb{E}^n$ defined by the union, as sets of points, of all the cells in X.

A *simplicial complex* Σ is a finite set of (closed) simplexes (convex hulls of affinely independent points), such that each face of a simplex in Σ is in Σ and each nonempty intersection of any two simplexes x and y in Σ is a face of both x and y. A simplicial d-complex X (where d is the maximum dimension of the simplexes in Σ) is a *pseudo-manifold* if (i) X is homogenous (each simplex is a face of some d-simplex), (ii) each $(d-1)$-simplex in X is a $(d-1)$-face of at most two d-simplexes and (iii) X is strongly connected (for any two distinct d-simplexes x and y in X there is a sequence $x = x_1, x_2, \ldots, x_l = y$, such that x_i and x_{i+1} share a $(d-1)$-simplex, $1 \leqslant i < l-1$).

The Euler–Poincaré formula [19] for a cell d-complex X with n_i i-cells states that $\sum_{i=0}^{d}(-1)^i n_i = \sum_{i=0}^{d}(-1)^i \beta_i$. Here, β_i is the ith Betti number of X, and it measures the number of independent non-bounding i-cycles in X, i.e., the number of independent i-holes. The alternating sum $n_0 - n_1 + \cdots + (-1)^d n_d$ is denoted as $\chi(X)$, and is called the *Euler–Poincaré characteristic* of X.

3. Topological atomic operators on cell complexes

In this section, we present the operators on cell complexes in arbitrary dimensions that we have first introduced in [15]. We show here that these operators form a minimally complete basis for the set of all possible operators that modify cell complexes in a topologically consistent manner.

3.1. Topological operators

There have been many proposals in the literature for manipulating two- and three-dimensional cell complexes. We propose here a minimal set of Euler operators on cell complexes in arbitrary dimensions, which subsume, as we will show, all the other Euler operators proposed in the literature. These operators can be classified as:

- homology-preserving operators: $MiC(i+1)C$ (*Make i-Cell and (i+1)-Cell*), which create an i-cell and an $(i+1)$-cell,
- homology-modifying operators: *MiCiCycle* (*Make i-Cell and i-Cycle*), which create an i-cell and an i-cycle.

There are in total d homology-preserving and $d+1$ homology-modifying operators on cell d-complexes.

Homology-preserving (*refinement*) operators $MiC(i+1)C$ change the number of cells in the complex X by increasing the number n_i of i-cells and the number n_{i+1} of $(i+1)$-cells by a unit. We have proven, by using discrete Morse theory [22], that the homology-preserving operators do not change the Euler characteristic, or the Betti numbers of the cell complex with respect to any Abelian group. The proof is reported in Appendix A. There are two types of homology-preserving operators, each of which has two distinct instances.

The operator $MiC(i+1)C$ of the first type has the following two instances:

- the first instance, that we denote as $MiC(i+1)C_{ex}(q,p,p')$ (*expand*), subdivides the existing i-cell p' into two by splitting its co-boundary, and creates the $(i+1)$-cell q bounded by the two i-cells p and p';
- the second instance, that we denote as $MiC(i+1)C_{in}(q,p,p')$ (*insert*), subdivides the existing $(i+1)$-cell p' into two by splitting its boundary, and creates the i-cell q separating the two $(i+1)$-cells p and p'.

In both cases, the new i-cell appears exactly once on the boundary of the new $(i+1)$-cell.

For a 2-complex X embedded in $\mathbb{E}^3$, the homology-preserving operators are usually called *MEV* (*Make Edge and Vertex*) and *MEF* (*Make Edge and Face*), which correspond to *M0C1C* and *M1C2C*, respectively. For a 3-complex X embedded in $\mathbb{E}^3$, there is an additional homology-preserving operator, *MFVl* (*Make Face and Volume (3-Cell)*) which creates a new 2-cell and a new 3-cell. It is the same as *M2C3C*.

The operator $MiC(i+1)C$ of the second type either creates an i-cell and an $(i+1)$-cell bounded only by the i-cell, or dually, it

creates an $(i+1)$-cell and an i-cell bounding only the $(i+1)$-cell. The new i-cell appears exactly once on the boundary of the new $(i+1)$-cell. We will denote the first instance of the operator as $MiC(i+1)C_{ex}(q,p)$, and the second one as $MiC(i+1)C_{in}(q,p)$.

The inverse $KiC(i+1)C$ (*Kill i-Cell and (i+1)-Cell*) (*simplification*) operators delete an i-cell and an $(i+1)$-cell from the complex X. Again, we have operators of two different types. The operator $KiC(i+1)C(q,p,p')$ of the first type is *feasible* under the following conditions:

- the $(i+1)$-cell q to be deleted is bounded by exactly two i-cells (the i-cell p to be deleted and the i-cell p' which will remain), and the i-cell p appears exactly once on the boundary of the $(i+1)$-cell q;
- the i-cell q to be deleted bounds exactly two $(i+1)$-cells (the $(i+1)$-cell p to be deleted and the $(i+1)$-cell p' which will remain) and the i-cell q appears exactly once on the boundary of the $(i+1)$-cell p.

In the first case, denoted as $KiC(i+1)C_{co}(q,p,p')$ (*contract*), the effect of the operator is as follows:

- the i-cell p is replaced with the i-cell p' on the boundary of each $(i+1)$-cell r in the co-boundary of the i-cell p.
- if the i-cell p appears k times on the boundary of the $(i+1)$-cell r, then k copies of the $(i+1)$-cell q are merged into each $(i+1)$-cell r.

The second instance, denoted as $KiC(i+1)C_{re}(q,p,p')$ (*remove*), is completely dual.

The operator $KiC(i+1)C(q,p)$ of the second type is *feasible* under the following conditions:

- the $(i+1)$-cell q (to be deleted) is bounded only by the i-cell p, which will be deleted as well ($KiC(i+1)C_{co}(q,p)$);
- the i-cell q (to be deleted) bounds only the $(i+1)$-cell p which will be deleted as well ($KiC(i+1)C_{re}(q,p)$).

In both cases, the deleted i-cell appears exactly once on the boundary of the deleted $(i+1)$-cell. The effect of the operator is to delete both cells from the complex.

Fig. 1 illustrates a sequence consisting of operators $K1C2C_{re}(q,p,p'), K1C2C_{re}(q_1,p_1,p_1')$ and $K0C1C_{co}(q_2,p_2,p_2')$ in 2D. $K1C2C_{re}(q,p,p')$ removes 1-cell q and 2-cell p similarly to $K1C2C_{re}(q_1,p_1,p_1')$, which removes 1-cell q_1 and 2-cell p_1, while $K0C1C_{co}(q_2,p_2,p_2')$ collapses 1-cell q_2 and 0-cell p_2 into 0-cell p_2'. Fig. 2 illustrates a sequence consisting of operators $K2C3C_{re}(q,p,p')$ and $K1C2C_{re}(q_1,p_1,p_1')$ in 3D. $K2C3C_{re}(q,p,p')$ removes 2-cell q and 3-cell p, while $K1C2C_{re}(q_1,p_1,p_1')$ removes 1-cell q_1 and 2-cell p_1.

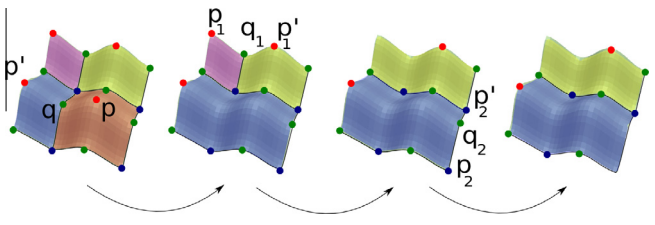

Fig. 1. Effect of the sequence consisting of (from left to right) $K1C2C_{re}(q,p,p')$, $K1C2C_{re}(q_1,p_1,p_1')$ and $K0C1C_{co}(q_2,p_2,p_2')$ on a portion of a two-dimensional cell complex. Blue dots (e.g., p_2 and p_2') correspond to 0-cells, green dots (e.g., q, q_1 and q_2) to 1-cells and red dots (e.g., p, p', p_1 and p_1') to 2-cells. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

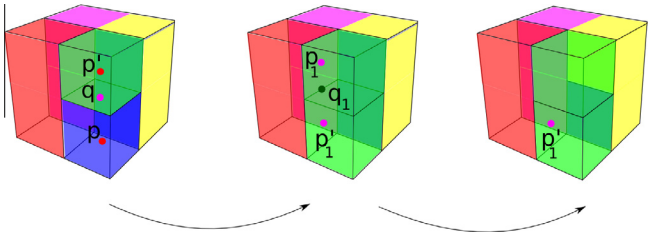

Fig. 2. Effect of the sequence consisting of (from left to right) $K2C3C_{re}(q,p,p')$ and $K1C2Cre(q_1,p_1,p_1')$ on a portion of a three-dimensional cell complex. Green dots (q_1) correspond to 1-cells, purple dots (q, p_1 and p_1') to 2-cells and red dots (p and p') to 3-cells. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Homology-modifying operators change the number of cells in a complex X plus its Betti numbers and Euler characteristic. Homology-modifying operator *MiCiCycle* increases the number n_i of i-cells and the number β_i of independent non-bounding i-cycles by a unit. It is feasible on a complex X if all the cells on the boundary of the cell to be created are in X. The inverse *KiCiCycle* (*Kill i-Cell and i-Cycle*) operator deletes an i-cell and destroys an i-cycle, thus decreasing the numbers n_i and β_i by a unit. It is feasible on a cell complex X if the co-boundary of the cell to be deleted is empty.

For a 2-complex X embedded in $\mathbb{E}^3$, the homology-modifying operators (illustrated in Fig. 3) are also called *MV0Cycle* (*Make Vertex and 0-Cycle*), *ME1Cycle* (*Make Edge and 1-Cycle*) and *MF2Cycle* (*Make Face and 2-Cycle*). Operator *MV0Cycle* creates a new vertex and a new connected component, it increases by a unit the number of vertices (0-cells) and the zeroth Betti number β_0. It is used as an initialization operator to create a new complex X. Operator *ME1Cycle* creates a new edge and a new 1-cycle, thus increasing both the number of edges (1-cells) and the first Betti number β_1 by a unit. Operator *MF2Cycle* creates a new 2-cell (face) and a new 2-cycle, thus increasing the number of 2-cells and the second Betti number β_2 by a unit. When considering a 3-complex X embedded in $\mathbb{E}^3$, the homology-modifying operators will be the same as for 2-complexes, since in this case the third Betti number β_3 is null.

3.2. Minimality and completeness

The operators described in Section 3.1 form a minimally complete set of basis operators for creating and updating cell d-complexes. To prove this fact, we interpret such operators as ordered $(2d+2)$-tuples $(x_0,x_1,\ldots,x_d,c_0,c_1,\ldots,c_d)$ in an integer grid, belonging to the hyperplane $\Pi: \sum_{i=0}^{d}(-1)^i x_i = \sum_{i=0}^{d}(-1)^i c_i$ defined by the Euler–Poincaré formula. The first $d+1$ coordinates denote the number of i-cells created or deleted by the operator, depending on the sign of the coordinate, while the last $d+1$ coordinates denote the change in the Betti numbers of the complex induced by the operator. Operator $MiC(i+1)C,\ 0 \leqslant i \leqslant d-1$, has coordinates $x_i = x_{i+1} = 1,\ x_j = 0,\ j \in \{0,1,\ldots,d\} \setminus \{i, i+1\},\ c_j = 0,\ j \in \{0,1,\ldots,d\}$. Operator *MiCiCycle*, $0 \leqslant i \leqslant d$, has coordinates $x_i = c_i = 1,\ x_j = c_j = 0,\ j \in \{0,1,\ldots,d\} \setminus \{i\}$. We will show that:

(i) the $2d+1$ $(2d+2)$-tuples corresponding to our operators are linearly independent, and
(ii) any $(2d+2)$-tuple in the hyperplane Π can be expressed as a linear combination of $2d+1$ $(2d+2)$-tuples corresponding to our operators,

which will imply the claim.

A linear combination $\sum_{i=0}^{d-1}\alpha_i MiC(i+1)C + \sum_{i=0}^{d}\beta_i MiCiCycle$ vanishes if and only if $(\alpha_0,\alpha_0,0,\ldots,0)+(0,\alpha_1,\alpha_1,\ \ldots,0)+\cdots+$

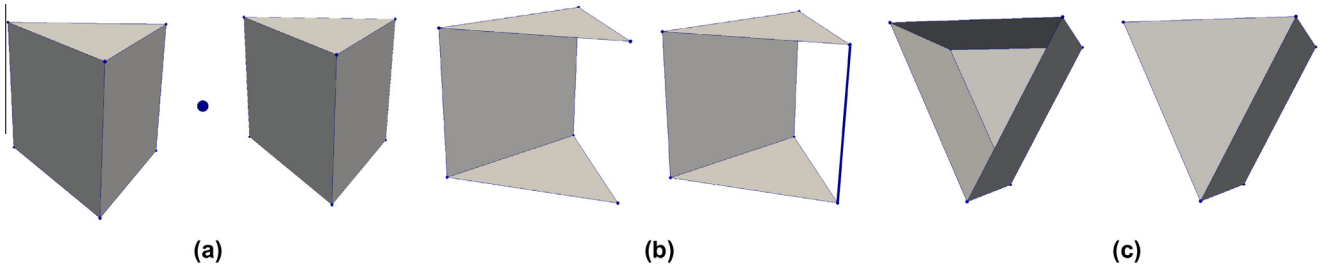

Fig. 3. Homology-modifying operators on a 2-complex in $\mathbb{E}^3$: *MV0Cycle* (*Make Vertex and 0-Cycle*) (a); *ME1Cycle* (*Make Edge and 1-Cycle*) (b); *MF2Cycle* (*Make Face and 2-Cycle*) (c).

$(0,\dots 0,\alpha_{d-1},\alpha_{d-1},0,\dots,0)+(\beta_0,0,\dots,0,\beta_0,0,\dots,0)+(0,\beta_1,0,\dots,0,\beta_1,0,\dots,0)+\cdots+(0,\dots,0,\beta_d,0,\dots,0,\beta_d)=0$, which is equivalent to $(\alpha_0+\beta_0,\alpha_0+\alpha_1+\beta_1,\alpha_1+\alpha_2+\beta_2,\ \dots,\alpha_{d-2}+\alpha_{d-1}+\beta_{d-1},\alpha_{d-1}+\beta_d,\beta_0,\beta_1,\dots,\beta_d)=0$. It follows that $\alpha_i=0,\ 0\leqslant i\leqslant d-1,\ \beta_i=0,\ 0\leqslant i\leqslant d$, implying that the tuples corresponding to our operators are linearly independent.

A tuple $(a_0,a_1,\dots,a_d,b_0,b_1,\dots,b_d)$ in the hyperplane Π (i. e., such that $\sum_{i=0}^{d}(-1)^i a_i=\sum_{i=0}^{d}(-1)^i b_i$) can be expressed through the $2d+1$ independent $(2d+2)$-tuples corresponding to our operators as $\sum_{i=0}^{d-1}\alpha_i MiC(i+1)C+\sum_{i=0}^{d}\beta_i MiCiCycle$ if $(\alpha_0+\beta_0,\alpha_0+\alpha_1+\beta_1,\alpha_1+\alpha_2+\beta_2,\dots,\alpha_{d-2}+\alpha_{d-1}+\beta_{d-1},\alpha_{d-1}+\beta_d,\beta_0,\beta_1,\dots,\beta_d)=(a_0,a_1,\dots,a_d,b_0,b_1,\dots,b_d)$. It follows that $\beta_i=b_i,\ 0\leqslant i\leqslant d$, and $\alpha_0=a_0-b_0,\ \alpha_1=a_1-b_1-\alpha_0=(a_1-a_0)-(b_1-b_0),\quad \alpha_2=a_2-b_2-\alpha_1=(a_2-a_1+a_0)-(b_2-b_1+b_0),\dots,\alpha_{d-1}=(a_{d-1}-a_{d-2}+\cdots+(-1)^d a_0)-(b_{d-1}-b_{d-2}+\cdots+(-1)^d b_0)=a_d-b_d$. In short, $\alpha_i=\sum_{j=0}^{i}(-1)^{i-j}a_j-\sum_{j=0}^{i}(-1)^{i-j}b_j, 0\leqslant i\leqslant d-1$ ($\alpha_{d-1}=a_d-b_d$) and $\beta_i=b_i,\ 0\leqslant i\leqslant d$. Thus, each operator satisfying Euler–Poincaré formula on a cell complex X can be expressed as a linear combination of our operators.

In the space (hyperplane) of dimension $2d+1$, a generating set consisting of $2d+1$ independent tuples forms a basis for the hyperplane.

4. Comparison with other update operators on cell complexes

We compare the operators proposed here with other update operators on cell complexes proposed in the literature, in particular with removal and contraction operators, Euler operators and handle operators. For a more detailed discussion, see [15].

4.1. Removal and contraction operators

Removal and contraction operators have been introduced in digital geometry as simplification operators on n-G-maps [11]. An i-cell $q,\ 0\leqslant i\leqslant n-1$, can be removed in two cases: if it bounds exactly two different $(i+1)$-cells p and p' and it appears exactly once on the boundary of both p and p'; or if it bounds exactly one $(i+1)$-cell p and it appears exactly twice on the boundary of p. The contraction operator is dual.

The first instance of the removal operator is a special case of $KiC(i+1)C_{re}(q,p,p')$, as it requires that the i-cell q appears exactly once not only on the boundary of the $(i+1)$-cell p but also on the boundary of the $(i+1)$-cell p'. The effect of the first instance of the removal operator is the same as the effect of $KiC(i+1)C_{re}$. The second instance of the removal operator may, but is not guaranteed to, preserve the topological characteristics of the complex (it may produce cells that are not topological cells, or it may disconnect the complex). Thus, it cannot be classified either as a homology-preserving, or as a homology-modifying operator.

In [12], homology generators of a cell complex are computed using two homology-preserving simplification operators: the removal of an i-cell incident in exactly two $(i+1)$-cells (which is the same as $KiC(i+1)C_{re}(q,p,p')$ and as the first instance of the removal operator in [11]) and the removal of a dangling cell (which is the same as $KiC(i+1)C_{re}(q,p)$). The inverse (refinement) insertion and expansion operators have been introduced in [13]. They are the same as $MiC(i+1)C_{in}(q,p,p')$ and $MiC(i+1)C_{ex}(q,p,p')$, respectively.

4.2. Euler operators

Virtually all the proposed sets of basis Euler operators on cell 2- and 3-complexes contain *MEV* (Make Edge and Vertex) and *MEF* (Make Edge and Face) operators, which are the same as *M0C1C* (Make 0-cell and 1-cell) and *M1C2C* (Make 1-cell and 2-cell), respectively (see Section 3.1).

Several homology-modifying operators have been proposed for 2-complexes that define the boundary of a solid in $\mathbb{E}^3$, called *boundary models*. In these models, there is only one implicitly represented volumetric cell, which is not necessarily homeomorphic to a 3D ball. The *glue* operator in [3] merges two 2-cells and deletes both of them. It corresponds to the connected sum operator on manifold surfaces. If the two glued 2-cells belong to two different connected components of the complex, one of the components is deleted (and β_0 is decreased by a unit). If the two glued faces belong to the same connected components, a handle or through-hole is created (and β_1 is increased by two units). In [4–6], the homology-modifying operator is called *MRKF* (*Make Ring, Kill Face*). It is similar to the *glue* operator in [3], but it imposes less restrictive conditions on the 2-cells to be glued, and it deletes only one of the 2-cells. The 2-cells are not supposed to be topological (homeomorphic to a 2D ball).

Homology-modifying operators defined for general (non-manifold) 2-complexes in $\mathbb{E}^3$ [7] are called *MECh* (*Make Edge and Complex Hole*), *MFKCh* (*Make Face, Kill Complex Hole*) and *MFCc* (*Make Face and Complex Cavity*). They are the same as operators *M1C1Cycle*, *M2CK1Cycle* (*Make 2-Cell Kill 1-Cycle*, which can be expressed as *K1C1Cycle*, *M1C2C*) and *M2C2Cycle*, respectively. For 3-complexes in $\mathbb{E}^3$ [8,9], an additional homology-modifying operator is defined, called *MVlKCc* (*Make Volume, Kill Complex Cavity*). It is the same as *M3CK2Cycle* (*Make 3-Cell, Kill 2-Cycle*) operator, and can be expressed as *K2C2Cycle*, *M2C3C*.

In [10], the operators defined in [8] have been extended to complexes called *stratifications*, in which the cells, called *strata*, are defined by analytic equalities and inequalities. The cells are not necessarily homeomorphic to a ball, and they may have incomplete boundaries. Among the operators on stratifications proposed in [10], operators on topological cells (that are homeomorphic to a ball) with complete boundaries can be classified as homology-preserving (called *cell subdividers*) and homology-modifying (called *global hole shapers*). Both types of operators are instances of the operators defined in Section 3.1. A *cell subdivider* subdivides an i-cell by inserting into it an $(i-1)$-cell. This operator is the same

as the $M(i-1)CiC$ operator. A *global hole shaper* either attaches or detaches a cell, thus creating a hole. There are two instances of this operator: the attached topological i-cell creates an i-hole or the detached topological i-cell creates an $(i-1)$-hole. The first instance of this operator corresponds to *MiCiCycle*. The second instance corresponds to $KiCM(i-1)Cycle$ (*Kill i-Cell, Make (i-1)-Cycle*), and can be expressed as $M(i-1)C(i-1)Cycle, K(i-1)CiC$. The inverse homology-modifying operators attach or detach a cell, thus deleting a hole. They correspond to *KiCiCycle* and $MiCK(i-1)Cycle$ (inverse to $KiCM(i-1)Cycle$), respectively.

4.3. Handle operators

(Homology-modifying) *handle operators* on a manifold cell 2-complex X triangulating a surface S have been introduced in [1]. They are based on the *handlebody theory* for surfaces [14], stating that any surface S can be obtained from a 2-ball by iteratively attaching handles (0-, 1- and 2-handles).

Attachment of a 0-handle creates a new surface with one face, three edges and three vertices. It can be expressed as one *M0C0Cycle* operator, two *M0C1C* operators and one *M1C2C* operator, which together create a triangle. The operator that corresponds to the attachment of a 1-handle identifies two boundary edges of X (incident in exactly one face) with no vertices in common. The operator that corresponds to the attachment of a 2-handle identifies two boundary edges of X with two vertices in common. They can be expressed through our operators in a similar manner.

Handle operators have been extended to 3D in [2]. The operator that creates a new 3-ball (initialization operator) corresponds to the attachment of a 0-handle. Other operators identify two boundary faces (incident in exactly one 3-cell) of a cell 3-complex X triangulating a solid S. The attachment of a 1-handle (2-handle, or 3-handle) can be applied if the two identified boundary faces have no edges (some edges, or all edges) in common. The handle operators in 3D generalize the glue operator in [3]. They can be expressed in terms of the operators defined in Section 3.1 in a similar manner.

5. Hierarchical cell complexes

In this section, we introduce a hierarchy of cell complexes, that we call a *Hierarchical Cell Complex (HCC)*, and we discuss its major properties. We define an *HCC* in terms of the refinement operators introduced in Section 3.

A *Hierarchical Cell Complex (HCC)* is generated from a d-complex X at full resolution by iteratively applying simplification operators $KiC(i+1)C$ and *KiCiCycle*. By applying first the homology-preserving operators, we obtain a complex X' having the same homology as the original complex X but with fewer cells and such that no homology-preserving operator is feasible on X'. By applying next the homology-modifying operators to iteratively remove the cells of X', the homology is affected at each step and the process is repeated until a complex is obtained that has at least one i-cell, $0 \leqslant i \leqslant d$. At each step, when we apply a homology-modifying operator, we remove a top cell from the complex. After each application of a homology-modifying operator, we perform feasible homology-preserving ones.

We call the application of a simplification operator a *simplification modification*. The complex obtained as the result of the simplification sequence is the coarsest representation of the original cell complex X. We denote such coarsest complex as X_B. It represents the first component of an *HCC*.

The second component of an *HCC* is the set $\mathcal{M}$ of the *refinement modifications* which are inverse with respect to the simplification modifications that have produced X_B from X. Each refinement introduces new cells (two cells if it is homology-preserving, one if it is homology-modifying).

The third component of an *HCC* is the dependency relation between the modifications in the set $\mathcal{M}$ of all refinement modifications. We consider, for simplicity, the creation of the coarse complex X_B as a dummy refinement modification that we denote as μ_0 (i.e., μ_0 generates X_B). We define the dependency relation between the refinement modifications in $\mathcal{M}$ as follows:

- a homology-preserving refinement $\mu = MiC(i+1)C$, which creates cells p and q and is defined by the cells that will appear on the immediate boundary or co-boundary of either p or q, *directly depends* on a refinement μ^*, if μ^* creates a cell that will be on the immediate boundary or co-boundary of p or q,
- a homology-modifying refinement $\mu = MiCiCycle$, which creates i-cell p and is defined by the $(i-1)$-cells that will be on the immediate boundary of p, *directly depends* on a refinement μ^*, if μ^* creates a cell that will be on the immediate boundary of p.

An *HCC* is thus a triple $(X_B, \mathcal{M}, \mathcal{R})$, where $\mathcal{R}$ denotes the direct dependency relation defined above. The dependency relation between refinement modifications is the transitive closure of the direct dependency relation. It is a partial order relation, since a cell is never introduced twice by the modifications in $\mathcal{M}$. An *HCC* can be naturally encoded as a Directed Acyclic Graph (*DAG*), in which the nodes encode the modifications in $\mathcal{M}$, the root encodes the creation of the base complex X_B (modification μ_0), and the arcs describe the direct dependency relation $\mathcal{R}$.

From an *HCC*, a large number of complexes at intermediate resolution can be obtained by applying sequences of refinement modifications in $\mathcal{M}$ to the base complex X_B. A sequence $U = (\mu_0, \mu_1, \ldots, \mu_k)$ is said to be *feasible* if each refinement μ_i in U is feasible on the complex obtained from the base complex X_B by applying all the refinements preceding μ_i in U. For a feasible sequence $U = (\mu_0, \mu_1, \mu_2, \ldots, \mu_m)$ of refinement modifications in $\mathcal{M}$, the complex obtained from the base complex X_B by applying U is called the *front complex* associated with U, and we denote it as X_U. A front complex is a complex at an intermediate resolution.

The refinement modification μ, which creates the cells p and q (if μ is homology-preserving), or the cell p (if μ is homology-modifying) is feasible on a front complex X_U (at some intermediate resolution) if and only if all the cells on the immediate boundary and co-boundary of the cells p and q (if μ is homology-preserving) or all the cells on the immediate boundary of the cell p (if μ is homology-modifying) are in the complex X_U, i.e., if the sequence U that creates X_U from X_B contains all refinement modifications on which μ depends.

A large number of adaptive morphological representations can thus be extracted from an *HCC* defined by the triple $(X_B, \mathcal{M}, \mathcal{R})$ by considering the closed sets of refinement modifications in $\mathcal{M}$ plus μ_0 under the dependency relation $\mathcal{R}$. Recall that the dependency relation $\mathcal{R}$ is a partial order relation, and thus it defines a closure operator on the set $\mathcal{M}$ of refinement modifications. We denote a *closed set* of such refinement modifications as $\mathcal{U}$. Set $\mathcal{U}$ implicitly defines complex X_U representing an approximation of the original complex.

The set $\mathcal{U} = \{\mu_0, \mu_1, \mu_2, \ldots, \mu_m\}$ of refinement modifications in $\mathcal{M}$ is *closed* with respect to the dependency relation $\mathcal{R}$ if for each $i, 1 \leqslant i \leqslant m$, each refinement modification on which the refinement μ_i depends is in $\mathcal{U}$. If the sequence $U = (\mu_0, \mu_1, \mu_2, \ldots, \mu_m)$ of refinement modifications in $\mathcal{M}$ is feasible, then the set $\mathcal{U} = \{\mu_0, \mu_1, \mu_2, \ldots, \mu_m\}$ is *closed* with respect to the dependency relation $\mathcal{R}$.

Two feasible refinement modifications μ_1 and μ_2 on a complex are *interchangeable* if the sequence (μ_1, μ_2) of refinements (consisting of μ_1 followed by μ_2) produces the same complex as the

sequence (μ_2, μ_1) (consisting of μ_2 followed by μ_1). Any two independent refinement modifications μ_1 and μ_2 are interchangeable. Thus, a closed subset $\mathcal{U}$ of refinement modifications can be applied to the base complex X_B in any total order U that extends the partial order, producing a complex X_U at an intermediate resolution. An *HCC* encodes a collection of all complexes at intermediate level of detail which can be obtained from the base complex X_B by applying a closed set of modifications on X_B.

From an *HCC* it is thus possible to dynamically extract representations of the original cell complex X at uniform and variable resolutions. The basic query for extracting a single-resolution representation from a multi-resolution model is known as selective refinement. A *selective refinement query* consists of extracting from an *HCC* a complex with the *minimum* number of cells, satisfying some application-dependent criterion. This criterion can be formalized by defining a Boolean function τ over all nodes of an *HCC*, such that the value of τ is *true* on nodes which satisfy the criterion, and *false* otherwise. The same value of τ is associated with the cells created by the modification encoded in the node of the *HCC* (p and q for homology-preserving modification, p for homology-modifying modification). The selective refinement query consists of extracting from the *HCC* an intermediate complex of minimum size among the complexes encoded in the *HCC* that satisfies τ. Equivalently, it consists of extracting a minimal closed set $\mathcal{U}$ of modifications in $\mathcal{M}$, such that the corresponding complex satisfies τ. Such closed set of modifications uniquely determines a front complex, which is obtained from the base complex X_B, by applying to it all modifications from $\mathcal{U}$ in any order that is consistent with the partial order defined by the dependency relation. Criterion τ can be defined based on various conditions posed on the cells in the extracted complex, like the size of the cell (which may be expressed as the maximum distance between its vertices or the diameter of its bounding box) or the portion of the complex in which full resolution is required (while in the rest of the complex, the resolution may be arbitrarily low).

In Fig. 4 we show the *HCC* built from the sequence of simplifications illustrated in Fig. 1. We can notice that each node, with the exception of the root, represents a refinement dual to a simplification applied in Fig. 1. Each closed subset of refinement modifications produces a different cell complex at intermediate resolution.

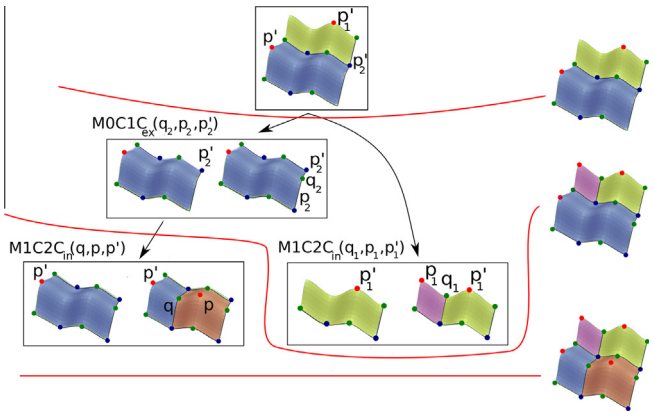

Fig. 4. An example of an *HCC* built from the simplification process illustrated in Fig. 1. The top level of the *HCC* is the root node encoding the complex at the coarsest resolution. At the bottom level are two $M1C2C_{in}$ operators. The $M1C2C_{in}(q,p,p')$ depends on the $M0C1C_{ex}(q_2,p_2,p'_2)$ and the $M1C2C_{in}(q_1,p_1,p'_1)$ depends only on the root. Blue dots (e.g., p_2 and p'_2) correspond to 0-cells, green dots (e.g., q, q_1 and q_2) to 1-cells and red dots (e.g., p, p' and p'_1) to 2-cells. On the right, three different complexes are shown, obtained by performing different closed sets of refinements on the *HCC* as indicated by the red lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6. Computing homology and homology generators

In this section, we present an approach for computing homology and homology generators at various resolutions using an *HCC* built based only on the homology-preserving operators, that we call a *homology-preserving HCC*.

We are interested in computing the homology groups $H_k(X;\mathbb{Z}_2)$ of a cell complex X with the coefficients in $\mathbb{Z}_2$. As described in [21], this corresponds to computing the Betti numbers of X with coefficients in $\mathbb{Z}_2$. Moreover, for each $k = 0,\ldots,d$, we are interested in computing the homology generators of degree k, that we call H_k *generators*. The H_k generators are the generators of the $\mathbb{Z}_2$-vector space $H_k(X;\mathbb{Z}_2)$, and they represent the independent non-bounding k-cycles in X. Each H_k generator of a cell complex X is a linear combination of k-cells in X with coefficients in $\mathbb{Z}_2$. In Fig. 5(a), two H_1 generators are shown as linear combination of 1-cells. The first generator is composed of the set of blue (bold) edges and the other one of the set of red (dotted) edges.

In a homology-preserving *HCC*, any front complex X_U is obtained from the base complex X_B by applying a sequence of homology-preserving refinement modifications $MiC(i+1)C$. In a homology-preserving *HCC* thus, the homology of the base complex is the same as the homology of any other complex implicitly encoded in the *HCC*. We use the Smith Normal Form (*SNF*) reduction algorithm [19] to compute homology and homology generators with coefficients in $\mathbb{Z}_2$ on the base complex X_B. Then, at each application of the refinement, we modify the homology generators in the currently extracted front complex X_U according to algorithm *ExpandGenerators* described below. We have shown (see the proof in Appendix B) that, when applying $MiC(i+1)C$, only the H_{i+1} generators are affected.

Let us consider refinement modification $MiC(i+1)C_{ex}(q,p)$, which creates an i-cell p and an $(i+1)$-cell q (the case of a refinement $MiC(i+1)C_{in}$ is entirely dual). Operator $MiC(i+1)C_{ex}(q,p)$ is applied on a complex Y producing a refined complex Y'. Algorithm *ExpandGenerators* checks if the introduced $(i+1)$-cell q in Y' breaks an $(i+1)$-cycle corresponding to an H_{i+1} generator in Y. This is done by considering the number of $(i+1)$-cells in the co-boundary of i-cell p that are involved in H_{i+1} generators. This idea is illustrated in Fig. 5(b) and (c), where we show two different applications of operator $M0C1C_{ex}$ to the same 2-complex (torus), depicted in Fig. 5(a). The application of operator $M0C1C_{ex}(q_1,p_1,p')$, illustrated in Fig. 5(b), modifies one of the two H_1 generators in the torus. We can notice that the new 0-cell p_1 has exactly one incident 1-cell belonging to the blue (bold) 1-chain. Thus the 1-cycle has been broken by the refinement and 1-cell q_1 is added to the 1-chain to reconstruct the cycle. On the contrary, the application of operator $M0C1C_{ex}(q_2,p_2,p')$, illustrated in Fig. 5(c), does not affect the generators. Note that 0-cell p_2 has no incident 1-cell belonging to some H_1 generator.

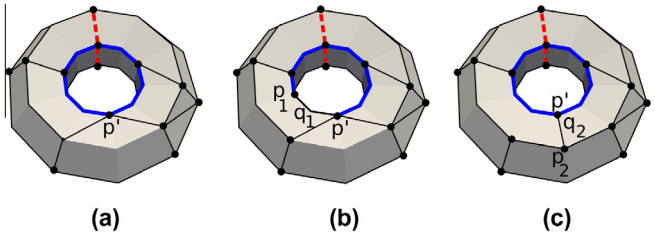

Fig. 5. (a) A cell complex representing a torus. Black dots represent 0-cells. Red (dotted) and blue (bold) edges correspond to the two H_1 generators. (b) Application of operator $M0C1C_{ex}(q_1,p_1,p')$, which affects one of the homology generators. (c) Application of operator $M0C1C_{ex}(q_2,p_2,p')$, which does not affect the homology generators. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Algorithm 1. *ExpandGenerators*(p,q,G)

Require: p is an i-cell
Require: q is an $(i+1)$-cell
Require: G is the set of H_{i+1} generators
// C is a map from a generator g to an integer m
C = empty map
//Extract the $(i+1)$-cells on the co-boundary of p
for all cofaces r of p **do**
  //G_r is the set of generators containing r
  G_r = getGeneratorsOn(r,G)
  // Consider the number of incidences between p and r
  for all generators g in G_r **do**
    $C[g]$=getIncidence(p,r)+$C[g]$
//Expand the generators on q if necessary
for all pairs (g,m) in C **do**
    if m is odd **then**
    addGenerator(g,q,G)
return G

In the description of Algorithm *ExpandGenerators*(p,q,G), p and q denote, respectively, the i-cell and the $(i+1)$-cell introduced by the refinement operator, and G represents the set of H_{i+1} generators of X. The algorithm makes use of a map C from a generator g to an integer m, that, for each generator g, stores the number of $(i+1)$-cells in the co-boundary of i-cell p which also belong to g.

Algorithm *ExpandGenerators*(p,q,G) uses the following three functions:

- *getGeneratorsOn*(r,G), which returns the set of generators G_r containing cell r in their chain,
- *getIncidence*(p,r), which returns the number of times i-cell p appears on the boundary of $(i+1)$-cell r,
- *addToGenerator*(g,q,G), which updates the generators in G by adding $(i+1)$-cell q to the $(i+1)$-chain corresponding to g.

Algorithm *ExpandGenerators*(p,q,G) considers only the $(i+1)$-cells in the co-boundary of p that are part of one or more H_{i+1} generators. For each such $(i+1)$-cell r, G_r is the set of generators that contain r (*getGeneratorsOn*(r,G)). For each generator $g \in G_r$, map C is updated by adding the number of times the i-cell p appears on the boundary of r (*getIncidence*(p,r)). Once all the $(i+1)$-cells in the co-boundary of p have been examined, cell q is added to generator g only if the number m of incidences for g is odd (*addGenerator*(g,q,G)).

7. Experimental evaluation

We have implemented the homology-preserving *HCC* by using a DAG for encoding the direct dependency relation and the Incidence Graph (*IG*) [23] for encoding the base complex X_B. We refer to [18] for details regarding the homology-preserving *HCC* encoding structure. We have performed experiments on the 2D and 3D complexes described in Table 1 by using a desktop computer with a 3.2 Ghz processor and 16 GB of memory. All complexes are simplicial complexes, that become cell complexes after undergoing some simplification.

Table 1
Four 2D shapes and two volumetric datasets used in our experiments. The columns from left to right indicate: the name of the dataset (*Dataset*), the number of the top cells in the datasets (*Cells*), the storage cost of the original cell complex (*Complex cost*), the storage cost of the *HCC* (*HCC cost*), the Betti numbers (*Homology*).

Dataset	Cells	Complex cost (MB)	*HCC* cost (MB)	Homology
Genus3	40 K	4.8	3.3	(1,6,1)
Fertility	1.4 M	176	122	(1,8,1)
Hand	2.1 M	256	177	(1,2,0)
Buddha	3.2 M	398	273	(1,208,1)
Skull	748 K	118	84	(1,2,1,0)
Fert-Solid	6.2 M	980	720	(1,4,0,0)

Table 2
Experimental results obtained by refining four 2D shapes and two volumetric datasets and by computing homology generators on them through the Smith Normal Form (*SNF*) reduction. The columns from left to right indicate: the name of the dataset (*Dataset*), time required to compute the homology generators on the base complex (*SNF Time*), the time needed to extract the complex at full resolution and to expand all the generators (*Tot Ref Time*), the number of refinements and the time needed to extract the complex and the geometry of the generators at uniform level of detail (*Uniform Ref.* and *Uniform Time*) and the number of refinements and the time needed to extract the complex and the generators concentrating the resolution only in the neighborhood of the generators (*Generators Ref.*) and (*Generators Time*). The time is expressed in seconds and the storage cost in megabytes (MB).

Dataset	*SNF* Time	Tot. Ref. Time	Uniform		Generators	
			Ref.	Time	Ref.	Time
Genus3	9.2×10^{-5} s	0.15 s	4 K	0.03 s	5 K	0.03 s
			10 K	0.07 s		
			16 K	0.12 s		
Fertility	8.3×10^{-5} s	9.31 s	144 K	1.8 s	68 K	1.48 s
			362 K	4.6 s		
			579 K	7.52 s		
Hand	9.8×10^{-5} s	14.9 s	200 K	2.6 s	19 K	1.6 s
			500 K	6.8 s		
			800 K	11.2 s		
Buddha	0.02 s	23.7 s	320 K	0.5 s	162 K	3.6 s
			800 K	4.3 s		
			1.2 M	19.2 s		
Skull	0.007 s	6.4 s	75 K	1.0 s	191 K	2.6 s
			187 K	2.9 s		
			299 K	5.0 s		
Fert-Solid	8.8 s	74.5 s	1.2 M	7.5 s	267 K	10.9 s
			3.1 M	29.1 s		
			4.9 M	69.3 s		

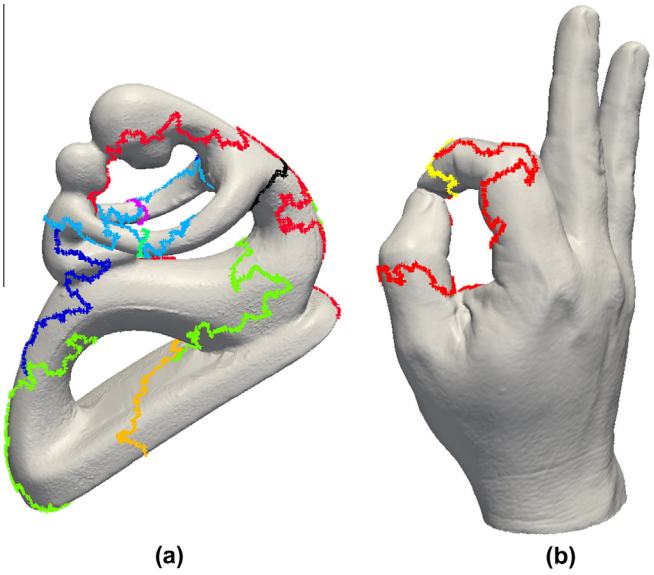

Fig. 6. The H_1 generators computed on the *Fertility* dataset (a) and on the *Hand* dataset (b) by fully refining the cell complex.

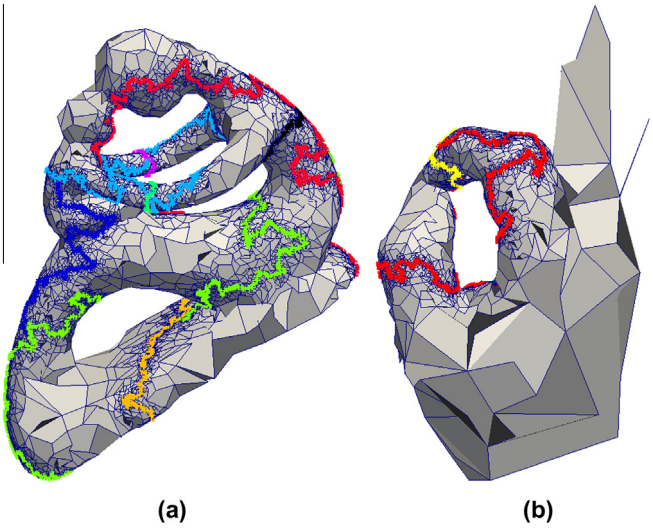

Fig. 7. The H_1 generators computed on the *Fertility* dataset and on the *Hand* dataset. In (a) and (b) the generators obtained by refining the cell complex only in a neighborhood of the generators.

The storage cost of the *HCC* encoding structure is about 25% less than the storage cost of the incidence graph representing the complex at full resolution (the original complex). These latter complexes have between 40 K and 3.2 M top cells in 2D case, and between 700 K and 6 M top cells in the 3D case, as shown in Table 1 (column *Cells*). The storage cost of the original cell complex is between 4.8 MB and 398 MB for 2D complexes, and between 118 MB and 980 MB for 3D complexes (column *Complex cost*). The storage cost of the corresponding *HCC* is between 3.3 MB and 273 MB, and between 84 MB and 720 MB (column *HCC cost*), respectively.

In the first set of experiments we have evaluated the time required to compute the homology and its generators of the original complex (the one at full resolution) by using the *HCC*. To this aim, we first compute the homology generators on the base complex, encoded in the root of the *HCC*. This computation requires between 8.3×10^{-5} and 8.8 s depending on the dataset (column *SNF Time* in Table 2). Then, we perform all the refinements in the *HCC*, by applying when necessary the refinement of the generators as described in Section 6. This produces the representation of the complex at full resolution together with the homology generators. The total cost of this computation is the sum of the time required to compute the homology of the base complex (column *SNF Time*) and the time needed to fully refine the complex and its generators (column *Tot. Ref. Time*). This takes from a minimum of 0.15 to a maximum of 83.3 s. Applying the same *SNF* reduction directly on the original complex, requires about 2.6 h on a relatively small complex (the dataset *Genus3*), while it results in very high computation times for the other datasets.

In Fig. 6, we show the H_1 generators computed on two 2D shapes *Fertility* and *Hand* and, in Fig. 8(b) and (c), we show the H_1 and H_2 generators computed on the 3D *Skull* dataset.

In the second set of experiments we have focused on extracting different representations of the complex by expanding the computed generators at different resolutions. We have considered first the extraction of representations at uniform resolution: we have extracted representations obtained from the base complex by applying approximatively 20%, 50% and 80% of the total possible refinements (column *Uniform Ref.* in Table 2). Refinements are forced to be evenly distributed inside the complex in order to obtain a uniformly refined complex. We can notice (see column *Uniform Time*) that the time required depends on the number of refinements performed and is between 0.03 and 7.5 s for extraction at 20% resolution and between 0.12 and 69.3 s for extraction at 80% resolution.

Then, we have extracted representations of the complexes varying the resolution inside the domain. The objective has been to obtain a cell complex, and the corresponding homology generators, with a maximum resolution only in a neighborhood of a specific homology class. This corresponds to computing the H_i generators on the base complex and, by traversing the *HCC*, to performing only those refinements that create an i-cell belonging to some H_i generator (and the refinements on which they depend). This kind of selective refinement produces cell complexes with a low number of cells outside the area around the generators and thus leads to a further saving (15–30%) with respect to extracting generators and complexes at maximum resolution. Note that the extraction at variable resolution is a distinctive feature of the *HCC* which cannot be performed on other hierarchical models. Examples of variable resolution extractions are shown in Fig. 7 and in Fig. 8(d).

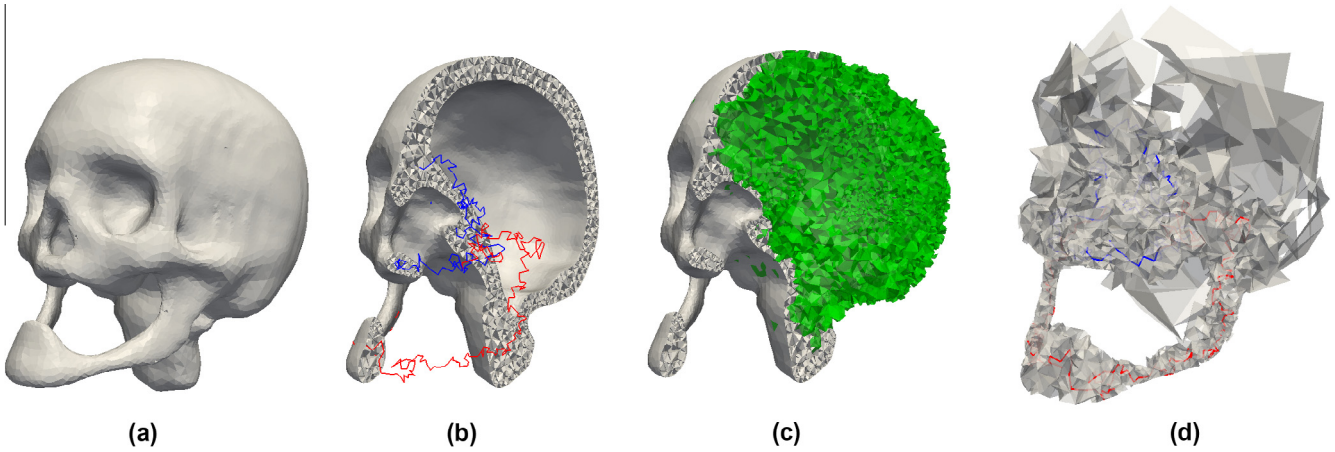

Fig. 8. The H_1 and H_2 generators computed on the *Skull* dataset. In (a) the original dataset, in (b) and (c) the H_1 and H_2 generators computed at full resolution and in (d) the H_1 generators extracted at variable resolution and visualized inside the extracted cell complex.

8. Concluding remarks

We have proposed a set of atomic modeling operators for simplifying and refining cell complexes in arbitrary dimensions, which either preserve the homology of a cell complex, or modify it in a controlled way. We have compared these operators with existing ones proposed in the literature. Based on our modeling operators, we have defined a hierarchical model for cell complexes, the Hierarchical Cell Complex (*HCC*), which implicitly encodes a virtually continuous set of complexes at different resolutions.

We have developed an implementation of the *HCC*, based on our homology-preserving operators, as the basis for computing the homology and its generators for a cell complex at various resolutions in an efficient and effective way. The advantages of our approach are that a homology-preserving *HCC* is dimension-independent, can be applied to general cell complexes and enables the extraction of homology generators at variable resolutions.

In our current and future work, we first plan to extend the previous approach to the computation of homology and homology generators with coefficients in $\mathbb{Z}$. We also plan to adapt the *HCC* framework to simplicial complexes. We will consider two simplification operators for generating an *HCC*: *simplex collapse* [24], which is an instance of simplification operator $KiC(i+1)C_{re}(q,p)$, and *edge contraction*, a widely used operator in mesh processing which has been proven to be homology-preserving [25].

Acknowledgments

We wish to thank Davide Bolognini for many helpful discussions. This work has been partially supported by the Italian Ministry of Education and Research under the PRIN 2009 program, and by the National Science Foundation under Grant Number IIS-1116747. The 2D and 3D datasets are courtesy of the AIM@SHAPE shape repository.

Appendix A

In this appendix, we introduce some basic definitions, and we prove that our homology-preserving operators preserve the homology of a cell complex with the coefficients in any Abelian group (see [21,22] for a more rigorous treatment).

Given a cell complex X, it is possible to define the chain complex associated with X, denoted as $C_*(X) := (C_k(X), \partial_k)_{k\in\mathbb{Z}}$, where $\forall k$ $C_k(X)$ is the free Abelian group generated by the k-cells of the cell complex X and $\partial_k : C_k(X) \to C_{k-1}(X)$ is a homomorphism called *boundary map* which encodes the boundary relations between the k-cells and the $(k-1)$-cells of X such that $\partial^2 = 0$. We denote as $Z_k(X) := \ker \partial_k$ the group of the k-cycles of X and as $B_k(X) := \operatorname{im} \partial_{k+1}$ the group of the k-boundaries of X. Then, we define the kth homology group of X with coefficients in $\mathbb{Z}$ as

$$H_k(X) := H_k(C_*(X)) = \frac{Z_k(X)}{B_k(X)}$$

Furthermore, given an arbitrary Abelian group A, we can define the kth homology group with coefficients in A of X as $H_k(X;A) := H_k(C_*(X)\otimes_{\mathbb{Z}} A)$, where $\otimes_{\mathbb{Z}}$ is the tensor product of Abelian groups. If we consider $A = \mathbb{Z}_2$, $C_*(X)\otimes_{\mathbb{Z}}\mathbb{Z}_2 := (C_k(X)\otimes_{\mathbb{Z}}\mathbb{Z}_2, \partial_k\otimes_{\mathbb{Z}}\mathbb{Z}_2)_{k\in\mathbb{Z}}$ is the chain complex whose groups $C_k(X)\otimes_{\mathbb{Z}}\mathbb{Z}_2$ are just the $\mathbb{Z}_2$-vector spaces generated by the k-cells of X and the homomorphisms $\partial_k\otimes_{\mathbb{Z}}\mathbb{Z}_2$ are the boundary maps ∂_k of X considered modulo 2.

One important tool, which allows simplifying the homology computation of a cell complex, is *discrete Morse theory* (see [22]). This powerful theory is based on the idea of providing the cell complex X with a function $f : X \to \mathbb{R}$, called a *discrete Morse function* such that, for every cell x in X,

- $c^+(x) := \#\{y \text{ in the immediate co-boundary of } x \mid f(y) \leqslant f(x)\} \leqslant 1$,
- $c^-(x) := \#\{z \text{ in the immediate boundary of } x \mid f(z) \geqslant f(x)\} \leqslant 1$,
- if x is face of y with incidence greater than one, then $f(y) > f(x)$.

A cell x in X is *critical* if $c^+(x) = c^-(x) = 0$. The *discrete Morse complex* associated with X is a chain complex $\mathcal{M}_*$, whose groups $\mathcal{M}_k$ are generated by the critical k-cells of the function f. For each Abelian group A, we have that $H_k(X;A) \cong H_k(\mathcal{M}_*;A)$.

Proposition 1. *Operators $MiC(i+1)C$ and $KiC(i+1)C$ preserve the homology with coefficients in any Abelian group.*

Proof. Since $MiC(i+1)C$ are the inverse operators of $KiC(i+1)C$, it is sufficient to prove that $KiC(i+1)C$ are homology-preserving operators. Let X be a cell complex, $KiC(i+1)C$ the operator that deletes an i-cell p and an $(i+1)$-cell q from X and let Y be the resulting cell complex. Let $f : X \to \mathbb{R}$ be the discrete Morse function defined by

$$f(x) = \begin{cases} \dim x & \text{if } x \in X \setminus \{p,q\} \\ \frac{\dim p + \dim q}{2} = i + \frac{1}{2} & \text{otherwise} \end{cases}$$

The chain complex associated with Y represents the discrete Morse complex associated with X with respect to the function f. By Theorem 8.2 [22] and the Universal Coefficient Theorem [21], we conclude that $H_*(X;A) \cong H_*(Y;A)$ for any Abelian group A. □

Appendix B

In this appendix, we provide the proof of correctness of the algorithm (see Section 6) for modifying the homology generators (with coefficients in $\mathbb{Z}_2$) when moving from a lower to a higher resolution.

Proposition 2. *Let X be a d-dimensional cell complex, Y the cell complex obtained from X by applying $MiC(i+1)C(q,p,p')$. For a fixed $k \in \{0,\cdots,d\}$, let $B = \{[c_1]_X, \cdots, [c_l]_X\}$ be a basis for $H_k(X;\mathbb{Z}_2)$, then*

(1) if $k \neq i+1$, $\{[c_1]_Y, \cdots, [c_l]_Y\}$ is a basis for $H_k(Y;\mathbb{Z}_2)$;
(2) if $k = i+1$, $B' = \{[c'_1]_Y, \cdots, [c'_l]_Y\}$ is a basis for $H_{i+1}(Y;\mathbb{Z}_2)$, where, if $[c]_X \in B$, $[c']_Y \in B'$ is defined by

$$c' = \begin{cases} c & \text{if } \partial_Y c = 0 \\ c + q & \text{otherwise} \end{cases}$$

Proof. Throughout the proof and the statement we denote as ∂_X and ∂_Y the boundary maps $\partial_X\otimes_{\mathbb{Z}}\mathbb{Z}_2$ and $\partial_Y\otimes_{\mathbb{Z}}\mathbb{Z}_2$ respectively and all calculations are to be considered modulo 2. We give the proof for the case when the refinement operator is of the type *expand* ($MiC(i+1)C_{ex}(q,p,p')$). The case when the operator is of the type *insert* ($MiC(i+1)C_{in}(q,p,p')$) is dual.

In order to prove that a set of elements of $C_k(Y;\mathbb{Z}_2)$ is a basis for the $\mathbb{Z}_2$ vector space $H_k(Y;\mathbb{Z}_2)$ we have to show that:

(a) each element is in $Z_k(Y;\mathbb{Z}_2)$;
(b) each element is not in $B_k(Y;\mathbb{Z}_2)$;
(c) the elements are linearly independent.

(1) The only non-trivial cases are for $k = i+2, i, i-1$.

Case $k = i+2$.

(a) Let $c \in C_{i+2}(X;\mathbb{Z}_2)$ be such that $[c]_X$ is a basis element for $H_{i+2}(X;\mathbb{Z}_2)$. We can consider c as an element in $C_{i+2}(Y;\mathbb{Z}_2)$. We have that

$$\partial_Y c = \partial_X c + mq = mq$$

where $m \in \{0,1\}$.
Suppose that $m = 1$, i.e., $\partial_Y c = q$. Then, since $\partial_Y^2 = 0$,

$$\partial_Y q = \partial_Y^2 c = 0$$

But, $\langle \partial_Y q, p\rangle = 1$, so

$$\partial_Y q = p + \cdots \neq 0 \quad \lightning$$

Hence, $c \in Z_{i+2}(Y;\mathbb{Z}_2)$.

(b) Suppose $\exists b \in C_{i+3}(Y;\mathbb{Z}_2)$ such that $\partial_Y b = c$. Since no $(i+3)$-cell and no $(i+2)$-cell is created in Y, we have that $\partial_{Y,i+3} = \partial_{X,i+3}$, and $\partial_X b = \partial_Y b = c$. So $c \in B_{i+2}(X;\mathbb{Z}_2)$. $\lightning$

(c) $[c_1]_Y, \cdots, [c_l]_Y$ are linearly dependent in $H_{i+2}(Y;\mathbb{Z}_2)$
$\iff$ for each $j = 1, \cdots, l, \exists \alpha_j \in \{0,1\}$ such that $[\alpha_1 c_1 + \cdots + \alpha_l c_l]_Y = [0]_Y$ and at least one $\alpha_j \neq 0$
$\iff \exists b \in C_{i+3}(Y;\mathbb{Z}_2)$ such that $\partial_Y b = \sum_{j=1}^{l} \alpha_j c_j$
$\iff \exists b \in C_{i+3}(X;\mathbb{Z}_2)$ such that $\partial_X b = \sum_{j=1}^{l} \alpha_j c_j$
$\iff$ for each $j = 1, \cdots, l, \exists \alpha_j \in \{0,1\}$ such that $[\alpha_1 c_1 + \cdots + \alpha_l c_l]_X = [0]_X$ and at least one $\alpha_j \neq 0$
$\iff [c_1]_X, \cdots, [c_l]_X$ are linearly dependent in $H_{i+2}(X;\mathbb{Z}_2)$ $\lightning$

Case $k = i$.

(a) Let $c \in C_i(X;\mathbb{Z}_2)$ such that $[c]_X$ is a basis element for $H_i(X;\mathbb{Z}_2)$. Since p does not appear in c, we have that

$$\partial_Y c = \partial_X c = 0$$

(b) Suppose $\exists b \in C_{i+1}(Y;\mathbb{Z}_2)$ such that $\partial_Y b = c$. There are two cases.
If q does not appear in b, then $b \in C_{i+1}(X;\mathbb{Z}_2)$ and

$$\partial_X b = \partial_Y b + \langle \partial_Y q, p'\rangle mp' = c + \langle \partial_Y q, p'\rangle mp'$$

where $m = \sum_{s \in \{r \text{ } (i+1)\text{–cell in } Y \mid r \text{ is in } b\}} \langle \partial_Y s, p\rangle$ is the number of the $(i+1)$-cells in b in which p is incident in Y. Since $\partial_Y b = \cdots + mp$ and we know that $\partial_Y b = c$, then m has to be an even number and so $\partial_X b = c$. $\lightning$
If q appears in b, then $b - q \in C_{i+1}(X;\mathbb{Z}_2)$ and

$$\partial_X(b-q) = c - \langle \partial_Y q, p'\rangle p' + \langle \partial_Y q, p'\rangle mp' = c + \langle \partial_Y q, p'\rangle mp'$$

where $m = \sum_{s \in \{r \text{ } (i+1)\text{–cell in } Y \mid r \neq q, r \text{ is in } b\}} \langle \partial_Y s, p\rangle$ is the number of the $(i+1)$-cells in $b-q$ in which p is incident in Y. Since $\partial_Y(b-q) = \cdots + mp$ and we know that $\partial_Y(b-q) = \partial_Y b - \partial_Y q = c + \cdots + p$, then m has to be an odd number and so $\partial_X(b-q) = c$. $\lightning$

(c) Suppose that $[c_1]_Y, \cdots, [c_l]_Y$ are linearly dependent in $H_i(Y;\mathbb{Z}_2)$. This implies that, for each $j = 1, \cdots, l, \exists \alpha_j \in \{0,1\}$ such that $[\alpha_1 c_1 + \cdots + \alpha_l c_l]_Y = [0]_Y$ and at least one $\alpha_j \neq 0$, i.e., $\exists b \in C_{i+1}(Y;\mathbb{Z}_2)$ such that $\partial_Y b = \sum_{j=1}^{l} \alpha_j c_j$.
With arguments analogous to those used in (b), we can conclude that either $\partial_X b = \sum_{j=1}^{l} \alpha_j c_j$ or $\partial_X(b-q) = \sum_{j=1}^{l} \alpha_j c_j$ and so, $[c_1]_X, \cdots, [c_l]_X$ are linearly dependent in $H_i(X;\mathbb{Z}_2)$. $\lightning$

Case $k = i-1$.

(a) Let $c \in C_{i-1}(X;\mathbb{Z}_2)$ such that $[c]_X$ is a basis element for $H_{i-1}(X;\mathbb{Z}_2)$. We can consider c as an element in $C_{i-1}(Y;\mathbb{Z}_2)$. Since no $(i-1)$-cell and no $(i-2)$-cell is created in Y, we have that $\partial_{Y,i-1} = \partial_{X,i-1}$ and

$$\partial_Y c = \partial_X c = 0$$

(b) Suppose $\exists b \in C_i(Y;\mathbb{Z}_2)$ such that $\partial_Y b = c$. For all i-cell $s \neq p$ of Y, we have that

$$\partial_X s = \partial_Y s \qquad (*)$$

There are two cases.
If p does not appear in b, then, by $(*)$, we have that

$$\partial_X b = \partial_Y b = c \quad \lightning$$

Otherwise, if p appears in b, since $\partial_Y^2 = 0$, we have that

$$0 = \partial_Y^2 q = \partial_Y(p + (\partial_Y q - p)) = \partial_Y p + \partial_Y(\partial_Y q - p)$$

Hence,

$$\partial_Y p = \partial_Y(\partial_Y q - p) = \partial_X(\partial_Y q - p) \qquad (**)$$

Then, $b + \partial_Y q \in C_i(X;\mathbb{Z}_2)$ and

$$\begin{aligned} \partial_X(b + \partial_Y q) &= \partial_X(b + p - p + \partial_Y q) \\ &= \partial_X(b+p) + \partial_X(\partial_Y q - p) \\ &= \partial_Y b + \partial_Y p + \partial_X(\partial_Y q - p) \quad \text{by}(*) \\ &= \partial_Y b + \partial_Y p + \partial_Y p \quad \text{by}(**) \\ &= c \quad \lightning \end{aligned}$$

(c) Analogous to the previous case.

(2) Let's prove now the case $k = i+1$. We have to show that if $B = \{[c_1]_X, \cdots, [c_l]_X\}$ is a basis for $H_{i+1}(X;\mathbb{Z}_2)$, then $B' = \{[c_1']_Y, \cdots, [c_l']_Y\}$ is a basis for $H_{i+1}(Y;\mathbb{Z}_2)$.

(a) Let $c \in B$, i.e., $c \in C_{i+1}(X;\mathbb{Z}_2)$ such that $[c]_X$ is a basis element for $H_{i+1}(X;\mathbb{Z}_2)$. We want to prove that $\partial_Y c' = 0$.
If $\partial_Y c = 0$, then $c' = c$ and $\partial_Y c' = \partial_Y c = 0$.
Otherwise $\partial_Y c \neq 0$ and $c' = c + q$. In order to conclude, we want to show that $\partial_Y c = \partial_Y q$. This is indeed true, because we have that

$$\partial_Y c = \partial_X c - m\langle \partial_Y q, p'\rangle p' + mp = m(\langle \partial_Y q, p'\rangle p' + p)$$

where $m = \sum_{s \in \{r \text{ } (i+1)\text{–cell in } Y \mid r \text{ is in } c\}} \langle \partial_Y s, p\rangle$ is the number of the $(i+1)$-cells in c in which p is incident in Y. Since $\partial_Y c \neq 0, m$ is an odd number, and

$$\partial_Y c = \langle \partial_Y q, p'\rangle p' + p = \partial_Y q$$

As a consequence,

$$\partial_Y(c') = \partial_Y(c+q) = \partial_Y c + \partial_Y q = 0$$

(b) For each $(i+2)$-cell t in Y, we have that $\partial_X t = \partial_Y t - \langle \partial_Y t, q\rangle q$.
If $\exists b \in C_{i+2}(Y;\mathbb{Z}_2)$ such that $\partial_Y b = c'$ then

$$\partial_X b = \partial_Y b + mq = c' + mq$$

where $m = \sum_{t \in \{r \text{ } (i+2)\text{–cell in } Y \mid r \text{ is in } b\}} \langle \partial_Y t, q\rangle$.
Since m is even if $c' = c$ and m is odd if $c' = c + q$, we can conclude that

$$\partial_X b = c \quad \lightning$$

(c) Analogous to the previous case. □

References

[1] H. Lopes, S. Pesco, G. Tavares, M. Maia, A. Xavier, Handlebody representation for surfaces and its applications to terrain modeling, Int. J. Shape Model. 9 (1) (2003) 61–77.

[2] H. Lopes, G. Tavares, Structural operators for modeling 3-manifolds, in: Proceedings Fourth ACM Symposium on Solid Modeling and Applications, ACM Press, 1997, pp. 10–18.
[3] C.M. Eastman, K. Weiler, Geometric modeling using the Euler operators, in: 1st Annual Conference on Computer Graphics in CAD/CAM Systems, MIT, 1979.
[4] I.C. Braid, R.C. Hillyard, I.A. Stroud, Stepwise construction of polyhedra in geometric modelling, in: K.W. Brodlie (Ed.), Mathematical Methods in Computer Graphics and Design, Academic Press, 1980, pp. 123–141.
[5] M. Mantyla, R. Sulonen, GWB: a solid modeler with Euler operators, IEEE Comput. Graph. Appl. 2 (1982) 17–31.
[6] M. Mantyla, An Introduction to Solid Modeling, Computer Science Press, 1988.
[7] S. Lee, K. Lee, Partial entity structure: a fast and compact non-manifold boundary representation based on partial topological entities, in: Proceedings Sixth ACM Symposium on Solid Modeling and Applications, Ann Arbor, Michigan, 2001, pp. 159–170.
[8] H. Masuda, K. Shimada, M. Numao, S. Kawabe, A mathematical theory and applications of non-manifold geometric modeling, in: IFIP WG 5.2/GI International Symposium on Advanced Geometric Modeling for Engineering Applications, North-Holland, Berlin, Germany, 1989, pp. 89–103.
[9] H. Masuda, Topological operators and boolean operations for complex-based non-manifold geometric models, Comput. Aided Des. 25 (2) (1993) 119–129.
[10] A.J.P. Gomes, Euler operators for stratified objects with incomplete boundaries, in: Proceedings of the Ninth ACM Symposium on Solid Modeling and Applications, SM '04, IEEE Computer Society, Los Alamitos, CA, USA, 2004, pp. 315–320.
[11] G. Damiand, P. Lienhardt, Removal and contraction for n-dimensional generalized maps, DGCI, Lecture Notes in Computer Science, vol. 2886, Springer, 2003, pp. 408–419.
[12] G. Damiand, R. Gonzalez-Diaz, S. Peltier, Removal operations in nD generalized maps for efficient homology computation, Proc. of 4th International Workshop on Computational Topology in Image Context (CTIC), Lecture Notes in Computer Science, vol. 7309, Springer, Berlin/Heidelberg, 2012, pp. 20–29.
[13] M. Baba-ali, G. Damiand, X. Skapin, D. Marcheix, Insertion and expansion operations for n-dimensional generalized maps, in: D. Coeurjolly, I. Sivignon, L. Tougne, F. Dupont (Eds.), DGCI, Lecture Notes in Computer Science, vol. 4992, Springer, 2008, pp. 141–152.
[14] Y. Matsumoto, An introduction to Morse theory, Vol. 208 of Translations of mathematical monographs, Am. Math. Soc. 208 (2002).
[15] L. Čomić, L. De Floriani, Topological operators on cell complexes in arbitrary dimensions, in: CTIC, Lecture Notes in Computer Science, vol. 7309, 2012, pp. 98–107.
[16] L. Brun, W.G. Kropatsch, Introduction to combinatorial pyramids, Digital and Image Geometry, Lecture Notes in Computer Science, vol. 2243, Springer, 2000, pp. 108–128.
[17] C. Simon, G. Damiand, P. Lienhardt, Pyramids of n-dimensional generalized maps, in: Proc. of 5th Workshop on Graph-Based Representations in Pattern Recognition (GBR), Lecture Notes in Computer Science, vol. 3434, Springer, Berlin/Heidelberg, 2005, pp. 142–152.
[18] L. Čomić, L. De Floriani, F. Iuricich, Multi-resolution cell complexes based on homology-preserving Euler operators, in: R. González-Díaz, M.J. Jiménez, B. Medrano (Eds.), DGCI, Lecture Notes in Computer Science, vol. 7749, Springer, 2013, pp. 323–334.
[19] M.K. Agoston, Computer Graphics and Geometric Modeling: Mathematics, Springer-Verlag London Ltd., 2005. ISBN: 1-85233-817-2.
[20] S. Peltier, A. Ion, Y. Haxhimusa, W.G. Kropatsch, G. Damiand, Computing homology group generators of images using irregular graph pyramids, in: GbRPR, Lecture Notes in Computer Science, vol. 4538, Springer, 2007, pp. 283–294.
[21] A. Hatcher, Algebraic Topology, Cambridge University Press, 2001. <http://www.math.cornell.edu/hatcher/AT/AT.pdf>.
[22] R. Forman, Morse theory for cell complexes, Adv. Math. 134 (1998) 90–145.
[23] H. Edelsbrunner, Algorithms in Combinatorial Geometry, Springer Verlag, Berlin, 1987.
[24] T. Kaczyński, M. Mrozek, M. Ślusarek, Homology computation by reduction of chain complexes, Comput. Math. Appl. 35 (4) (1998) 59–70. http://dx.doi.org/10.1016/S0898-1221(97)00289-7.
[25] D. Attali, A. Lieutier, D. Salinas, Efficient data structure for representing and simplifying simplicial complexes in high dimensions, Int. J. Comput. Geom. Appl. 22 (4) (2012) 279–304. http://dx.doi.org/10.1142/S0218195912600060.