Multi-resolution Cell Complexes Based on Homology-Preserving Euler Operators

Lidija Čomić[1], Leila De Floriani[2], and Federico Iuricich[2]

[1] University of Novi Sad, Faculty of Technical Sciences
comic@uns.ac.rs
[2] University of Genova, Department of Computer Science
deflo@disi.unige.it,
federico.iuricich@unige.it

Abstract. We have proposed a complete set of basis Euler operators for updating cell complexes in arbitrary dimensions, which can be classified as homology-preserving and homology-modifying. Here, we define the effect of homology-preserving operators on the incidence graph representation of cell complexes. Based on these operators, we build a multi-resolution model for cell complexes represented in the form of the incidence graph, and we compare its 2D instance with the pyramids of 2-maps, designed for images.

Keywords: geometric modeling, cell complexes, homology-preserving operators, multi-resolution representations.

1 Introduction

Cell complexes, together with simplicial complexes, have been used as a modeling tool in a variety of application domains. Several data structures have been designed in the literature for representing the connectivity of a cell complex (incidence and adjacency relations among the cells in the complex), such as incidence graphs, introduced in [6], and n-maps, introduced informally in [7].

Many topological operators have been designed for building and updating data structures representing 2D and 3D cell complexes. In [3], we have proposed a set of Euler operators which form a minimally complete basis for building and updating cell complexes in arbitrary dimensions in a topologically consistent manner. We distinguish between operators that preserve the homology of the complex, and the ones 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 (decrease) the ith Betti number.

Here, we define the effect of homology-preserving operators on the incidence graph, based on which we build a multi-resolution model for the topology of the complex, that we call the *Multi-Resolution Cell Complex* (MCC). We present some experimental results validating the MCC, and we compare its 2D instance with the pyramidal model used for images represented in the form of a 2-map.

R. Gonzalez-Diaz, M.-J. Jimenez, B. Medrano (Eds.): DGCI 2013, LNCS 7749, pp. 323–334, 2013.

2 Background Notions

We review some notions on the topology of cell complexes (see [1] for details).

A *k-cell* in the Euclidean space $\mathbb{E}^n$ is a homeomorphic image of an open k-dimensional ball, and a *cell d-complex* in $\mathbb{E}^n$ is a finite set Γ of cells in $\mathbb{E}^n$ of dimension at most d, $0 \leq d \leq n$, such that (i) the cells in Γ are pairwise disjoint and (ii) for each cell $\gamma \in \Gamma$, the boundary of γ is a disjoint union of cells of Γ.

Intuitively, an n-dimensional quasi-manifold is an n-dimensional complex which can be obtained by gluing together n-cells along $(n-1)$-cells (for details see [11]). In a quasi-manifold, an $(n-1)$-cell belongs to the boundary of at most two n-cells. The notion of quasi-manifold is weaker that the notion of pseudo-manifold. Recall that a simplicial complex Σ is a *pseudo-manifold* if (i) Σ is homogenous (each simplex is a face of some n-simplex), (ii) each $(n-1)$-simplex in Σ is an $(n-1)$-face of at most two n-simplexes and (iii) Σ is strongly connected (for any two distinct n-simplexes σ and τ in Σ there is a sequence $\sigma = \sigma_1, \sigma_2,..,\sigma_k = \tau$, such that σ_i and σ_{i+1} share an $(n-1)$-simplex, $1 \leq i < n$).

A variety of data structures have been proposed for representing the topology of cell complexes. Some represent the cells in the complex explicitly, e.g. incidence graphs, which can be used to represent arbitrary cell complexes, and abstract cellular complexes [9]. Some represent them implicitly, e.g. n-maps, which are used to represent orientable quasi-manifolds without boundaries.

An *Incidence Graph* (*IG*) [6] representing a cell complex Γ is a multigraph $G = (N, A)$, such that:

1. the set of nodes N is partitioned into $n+1$ subsets N_0, N_1,...,N_n, such that there is a one-to-one correspondence between the nodes in N_i (which we call *i-nodes*) and the i-cells of Γ,
2. there are k arcs joining an i-node p with an $(i+1)$-node q if and only if i-cell p appears k times on the boundary of $(i+1)$-cell q in Γ.

We model the incidence (multi-)graph as an ordinary labeled graph, in which each node is labeled with the dimension of the corresponding cell, and each arc between two nodes is labeled with its multiplicity φ (the number of arcs between the two nodes in the corresponding multi-graph). If Γ is a simplicial complex then all the arcs in A are simple (with label equal to one).

An *n-map* (or n-dimensional *combinatorial map*) [2] is a finite set D of elements, called *darts*, together with n permutations β_i on D, $1 \leq i \leq n$, such that β_i is an involution, $2 \leq i \leq n$, and $\beta_i \circ \beta_j$ is an involution, $i+2 \leq j$, $i, j \in \{1, , ..., n\}$. Intuitively, a dart in D corresponds to an $(n+1)$-tuple of cells $(c_0, .., c_n)$, where c_i is an i-cell, $0 \leq i \leq n$, and each c_i is on the boundary of c_{i+1}. For an n-map $M = (D, \beta_1, .., \beta_n)$, $n \geq 2$, and a dart b in D, the 0-cell incident in b is the set of all darts that can be reached starting from b by applying any combination of permutations in the set $\{\beta_1^{-1} \circ \beta_2, .., \beta_1^{-1} \circ \beta_n\}$; the i-cell incident in b, $1 \leq i \leq n$, is obtained by applying permutations in $\{\beta_1, .., \beta_n\} \backslash \{\beta_i\}$. 2-maps are widely used for image processing and geometric modeling. In the 2D case, permutations β_1 and β_2 are usually denoted as σ and α, respectively.

The Euler-Poincaré formula expresses the necessary validity condition of a cell complex with manifold or non-manifold carrier [1]. The Euler-Poincaré formula for a cell d-complex Γ with n_i i-cells states that

$$\sum_{i=0}^{d}(-1)^i n_i = n_0 - n_1 + .. + (-1)^d n_d = \sum_{i=0}^{d}(-1)^i b_i = b_0 - b_1 + .. + (-1)^d b_d.$$

Here, b_i is the ith Betti number of Γ, and it measures the number of independent non-bounding i-cycles in Γ, i.e., the number of independent i-holes.

3 Related Work

A general idea of multi-resolution modeling is to provide several decompositions of a shape at different, uniform or variable, scales. We review related work on a hierarchical model for cell complexes, called *combinatorial* (or *n-map) pyramid.*

A 2-map pyramid [2] is a hierarchical data structure used for image analysis. Each level in a 2-map pyramid is a 2-map. The first level describes the initial full-resolution data; the other levels describe successive reductions of the previous levels. Usually, a pixel in the initial full-resolution 4-connected image is represented as a vertex in a 2D cubical complex, and adjacency relation between pixels is represented through edges in the complex. The reduction is obtained by applying operators that merge regions in the lower level into one region in the successive level (called *contraction operators*) and simplify the boundaries between the new merged regions (called *removal operators*). Each region in a coarser resolution image is a (connected) set of vertices, the representative of a region is an element of this set, called a *surviving vertex*, and other elements are called *non-surviving vertices.*

More formally, a 2-map $(m+1)$-level pyramid P is the set $P = \{G^k\}_{0 \leq k \leq m}$ of 2-maps such that for each k, $0 < k \leq m$, G^k is obtained from G^{k-1} by contracting the cells (edges) in a set of cells C^{k-1} (contraction kernel) and removing the cells (edges) in a set of cells R^{k-1} (removal kernel). Several strategies have been proposed to choose the sets of the removed and contracted cells [8].

Another general multi-resolution framework, used mainly for simplicial complexes, called a *Multi-Complex*, has been introduced in [5].

4 Homology-Preserving Euler Operators

We review the Euler operators on cell complexes, proposed in [3], and we define the effect of homology-preserving Euler operators on the IG representing them.

4.1 Homology-Preserving Euler Operators on Cell Complexes

Operators that modify a cell complex, by modifying the number of cells in the complex and its Betti numbers, and maintain the validity of Euler-Poincaré formula, are called *Euler operators*. In the literature, a variety of sets of basis Euler operators have been proposed, mainly for the 2D and the 3D case.

In [3], we have proposed a minimal set of Euler operators on cell complexes in arbitrary dimensions, which subsume 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*),
– homology-modifying operators: $MiCiCycle$ (*Make i-Cell and i-Cycle*).

Homology-preserving operators $MiC(i+1)C$ change the number of cells in the complex Γ, by increasing the number n_i of i-cells and the number n_{i+1} of $(i+1)$-cells by one. The Euler characteristic and the Betti numbers of the complex remain unchanged. Homology-preserving operator $MiC(i+1)C$ can create two new cells p and q from an existing i- or $(i+1)$-cell, or insert the new cells in the complex.

The first type of $MiC(i+1)C$ operator has two instances. It either splits an existing i-cell p' in two by splitting its co-boundary, and creates an $(i+1)$-cell q bounded by the two i-cells p and p', or dually, it splits an existing $(i+1)$-cell p' into two by splitting its boundary, and creates an i-cell q separating the two $(i+1)$-cells p and p'. In both cases, the created i-cell appears exactly once on the boundary of the created $(i+1)$-cell.

The second type of $MiC(i+1)C$ operator 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. In both cases, the created i-cell appears exactly once on the boundary of the created $(i+1)$-cell.

Figure 1 illustrates a sequence consisting of $M0C1C(p_1, q_1)$ (second type, second instance), $M1C2C(p_2, q_2)$ (first type, second instance) and $M0C1C(p_3, q_3)$ (first type, first instance) in 2D. Figure 2 illustrates a sequence consisting of $M1C2C(p_1, q_1)$ and $M2C3C(p_2, q_2)$ (both of first type, second instance) in 3D. For brevity, we will consider only the operators of the first type.

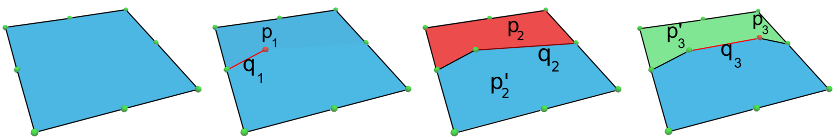

Fig. 1. A sequence consisting of $M0C1C$, $M1C2C$ and $M0C1C$ on a 2D cell complex; $M0C1C$ creates 0-cell p_1 and 1-cell q_1, $M1C2C$ creates 1-cell q_2 and 2-cell p_2, $M0C1C$ creates 0-cell p_3 and 1-cell q_3

The inverse $KiC(i+1)C$ (*Kill i-Cell and (i+1)-Cell*) operators delete an i-cell and an $(i+1)$-cell from Γ. The first type of $KiC(i+1)C$ operator is feasible in the following two cases:

(i) the deleted $(i+1)$-cell q is bounded by exactly two i-cells (the deleted i-cell p and the non-deleted i-cell p') and the deleted i-cell p appears exactly once on the boundary of $(i+1)$-cell q;

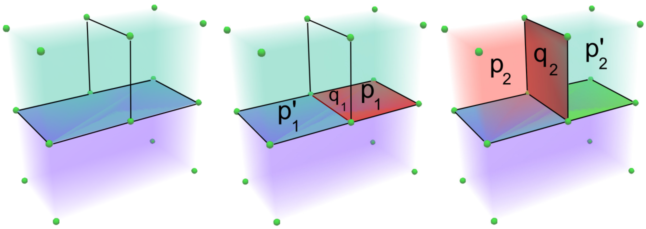

Fig. 2. A sequence consisting of $M1C2C$ and $M2C3C$ on a 3D cell complex; $M1C2C$ creates 1-cell q_1 and 2-cell p_1, $M2C3C$ creates 2-cell q_2 and 3-cell p_2

(ii) the deleted i-cell q bounds exactly two $(i+1)$-cells (the deleted $(i+1)$-cell p and the non-deleted $(i+1)$-cell p') and the deleted i-cell q appears exactly once on the boundary of $(i+1)$-cell p.

In the first case, the effect of the operator is that the deleted i-cell p is replaced with the non-deleted i-cell p' in the boundary of each $(i+1)$-cell r in the co-boundary of the deleted i-cell p. One copy of $(i+1)$-cell q is merged into $(i+1)$-cell r for each time i-cell p appears on the boundary of $(i+1)$-cell r. The second case is dual.

Homology-modifying operators change both the number of cells in the complex Γ and its Betti numbers, and they change the Euler characteristic of Γ. They increase the number n_i of i-cells and the number b_i of non-bounding i-cycles by one. The inverse *KiCiCycle* (*Kill i-Cell and i-Cycle*) operators delete an i-cell and destroy an i-cycle, thus decreasing the numbers n_i and b_i by one.

4.2 Homology-Preserving Euler Operators on Incidence Graphs

$KiC(i+1)C$ operator on an IG $G = (N, A)$ deletes an i-node and an $(i+1)$-node from N, and suitably reconnects the remaining nodes. Its first instance is feasible on IG G if

- $(i+1)$-node q is connected to exactly two different i-nodes p and p', and
- there is exactly one arc in A connecting $(i+1)$-node q and i-node p.

The effect of $KiC(i+1)C(p, q)$ on G is that

- nodes p and q, all the arcs incident in $(i+1)$-node q and all the arcs incident in i-node p and connecting p to $(i-1)$-nodes are deleted,
- all the arcs incident in i-node p and connecting p to $(i+1)$-nodes are replaced with arcs connecting i-node p' to the same $(i+1)$-nodes for each arc connecting i-node p' to $(i+1)$-node q.

In terms of the ordinary labeled incidence graph, let us denote as $\varphi'(p', r)$ the label of the arc (p', r) after the simplification, where r is an $(i+1)$-node connected to the deleted i-node p. The label $\varphi'(p', r)$ is increased by the product of the label of the arc connecting nodes p' and q and the label of the arc connecting nodes p and r ($\varphi'(p',r) = \varphi(p',r) + \varphi(p',q) \cdot \varphi(p,r)$).

The second instance of the $KiC(i+1)C$ operator can be expressed as a modification of the IG $G = (N, A)$ in a completely dual fashion.

The inverse $MiC(i+1)C$ on an IG $G = (N, A)$ also has two instances. The first instance is specified by the two inserted nodes ($(i+1)$-node q and i-node p), the i-node p' that is the only i-node apart from i-node p that will be connected to $(i+1)$-node q, the $(i+2)$-nodes that will be connected to $(i+1)$-node q, and the $(i-1)$-nodes and $(i+1)$-nodes that will be connected to i-node p, together with the multiplicity (labels φ') of all the inserted arcs. It is feasible if all the specified nodes are in N, and the label $\varphi(p',r)$ before the refinement for each $(i+1)$-node r that will be connected to i-node p is greater than or equal to $\varphi'(p',q) \cdot \varphi'(p,r)$. Its effect is to add nodes p and q in N and all the specified arcs in A and to set $\varphi'(p',r) = \varphi(p',r) - \varphi'(p',q) \cdot \varphi'(p,r)$. The second instance has a completely dual effect.

4.3 Homology-Preserving Operators on 2-Maps

Simplification operators have been defined on 2-maps in terms of elimination of darts from set D and modifications of permutations on the remaining darts. The simplification operators are called *removal* and *contraction*. They are the same as $K0C1C$ and $K1C2C$ operators, respectively.

5 Multi-resolution Model

We have defined and implemented a multi-resolution model for the topology of cell complexes represented through an IG, that we call a *Multi-Resolution Cell Complex (MCC)*. It is generated from the IG representing the cell complex at full resolution by iteratively applying $KiC(i+1)C$ operators. The IG $G_B = (N_B, A_B)$ obtained as a result of a specific simplification sequence (determined by the error criterion adopted) applied to the initial full-resolution graph is the coarsest representation of the topology of the complex, and we call it the *base graph*. It is the first ingredient of the multi-resolution model.

The second ingredient is the set $\mathcal{M}$ of refinement operators, inverse to the simplification operators applied in the simplification process.

The third ingredient of the multi-resolution model is a dependency relation $\mathcal{R}$ on the set $\mathcal{M}$ plus μ_0, where μ_0 is a dummy refinement that generates $G_B = (N_B, A_B)$. We define a dependency relation between refinements in $\mathcal{M} \cup \mu_0$ as follows: refinement μ, which introduces nodes p and q, *directly depends* on refinement μ^* if and only if μ^* creates at least one node that is connected to either p or q by μ. The transitive closure of the direct dependency relation defined above is a partial order relation, since a node is never introduced twice by the refinements in $\mathcal{M}$. A multi-resolution model for the topology of cell complexes is, thus, a triple $MCC = (G_B, \mathcal{M}, \mathcal{R})$, where G_B is the IG representing the cell complex at the coarsest resolution, $\mathcal{M}$ is the set of refinements inverse to the simplifications applied in the generalization process, and $\mathcal{R}$ is the direct dependency relation defined over $\mathcal{M}$.

The MCC can be encoded as a Directed Acyclic Graph (DAG), in which the root corresponds to modification μ_0, i.e., to the creation of the base graph G_B, the other nodes correspond to the modifications in $\mathcal{M}$, and the arcs represent the direct dependency relation $\mathcal{R}$.

6 Selective Refinement

We discuss how to extract a large number of adaptive representations from an $MCC = (G_B, \mathcal{M}, \mathcal{R})$ and briefly discuss some algorithmic aspects.

The set $\mathcal{U} = \{\mu_0, \mu_1, \mu_2, ..., \mu_m\} \subseteq \mathcal{M}$ of refinements in $\mathcal{M}$ is *closed* with respect to dependency relation $\mathcal{R}$ if for each $1 \leq l \leq m$ in $\mathcal{U}$, each refinement on which refinement μ_l depends is in $\mathcal{U}$. Let $U = (\mu_0, \mu_1, \mu_2, ..., \mu_m)$ be a sequence of the refinements belonging to $\mathcal{U} \subseteq \mathcal{M}$, such that, for each $\mu_l \in U$ and each refinement ν on which μ_l depends, $\nu = \mu_j \in U$, $0 \leq j < l$. Then, U is called a *feasible sequence.* The *front graph* G_U associated with a feasible sequence U is the graph obtained from the base graph G_B by applying the sequence of refinements U. It can be shown that any two feasible sequences U_1 and U_2 obtained from the same closed set $\mathcal{U}$ produce the same front graph. Thus, a closed subset $\mathcal{U}$ of refinements can be applied to the base IG G_B in any total order U that extends the partial order, producing an IG G_U at an intermediate resolution. If a feasible sequence U contains all refinements in $\mathcal{M}$, then the front graph G_U associated with U is the same as the IG at full resolution.

An MCC encodes the collection of all representations of a cell complex, at intermediate levels of resolution, which can be obtained from the base representation G_B by applying a closed set of modifications on G_B. From an MCC it is thus possible to dynamically extract representations of the topology of a cell n-complex 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 on an MCC consists of extracting from it the IG with the minimum number of nodes, satisfying some application-dependent criterion. This criterion can be formalized by defining a Boolean function τ over all nodes of an MCC, such that the value of τ is *true* on nodes which satisfy the criterion, and $false$ otherwise. An IG $G = (N, A)$ is said to satisfy a criterion τ if function τ assumes the value *true* on all nodes in N. Thus, a selective refinement query consists of extracting from the MCC an intermediate graph of minimum size that satisfies τ. Equivalently, it consists of extracting a minimal closed set $\mathcal{U}$ of modifications from $\mathcal{M}$ such that the corresponding complex satisfies τ. Such closed set of modifications uniquely determines a front graph, which is obtained from the base graph $G_B = (N_B, A_B)$ by applying to it all modifications from $\mathcal{U}$ in any order that is consistent with the partial order defined by the dependency relation. The 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 the maximum resolution is required (while in the rest of the complex, the resolution may be arbitrarily low).

We have implemented a depth-first algorithm for the selective refinement query. The algorithm starts from the coarse IG G_B and recursively applies to it all refinements μ_i which are required to satisfy the error criterion. In order that a new modification μ be applied, all its ancestor modifications need to be applied before μ to maintain the partial order. It can easily be proven that the result of a selective refinement algorithm is a graph $G = G_U$ with minimal number of nodes among the graphs that can be extracted from the MCC, such that all nodes in G_U satisfy criterion τ.

7 Experimental Results

The purpose of experiments is twofold. In the first set of experiments, we have tested two simplification strategies to build the MCC: one approach is based on performing simplifications one at the time, and the other on performing a set of independent simplifications. In the second set, we show the capabilities of the MCC to extract adaptive representations at variable resolutions, and compare timings for the two approaches. We have performed the experiments on 2D and 3D simplicial complexes available on the Web and encoded in an IG, that become cell complexes after undergoing some simplification. The initial size of these complexes is between 400K and 953K triangles for 2D data sets, and between 68K and 577K tetrahedra for 3D data sets. Experiments have been performed on a desktop computer with a 3.2Ghz processor and 16Gb of memory.

To build the MCC, we start from the IG at full resolution and perform all the feasible simplifications in the order guided by some criterion τ until the coarsest representation is reached. The implementation of the simplification algorithm is independent of criterion τ. We have used a geometric criterion computed on the vertices of the deleted cells, and we have implemented two different simplification approaches. In the first one, called *step-by-step simplification*, simplifications are extracted from the priority queue in ascending order and performed if still feasible. After each simplification, the local connectivity of the nodes involved in it changes and each new feasible simplification is enqueued. The algorithm ends when there are no more feasible simplifications.

The second approach, called *batch simplification*, tries to execute at each step a large number of feasible independent simplifications (that involve nodes not involved by any other already selected simplification). At each step, we build a priority queue with all the feasible simplifications sorted in ascending order. We select a set of simplifications from the queue, we perform all of them, and we initialize a new priority queue.

In Table 1 we summarize the results obtained with the two approaches. The columns show, from left to right, data set name (*Data set*), total number of cells (*Cells*), number of simplifications (*Simpl. Num.*), time needed to perform them (*Simpl. Time*), time needed to build the MCC (*MCC Time*), storage cost of the *MCC* (*MCC storage*), time needed to perform all the refinements in the *MCC* (*Ref. Time*), storage cost of the cell complex at full resolution (*Full complex*) and storage cost of the base complex (*Base complex*).

Table 1. Experimental results for the DAG construction. The storage cost is expressed in Megabytes and the computation time in seconds.

	Data set	Cells	Simpl. Num.	Simpl. Time	MCC Time	MCC storage	Ref Time	Full complex	Base complex
2D				Step-by-step simplification					
	Eros	2859566	1429781	74.4	5.3	254.9	18.1	349.0	0.0002
	Hand	1287532	643694	35.4	2.3	117.2	7.58	157.1	0.01
	VaseLion	1200002	599999	26.7	2.1	105.8	6.8	146.4	0.00028
				Batch simplification					
	Eros	2859566	1429781	218.8	6.4	241.0	18.7	349	0.0002
	Hand	1287532	643741	99	2.6	120.7	7.6	157.1	0.004
	VaseLion	1200002	599999	90.7	2.3	110.5	7.7	146.4	0.00028

	Data set	Cells	Simpl. Num.	Simpl. Time.	MCC Time.	MCC storage	Ref Time	Full complex	Base complex
3D				Step-by-step simplification					
	VisMale	297901	147594	45.1	0.6	40.4	5.1	48	0.46
	Bonsai	1008357	498790	380.6	2.7	146.9	27.2	162.5	1.8
	Hydrogen	2523927	1248743	8643.8	7.8	395.7	419.5	407.4	4.4
				Batch simplification					
	VisMale	297901	148116	69.2	0.7	37.6	2.5	48	0.28
	Bonsai	1008357	501524	305.8	2.69	126.4	10.4	162.5	0.89
	Hydrogen	2523927	1253913	1412.9	7.4	321.3	33.9	407.4	2.7

We can notice that the time needed to perform all the refinements is always much less than the time needed to perform all the simplifications (refinement is 5 to 10 times faster than simplification). An important aspect is that the storage cost of the *MCC* structure plus the base graph is less than the storage cost of the graph at full resolution, with the exception of the largest tested (*Hydrogen*) data set using the step-by-step method. Although the total number of simplifications is slightly higher for the batch simplification approach, the time required to perform all simplifications that lead to the base complex is less in the case of step-by-step simplification, since it requires fewer computations. On the other hand, the *MCC* generated through batch simplification uses less memory and consequently can be visited in less time. We have observed that

Table 2. Experimental timing results (in seconds) for extraction at variable resolution

2D				3D			
Data set	Perc.	Refinement Time step-by-step	batch	Data set	Perc.	Refinement Time step-by-step	batch
	50%	0.80	0.92		50%	3.45	0.12
Eros	80%	1.42	1.01	*VisMale*	80%	3.77	0.15
	100%	2.63	2.60		100%	4.01	0.53
	50%	0.31	0.57		50%	15.3	0.65
Hand	80%	0.45	0.65	*Bonsai*	80%	17.4	0.69
	100%	1.20	1.19		100%	19.1	1.88
	50%	0.73	0.69		50%	106.3	8.1
VaseLion	80%	1.01	0.99	*Hydrogen*	80%	127.7	8.7
	100%	1.10	1.06		100%	172.1	11.3

the DAGs produced by the batch simplification have less dependency relations compared to the ones produced by step-by-step simplification.

In Table 2, we show timing results for performing extractions at variable resolution. Column *Perc.* indicates the desired percentage of operations performed inside a query box. $Refinement\,Time$ indicates the time needed to perform the required number of refinements with the step-by-step method (*step-by-step*) or the batch (*batch*) simplification methods. The query box has been chosen by hand to cover a relevant part for each data set and with size between 15 and 30 percent of the whole data set at full resolution. We can observe that the extraction times for refinements are similar for the two methods in the 2D case, while they differ considerably in the 3D case. Note that in 2D each 1-node in the incidence graph is connected with at most two different 0-nodes and two different 2-nodes, while in 3D there is a variable number of arcs between 1-nodes and 2-nodes: a larger number of arcs in the IG leads to a larger number of dependency relations in the MCC. This has a a significant impact in the use of a simplification method that reduces the MCC complexity.

In Figure 3, we show examples of refinement queries at uniform and variable resolution performed on the *VaseLion* data set. The holes that seem to appear in the crown of the lion are rendering artifacts.

8 Discussion and Outlook

We compare the 2D instance of the MCC defined on IGs with the pyramid model defined on 2-maps.

The first advantage of the MCC over pyramidal models is its space efficiency. This is a consequence of the fact that the IG occupies less memory than the n-map representing the same complex. Each dart in an n-map corresponds to a path in the IG representing the same n-dimensional cell complex from an

(a) (b) (c) (d) (e)

Fig. 3. In (a), (b) and (c) the representations obtained from the *MCC* after 10000, 50000 and 2000000 refinements, respectively. In (d), the complex at full resolution of the *VaseLion* data set. In (e) the representation obtained with a query at variable resolution.

n-node to a 0-node. For each dart, the set of n involutions is encoded plus a pointer for each entity which points to the geometric and attribute description of such entities, as discussed in [10]. This leads to a storage cost of $B * (2n + 1)$ items, where B is the number of darts. For $n = 2$, it can be easily seen that $B = 4n_1$, where n_1 denotes the number of 1-cells in the complex, while the number of arcs in the IG is equal to $4n_1$ (they can be encoded through $8n_1$ pointers), and the number of nodes is equal to the total number of cells in the complex. In general, we can observe that each path in the IG is defined by a set of $n - 1$ arcs, and the storage cost is less than $B * (n - 1)$ items, since the paths overlap. We have evaluated on a set of 2-complexes and 3-complexes, the ratio between the storage cost for the IG and for the n-map; the value for this ratio is around 50% for 2-complexes, and around 18% for 3-complexes.

The second advantage is a wider representation domain. IGs can represent arbitrary cell complexes, while n-maps can represent (closed orientable) quasi-manifolds, which are a class of pseudo-manifolds.

We plan to apply the homology-preserving operators to the computation of homology of a cell complex. An arbitrary cell complex at full resolution can be simplified by applying a sequence of homology-preserving operators, until no further simplification is possible. Homology can be computed on the simplified complex using standard techniques [1]. Homology generators on the simplified complex can be computed using the method similar to the ones in [12,4], proposed for images and complexes represented as n-G-maps, respectively, and propagated from such complex to the full-resolution complex using the MCC.

Acknowledgments. 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.

References

1. Agoston, M.K.: Computer Graphics and Geometric Modeling: Mathematics. Springer-Verlag London Ltd. (2005) ISBN:1-85233-817-2
2. Brun, L., Kropatsch, W.G.: Introduction to Combinatorial Pyramids. In: Bertrand, G., Imiya, A., Klette, R. (eds.) Digital and Image Geometry. LNCS, vol. 2243, pp. 108–128. Springer, Heidelberg (2001)
3. Čomić, L., De Floriani, L.: Topological Operators on Cell Complexes in Arbitrary Dimensions. In: Ferri, M., Frosini, P., Landi, C., Cerri, A., Di Fabio, B. (eds.) CTIC 2012. LNCS, vol. 7309, pp. 98–107. Springer, Heidelberg (2012)
4. Damiand, G., Gonzalez-Diaz, R., Peltier, S.: Removal Operations in nD Generalized Maps for Efficient Homology Computation. In: Ferri, M., Frosini, P., Landi, C., Cerri, A., Di Fabio, B. (eds.) CTIC 2012. LNCS, vol. 7309, pp. 20–29. Springer, Heidelberg (2012)
5. De Floriani, L., Magillo, P., Puppo, E.: Multiresolution Representation of Shapes Based on Cell Complexes. In: Bertrand, G., Couprie, M., Perroton, L. (eds.) DGCI 1999. LNCS, vol. 1568, pp. 3–18. Springer, Heidelberg (1999)
6. Edelsbrunner, H.: Algorithms in Combinatorial Geometry. Springer, Berlin (1987)
7. Edmonds, J.: A Combinatorial Representation for Polyhedral Surfaces. Notices Amer. Math. Soc. 7, 646 (1960)
8. Haxhimusa, Y., Glantz, R., Kropatsch, W.G.: Constructing Stochastic Pyramids by MIDES - Maximal Independent Directed Edge Set. In: Hancock, E.R., Vento, M. (eds.) GbRPR 2003. LNCS, vol. 2726, pp. 24–34. Springer, Heidelberg (2003)
9. Kovalevsky, V.: Axiomatic Digital Topology. Journal of Mathematical Imaging and Vision 26(1-2), 41–58 (2006)
10. Lévy, B., Mallet, J.L.: Cellular Modelling in Arbitrary Dimension Using Generalized Maps. Technical report, INRIA (1999)
11. Lienhardt, P.: N-Dimensional Generalized Combinatorial Maps and Cellular Quasi-Manifolds. International Journal of Computational Geometry and Applications 4(3), 275–324 (1994)
12. Peltier, S., Ion, A., Haxhimusa, Y., Kropatsch, W.G., Damiand, G.: Computing Homology Group Generators of Images Using Irregular Graph Pyramids. In: Escolano, F., Vento, M. (eds.) GbRPR 2007. LNCS, vol. 4538, pp. 283–294. Springer, Heidelberg (2007)