ACM Symposium on Solid Modeling and Applications (2004)
G. Elber, N. Patrikalakis, P. Brunet (Editors)

Update operations on 3D simplicial decompositions of non-manifold objects

Leila De Floriani and Annie Hui

Department of Computer and Information Sciences, University of Genova,
Via Dodecaneso, 35 - 16146 Genova (Italy).
Department of Computer Science, University of Maryland,
College Park, MD 20742 (USA).
Email: deflo@disi.unige.it, huiannie@cs.umd.edu

Abstract

We address the problem of updating non-manifold mixed-dimensional objects, described by three-dimensional simplicial complexes embedded in 3D Euclidean space. We consider two local update operations, edge collapse and vertex split, which are the most common operations performed for simplifying a simplicial complex. We examine the effect of such operations on a 3D simplicial complex, and we describe algorithms for edge collapse and vertex split on a compact representation of a 3D simplicial complex, that we call the Non-Manifold Indexed data structure with Adjacencies (NMIA). *We also discuss how to encode the information needed for performing a vertex split and an edge collapse on a 3D simplicial complex. The encoding of such information together with the algorithms for updating the NMIA data structure form the basis for defining progressive as well as multi-resolution representations for objects described by 3D simplicial complexes and for extracting variable-resolution object descriptions.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling -*Curve, surface, solid and object representations*

1. Introduction

We consider the problem of locally updating a three-dimensional simplicial complex describing a non-manifold object by performing edge collapses and vertex splits. An *edge collapse* applied to a simplicial complex contracts an edge e of the complex into a new vertex v, or into one of the extreme vertices of e. *Vertex split* is the inverse of edge collapse: it replaces a vertex v of a complex with an edge e. Edge collapse and vertex split are common operations performed for coarsening or refining a complex in simplification algorithms [10, 25].

The ultimate goal of our work is to develop a continuous multi-resolution representation for non-manifold objects described by three-dimensional simplicial decompositions, i.e., by combinations of tetrahedral meshes with lower dimensional entities described by chains of edges, or by triangle meshes. In order to develop a multi-resolution representation, we need a data structure for encoding 3D simplicial complexes, that can be used to describe the variable-resolution complexes extracted from a multi-resolution model. Such a data structure must be compact, and must support efficient navigation within the complex through adjacencies, as required by common geometric modeling operations. Since most objects encountered in the applications contain a relatively small number of non-manifold singularities, the data structure should scale to the manifold case with a small overhead. Moreover, we need algorithms to perform edge collapse and vertex split on such representation as well as a compact way of encoding vertex splits and edge collapses. This will be the basis for designing a data structure for representing the multi-resolution model.

In [5], we have defined a model for non-manifold objects described by simplicial complexes, called a *Non-Manifold Multi-Tessellation (NMT)*, and we have described an implementation of the NMT for the case of two-dimensional complexes. Here, we plan to develop some basic tools for designing and implementing an NMT in the three-dimensional case.

In [4], we have defined a compact data structure, called a *Non-Manifold Indexed data structure with Adjacencies (NMIA)*, for describing three-dimensional simplicial complexes embedded in the 3D Euclidean space. The NMIA data structure encodes the vertices and the top simplexes of the complex plus a restricted subset of topological relations, and it supports retrieval of incidence and adjacency relations among the entities in the complex in time linear in the number of entities involved in the relations. Here, we briefly describe the entities and relations defining the NMIA data structure, and an improved implementation of such structure, which exhibits an even better scalability to the manifold case, i.e., a very low overhead cost with respect to a similar data structure, the indexed data structure with adjacencies, commonly used to represent manifold simplicial meshes.

In [10], conditions are studied under which an edge collapse can be performed on a complex without changing its topological type, which have been applied in simplification algorithms for 3D simplicial complexes with a manifold domain to avoid creating non-manifold situations [2]. Here, we allow edge collapses and vertex splits to change the topological type of the domain of the simplicial complex as well as to create, or remove, lower-dimensional entities, such as wire edges and dangling faces.

In [25], Popovic and Hoppe describe the effect of applying a generalization of edge collapse, called vertex-pair collapse, which consists of collapsing a pair of vertices (not necessarily connected by an edge), into a vertex, without imposing restrictions on topology changes. They sketch updating algorithms, which assume a verbose representation of the complex as an incidence graph, but they do not specify how topological relations in the incidence graph are affected. The storage cost of an incidence graph is three times the cost of the NMIA data structure. The restricted number of entities and relations encoded in the NMIA data structure makes updating algorithm considerably more complex.

Novel contributions of this paper are:

- a compact and scalable implementation of the NMIA data structure;
- an analysis of the effect of edge collapse and of vertex split on a 3D simplicial complex;
- algorithms for performing edge collapse and vertex split on the NMIA data structure;
- an encoding of vertex split to be used as the basis for a progressive volumetric representation of the object and for a data structure for encoding a 3D Non-Manifold Multi-Tessellation.

The remainder of this paper is organized as follows. In Section 2, we summarize some background notions. In Section 3, we review some related work. In Section 4, we describe the NMIA data structure, and we discuss our new implementation and its storage costs. In Section 5, we analyze the effect of edge collapse and vertex split on a three-dimensional simplicial complex. In Section 6, we describe an algorithm for performing edge collapse on a 3D simplicial complex encoded in the NMIA data structure. In Section 7, we present an encoding for the vertex split operation, and we describe an algorithm for performing vertex split. In Section 8, we present an instance of the NMT based on 3D simplicial complexes, generated through edge collapse. Finally, in Section 9, we draw some conclusions and discuss future work.

2. Background

In this Section, we review some basic notions about Euclidean simplicial complexes in arbitrary dimensions, and the topological relations among the cells of a complex.

A *manifold* object is a subset of the Euclidean space for which the neighborhood of each internal point is homeomorphic to an open ball. Objects, that do not fulfill this property at one or more points, are called *non-manifold* objects.

A Euclidean simplex of dimension d is the convex hull of $d+1$ linearly independent points in the 3-dimensional Euclidean space E^3, with $d \leq 3$. We simply call a *Euclidean d-simplex* a *d-simplex*: a 0-simplex is a *vertex*; a 1-simplex an *edge*; a 2-simplex a *triangle*; a 3-simplex a *tetrahedron*. d is called the *dimension* of σ and is denoted as $dim(\sigma)$. Any Euclidean k-simplex, with $k < d$, σ' generated by a set $V_{\sigma'} \subseteq V_\sigma$ of cardinality $k+1 \leq d$ is called a *k-face* of σ.

A finite collection Σ of Euclidean simplexes is a *Euclidean simplicial complex* when (i) for each simplex $\sigma \in \Sigma$, all faces of σ belong to Σ, and (ii) for each pair of simplexes σ and σ', either $\sigma \cap \sigma' = \emptyset$ or $\sigma \cap \sigma'$ is a face of both σ and σ'. The *domain*, or *carrier*, of a Euclidean simplicial complex Σ embedded in E^3, with $d \leq 3$, is the subset of E^3 defined by the union, as point sets, of all the simplexes in Σ. We will consider in the paper three-dimensional simplicial complexes (3-complexes) embedded in E^3.

The *boundary* $b(\sigma)$ of a simplex σ is defined as the set of all faces of σ. Similarly, the *co-boundary*, or *star*, of a simplex σ is defined as $\star\sigma = \{\xi \in \Sigma \mid \sigma \text{ is a face of } \xi\}$. Simplexes ξ in $\star\sigma$ are called *co-faces* of σ. In the following, we will call *restricted star* of a simplex σ, $\star\sigma - \{\sigma\}$, and we will denote it as $st(\sigma)$. The *link* of a simplex σ, denoted $link(\sigma)$, is the set of all faces of the co-faces of σ, that are not incident at σ. Any simplex σ such that $\star\sigma = \{\sigma\}$ is called a *top simplex* of Σ.

Two distinct simplexes are said to be *incident* if one of them is a face of the other. Two simplexes are called *k-adjacent* if they share a k-face. For example, a tetrahedron and a triangle are *1-adjacent* if they share an edge. Two p-simplexes, with $p > 0$, are said to be *adjacent* if they are $(p-1)$-adjacent. Two vertices (i.e., 0-simplexes) are called *adjacent* if they are both incident at a common 1-simplex.

An *h-path* is a sequence of simplexes $(\sigma_i)_{i=0}^k$ such that two successive simplexes σ_{i-1}, σ_i in the sequence are h-adjacent. Two simplexes σ and σ' are *h-connected* if and only if there exists an h-path $(\sigma_i)_{i=0}^k$ such that σ is a face of σ_0 and σ' is a face of σ_k. A subset Σ' of a complex Σ is called *h-connected* if and only if every pair of its vertices are h-connected. Any maximal h-connected sub-complex of a complex Σ is called an *h-connected component* of Σ.

A complex Σ, where all top simplexes are maximal (i.e., of dimension d), is called *regular*. A regular, $(d-1)$-connected complex in which each $(d-1)$-simplex is on the boundary of one or two d-simplexes is called a *pseudo-manifold*. Note that every 3-complex embedded in E^3 is always a pseudo-manifold. A pseudo-manifold satisfying the additional property that all its vertices have a link combinatorially equivalent to the $(d-1)$-dimensional sphere is called a *(combinatorial) manifold*. The domain of a Euclidean simplicial complex, which is described by a combinatorial manifold, is a manifold in E^3.

Let Σ be a d-complex and let $\sigma \in \Sigma$ be a p-simplex, with $0 \leq p \leq d$. For each integer value q, $0 \leq q \leq d$, we define the *topological relation* $R_{pq}(\sigma)$ as a retrieval function that returns the q-simplexes of Σ that are not disjoint from σ. In particular:

- For $p < q$, $R_{pq}(\sigma)$ consists of the set of simplexes of order q in the star of σ.
- For $p > q$, $R_{pq}(\sigma)$ consists of the set of simplexes of order q in the set of faces of σ.
- For $p > 0$, $R_{pp}(\sigma)$ is the set of p-simplexes in Σ that are $(p-1)$-adjacent to σ.
- $R_{00}(v)$, where v is a vertex, consists of the set of vertices w such that $\{v, w\}$ is a 1-simplex of Σ.

Relation R_{pq} is called a *boundary relation* if $p > q$, a *co-boundary relation* if $p < q$, and an *adjacency relation* if $p = q$. Boundary and co-boundary relations are called *incidence relations*. Boundary relations in a *3-complex* are R_{30}, R_{31}, R_{32}, R_{20}, R_{21} and R_{10}. Adjacency relations in a *3-complex* are R_{33}, R_{22}, R_{11} and R_{00}. Co-boundary relations are R_{23}, R_{12}, R_{13}, R_{01}, R_{02} and R_{03}.

3. Related Work

In this Section, we review related work on data structures for describing three-dimensional simplicial and cell complexes, and on simplification algorithms for tetrahedral meshes (i.e., simplicial complexes with a manifold domain).

The *Facet-Edge* data structure [11] and the *Handle-Face* data structure [23] have been proposed for three-dimensional cell complexes. Both describe 3-cells implicitly by encoding the 2-manifold complexes that form the boundary of such cells. The most common representation for three-dimensional simplicial complexes with a manifold domain is the *Indexed data structure with Adjacencies (IA)*, which directly extends to arbitrary dimension, being called *winged representation* [24]. In a d-dimensional IA data structure, vertices and top simplexes are encoded together with R_{d0} and R_{dd} relations. The IA data structure can be extended by encoding only one incident d-simplex for each vertex v, thus allowing efficient navigation around a vertex. In three dimensions, the storage cost of the indexed data structure with adjacencies is equal to $8n_t + n$ integers, where n_t is the number of tetrahedra and n is the number of vertices of the complex.

Dimension-independent data structures have been proposed for d-dimensional manifold complexes, which include the *Cell Tuple* [1], and the *n-G-maps* [22] for cellular complexes. When used to describe simplicial complexes, the storage cost of such data structures is much higher than that of the IA data structure, for a factor that grows combinatorially with the dimension of the complex (see [8]). The representation domain of all such data structures is larger than that of d-manifolds: the IA data structure can describe Euclidean pseudo-manifolds embedded in the Euclidean d-dimensional space, while the other two can describe a sub-class of pseudo-manifolds introduced in [22], and called *cellular quasi-manifolds*. Selective Geometric Complexes (SGCs) [27] can describe non-manifold and non-regular objects through cell complexes whose cells can be even open, or not simply connected. In SGCs, cells and their mutual adjacencies are encoded in an incidence graph [12].

An alternative approach to the design of non-manifold data structures consists of decomposing a non-manifold object into simpler and more manageable parts [6, 9, 14, 18, 26]. Most proposals decompose the two-dimensional boundaries of r-sets [9, 14]. The algorithm presented in [26] tries to minimize the number of duplications introduced by the decomposition process. In [18], the idea of cutting a two-dimensional non-manifold complex into manifold pieces is exploited to develop compression algorithms. In [6], a decomposition for d-dimensional non-manifold objects described through simplicial complexes is defined, which is unique and produces a description of a d-complex (not necessarily embeddable in the Euclidean space) as a combination of nearly manifold components. A dimension-independent data structure for such decomposition is defined in [7], which describes the components and their connectivity in a two-level representation.

Data structures for non-manifold, non-regular, three-dimensional cell complexes have been proposed for modeling non-manifold solids [19, 20, 30]. Experimental evaluations reported in [21] show that these data structures do not scale well to the manifold case, since their storage cost is between 2.1 and 4.4 times higher than that of the *winged edge* data structure. The *partial entity structure* [21] has been shown to require half of the space of the radial-edge structure introduced in [20]. A data structure for encoding any two-dimensional simplicial complex, called the *triangle-segment* data structure, has been proposed in [5], which extends the IA data structure to deal with non-regular (1-dimensional) parts and with non-manifold adjacencies of 2-simplexes at an edge. This data structure is compact, and scales very well to the manifold case, since it requires just one byte per vertex more than the IA data structure when applied to a manifold complex.

Mesh simplification has been extensively studied for triangle meshes (see, e.g., [16] for a survey). Some techniques have been extended to the case of tetrahedral meshes. In [15], Farias et al. propose a non-incremental decimation method based on vertex clustering. The techniques proposed in [2, 17, 28] are all based on edge collapse and differ in the way they control the error for producing a simplified mesh. Gross and Staadt [17] present a decimation technique based on collapsing an edge to an arbitrary interior point, and propose various cost functions to drive the collapsing process. Cignoni et al. [2] propose an algorithm based on collapsing an edge to one of its extreme vertices (called *half-edge collapse*), in which the simplification process is driven by a combination of the geometric error introduced in simplifying the shape of the domain and of the error introduced in approximating the scalar field with fewer points. In [29], Véron and Léon describe how to perform simplification on a two-dimensional simplicial complex embedded in the three-dimensional Euclidean space. The complex is simplified through vertex removal as the primary operation, and then through edge collapse and face removal as secondary operations.

Popovic and Hoppe [25] consider general arbitrary-dimensional simplicial complexes, and describe how to perform vertex-pair collapse and its inverse by considering only how the entities in the complex are updated. In [10], a set of necessary and sufficient conditions to preserve the topological type of a simplicial complex when simplified through edge collapse are studied.

4. The NMIA Data Structure

The NMIA data structure describes 3D simplicial complexes containing one- and two-dimensional top simplexes, that we call *wire edges* and *dangling faces*, respectively, (see Figure 1(a)), in order to represent parts of different dimensionalities in the object. Also, it describes situations in which the restricted star of an edge, or of a vertex, consists of more than one connected component. At such edges and vertices, the manifold condition does not hold (see Figures 1(b) and 1(c)). Since we are dealing with 3-complexes embedded in E^3, any 2-simplex, which is not a top simplex, must be on the boundary of either one or two tetrahedra.

We call each 2-connected component in the restricted star of an edge an *edge-based cluster*. Edge-based clusters can be ordered around the edge, for instance, in a counter-clockwise direction. An edge-based cluster consists either of a single dangling face, or of a

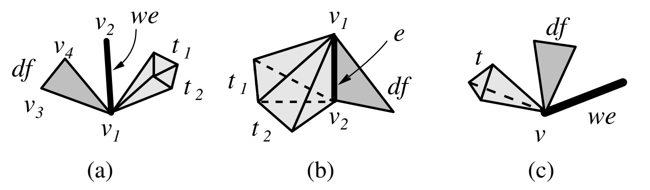

Figure 1: *(a) a wire edge, $we = \{v_1, v_2\}$, and a dangling face, $df = \{v_1, v_3, v_4\}$, (b) $e = \{v_1, v_2\}$, $st(e)$ has two connected components, (c) $st(v)$ has three connected components*

2-connected set of tetrahedra sharing edge e (see Figure 1b). Similarly, each 1-connected component of the restricted star of a vertex is called a *vertex-based cluster*. If two simplexes σ_I and σ_j belong to the same vertex-based cluster in $st(v)$, then there exists a 1-path passing through only those simplexes which form the cluster.

To describe edge- and vertex-based clusters, we introduce three auxiliary relations, that represent the incidence of the edge-based clusters at an edge, and of the vertex-based clusters at a vertex:

- Relation $R_{0,clusters}(v)$ is a retrieval function which associates, with vertex v, one representative k-simplex for each vertex-based cluster incident at v. If dangling faces and tetrahedra belong to the same cluster, a tetrahedron is chosen.
- Relation $R_{2,clusters}(f)$ is a retrieval function which associates, with each edge e_i of a dangling face f, the two edge-based clusters incident at e_i, and following and preceding face f in a counter-clockwise order around e_i.
- Relation $R_{3,clusters}(t)$ is a retrieval function which associates, with each edge e_i of a tetrahedron t, at most two edge-based clusters incident at e_i, and following and preceding tetrahedron t in a counter-clockwise order around e_i.

Note that each edge-based cluster at an edge e is represented either by a dangling face incident at e, or by a tetrahedron, if the edge-based cluster consists of just one tetrahedron, or by two tetrahedra, i.e., the two tetrahedra that do not have 2-adjacent neighbors along one of their two faces incident at e.

4.1. Entities and Relations

In the NMIA data structure, *vertices*, *tetrahedra*, *wire edges* and *dangling faces* are encoded as entities. Edges bounding a face or a tetrahedron, or faces bounding a tetrahedron are not explicitly represented.

The following relations are stored:

- for each tetrahedron t:
  - relation $R_{30}(t)$, which associates, with each tetrahedron t, its four vertices ordered according to the orientation chosen for t.
  - relation $R_{33}(t)$, which associates, with each tetrahedron t, the four tetrahedra adjacent to t through a 2-simplex, (the i-th tetrahedron in $R_{33}(t)$ is the one that does not contain the i-th vertex of t).
  - relation $R_{3,clusters}(t)$, as defined above, for each of the six edges of tetrahedron t (considered in an order compatible with the orientation of t).
- for each dangling face f:
  - relation $R_{20}(f)$, which associates, with each dangling face f, its three vertices (ordered according to the orientation chosen for f).
  - relation $R_{2,clusters}(f)$, as defined above, for each of the three edges of dangling face f (considered in an order compatible with the orientation of f).
- for each wire edge e, relation $R_{10}(e)$, which associates, with edge e, its two extreme vertices.
- for each vertex v, relation $R_{0,clusters}(v)$, as defined above.

4.2. Implementation

$R_{30}(t)$, $R_{20}(f)$ and $R_{10}(e)$ relations are encoded as arrays of indexes for each tetrahedron t, dangling face f and wire edge e, respectively. $R_{33}(t)$ relation is encoded as a fix-sized array A of four elements. If a tetrahedron t has less than four adjacent neighbors, then the last element of array A stores a pointer to a variable-sized array B encoding $R_{3,clusters}(t)$. A 1-bit flag $f_1(t)$ is used to indicate whether $R_{3,clusters}(t) \neq \emptyset$. In the first element of array B, we store three flags, namely:

- a 4-bit flag, $f_2(t)$, which indicates which 2-adjacent neighbors are present in $R_{33}(t)$,
- a variable-sized flag, $f_3(t)$, which indicates on which edges tetrahedron t has 1-adjacent neighbors related through $R_{3,clusters}$ relation,
- a variable-sized flag, $f_4(t)$, which indicates whether 1-adjacent neighbors in $R_{3,clusters}(t)$ are tetrahedra or dangling faces.

For each dangling face f, relation $R_{2,clusters}$ is encoded in a similar way. Two flags, $f_5(f)$ and $f_6(f)$, are used to encode the positions and the values of 1-adjacent neighbors related to f through $R_{2,clusters}$ relation.

For each vertex v, $R_{0,clusters}(v)$ is encoded as follows:

- a 1-bit flag $f(v)$ is used to indicate whether the restricted star of v, $st(v)$, consists of one cluster formed by a 2-connected set of tetrahedra
- a field which contains either the index of a tetrahedron in $st(v)$, if $st(v)$ is a 2-connected set of tetrahedra, or a link to a list of representative elements (tetrahedra, dangling faces, or wire edges), one for each vertex-based cluster in $st(v)$.

4.3. Storage Cost

Let n_t, n_d, n_w, n denote the number of tetrahedra, dangling faces, wire edges, and vertices in a simplicial complex Σ, respectively. Let c_v denote the sum of all vertex-based clusters over all non-manifold vertices of Σ. Let c_e denote the sum of all edge-based clusters over all non-manifold edges of Σ. Let n_b be the number of tetrahedra t such that $R_{3,clusters}(t) \neq \emptyset$.

The storage cost of the NMIA data structure is equal to $(8n_t + 5n_d + 2n_w + n_b + 2c_e + 2(c_v - n) + n)$ integers + $(n_t + n)$ bits, by assuming that pointers and indexes are stored as integers on 32 bits.

In the case of manifold complexes, the cost of the NMIA data structure reduces to $(8n_t + n)$ integers + $(n_t + n)$ bits, since such complexes do not contain dangling faces ($n_d = 0$), wire edges ($n_w = 0$), non-manifold edges ($c_e = 0$), or non-manifold vertices

($c_v - n = 0$). For tetrahedral complexes used as the decomposion of the domain of a three-dimensional scalar field, it has been shown experimentally that $n_t \approx 6n$ [3]. Thus the cost of the NMIA data structure becomes $49n$ integers + n bytes.

The extended IA data structure exhibits the same performances as the NMIA in the manifold case. The extended IA data structure encodes, for each tetrahedron t, $R_{30}(t)$ and $R_{33}(t)$ relations and, for each vertex v, a link to one tetrahedron incident at v. The storage cost is, thus, equal to $8n_t + n$ integers, which becomes $49n$ integers under the assumption that $n_t \approx 6n$. Thus, the NMIA data structure exhibits an overhead of just one byte per vertex due to the representation of non-manifold singularities.

5. Edge Collapse and Vertex Split in a 3D Simplicial Complex

In this Section, we discuss the effect of applying an edge collapse or a vertex split on a three-dimensional simplicial complex Σ.

5.1. Edge Collapse

Let $e = \{v_1, v_2\}$ be the edge to be collapsed in a 3D simplicial complex Σ. Let Σ' be the complex resulting from Σ by collapsing edge e into a vertex v.

An *edge collapse* applied to $e = \{v_1, v_2\}$ in Σ consists of replacing edge e with the new vertex v in Σ', collapsing all dangling faces and tetrahedra in the restricted star of e, $st(e)$, to wire edges and faces incident at v, respectively, and in updating all the simplexes in $st(v_1) \cup st(v_2)$, i.e., all the simplexes incident at either v_1 or v_2.

Given a k-simplex, σ, $k = 1,2,3$, in Σ, such that $\sigma \in st(e) \cup st(v_1) \cup st(v_2)$, either σ is incident at edge e, or σ is incident at v_1 or v_2 but not at both. In the former case, σ must be either a dangling face or a tetrahedron. In the latter case, there are two possible situations. Let us assume that σ is incident at v_1, then either σ intersects some other simplex $\overline{\sigma}$ incident at the other vertex, v_2, or σ has an empty intersection with all other simplexes incident at v_2.

Therefore, the following three cases may occur:

1. $\sigma \in st(e)$: in this case, σ can either be a 2-simplex or a 3-simplex.
2. $\sigma \in st(v_1)$, $\sigma \notin st(v_2)$ and there exists $\overline{\sigma} \in st(v_2)$ such that $\sigma \cap \overline{\sigma} \neq \emptyset$: in this case, σ is incident at v_1 but not in v_2, and there exists a simplex $\overline{\sigma}$ incident at v_2 which intersects σ. This case also includes the symmetric situation in which $\sigma \in st(v_2)$ and $\sigma \notin st(v_1)$ and there exists $\overline{\sigma} \in st(v_1)$ such that $\sigma \cap \overline{\sigma} \neq \emptyset$.
3. $\sigma \in st(v_1)$, $\sigma \notin st(v_2)$ and $\sigma \cap \overline{\sigma} = \emptyset$, for every $\overline{\sigma} \in st(v_2)$: in this case, σ is incident at v_1 and not in v_2 and it does not have any intersection with simplexes incident at v_2. This case also includes the symmetric situation in which $\sigma \in st(v_2)$ and $\sigma \notin st(v_1)$ and $\forall \overline{\sigma} \in st(v_1)$, $\sigma \cap \overline{\sigma} = \emptyset$.

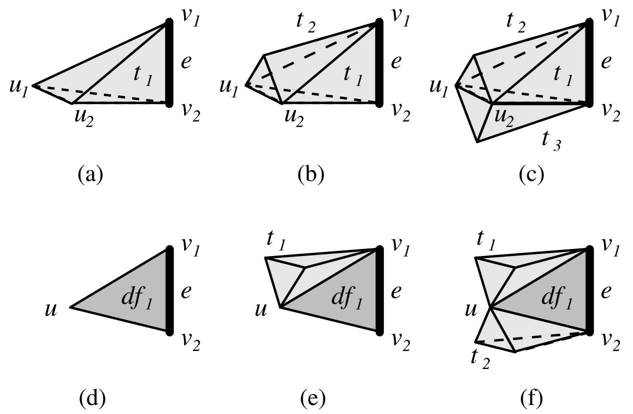

Figure 2: *Six examples of simplexes that are incident at the edge $e = \{v_1, v_2\}$ to be collapsed: in (a), (b) and (c), the simplexes incident at e are tetrahedron t_1 and two of its faces, $\{u_1, v_1, v_2\}$ and $\{u_2, v_1, v_2\}$, i.e., $st(e) = \{t_1, \{u_1, v_1, v_2\}, \{u_2, v_1, v_2\}\}$. In (d), (e) and (f), the dangling face df_1 is incident at e, i.e., $st(e) = \{df_1\}$. (Case 1)*

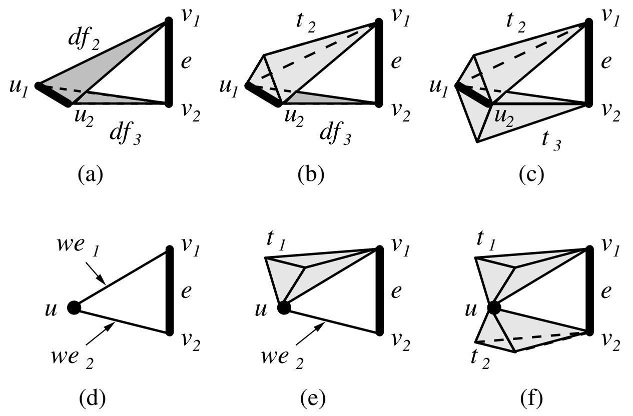

Figure 3: *Six examples of simplexes that are incident at one vertex of the edge e to be collapsed, and that have non-empty intersection with some other simplexes that are incident at the other vertex. In (a), dangling face df_2 is incident at v_1 and dangling face df_3 is incident at v_2. Their intersection is edge $\{u_1, u_2\}$. (b) and (c) are similar to (a), except that the simplex incident at either vertex may also be a tetrahedron. In (d), wire edges we_1 and we_2 are incident at v_1 and v_2, respectively, and their intersection is vertex u. Similarly, in (e) and (f), the intersection is at vertex u. (Case 2)*

Figure 2 gives six examples of simplexes that are incident at edge $e = \{v_1, v_2\}$ to be collapsed as in case 1. In Figures 2(a) to 2(c), $st(e)$ consists of tetrahedron t_1, and of its two faces $\{u_1, v_1, v_2\}$ and $\{u_2, v_1, v_2\}$ which are incident at e. In Figures 2(d) to 2(f), $st(e)$ consists of dangling face df_1. In Figure 3, six examples are given of simplexes that are incident at only one extreme vertex of the edge to be collapsed, and that intersect some simplexes that are incident at the other extreme vertex. As in Figure 2, e is the edge to be collapsed. In Figures 3(a) to 3(c), the non-empty intersection of those simplexes incident at v_1 and those incident at v_2 is edge $\{u_1, u_2\}$. In the other three examples, the non-empty intersection is vertex u. Figure 4(a) shows an example where a simplex incident at v_1 does not intersect any simplex incident at v_2.

Let λ: $st(e) \rightarrow st(v)$ be a map such that $\sigma' = \lambda(\sigma) \in \Sigma'$ is obtained from σ by replacing $\{v_1, v_2\}$ with v. We call it a *dimension-reduction map*. Let μ: $st(v_1) \cup st(v_2) - st(e) \rightarrow st(v)$ be a map that, given a simplex $\sigma \in st(v_1) \cup st(v_2) - st(e)$, provides $\sigma' = \mu(\sigma)$ such that $\sigma' \in st(v)$ and $\sigma' = \{v\} \cup \{w \mid \forall w \in \sigma - \{v_i\}, v = 1,2\}$. We call map μ a *simplex transformation map*.

When performing edge collapse, we apply:

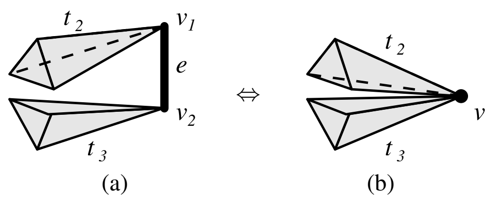

Figure 4: *An example in which a simplex, that is incident at one vertex of the edge e to be collapsed, has an empty intersection with all other simplexes that are incident at the other vertex. (a) Tetrahedra t_2 and t_3 are incident at v_1 and v_2, respectively, and they do not intersect each other. (Case 3) (b) When edge e is collapsed, the two tetrahedra become 0-adjacent.*

- in case 1: a reduction map λ for every k-simplex $\sigma \in st(e)$ into a $(k-1)$-simplex σ', $k = 2, 3$.
- in cases 2 and 3: a transformation map μ.

Figure 5 and Figure 6 show the effect of edge collapse on each of the examples in discussed in Figures 2 and 3. In each example, the drawing on top is a reproduction of that in Figures 2 and 3. The drawing at the bottom shows what happens after edge e is collapsed. In Figures 5(a) to 5(c), tetrahedron t_1 is incident at e. After edge collapse, t_1 may be become a dangling face, as in Figure 5(a), a boundary face, as in Figure 5(b), or an internal face, as in Figure 5(c). In Figures 5(d) to 5(f), df_1 is incident at e. After edge collapse, df_1 may become a wire edge, as in Figure 5(d), or a boundary edge, as in Figures 5(e) and 5(f). Figure 6 is similar. The result of edge collapse on the example of Figure 4(a) is shown in Figure 4(b). Note that, in all the cases, at most two simplexes are merged into one simplex as the effect of transformation map μ.

5.2. Vertex Split

Vertex split is the inverse operation with respect to edge collapse. It consists of splitting a vertex v in a 3D simplicial complex Σ into an edge $e = \{v_1, v_2\}$. The k-simplexes in $st(v)$ either expand into $(k+1)$-simplexes forming $st(e)$, or become incident at v_1 or v_2, or are duplicated.

Given a k-simplex $\sigma' \in st(v)$, the following three cases may occur:

1. σ' is expanded into a simplex σ in $st(e)$. In this case, we apply the inverse λ^{-1} of the dimension-reduction map λ which maps $\sigma' \in st(v)$ into $\sigma \in st(e)$ such that $\lambda(\sigma) = \sigma'$. This is equivalent to replacing v in σ' with $\{v_1, v_2\}$. (Each of the examples of Figure 5, when read from bottom to top, illustrates the effect of vertex split.)
2. $\sigma' \in st(v)$ is mapped into two k-simplexes σ_a and σ_b such that $\sigma_a \in st(v_1)$ and $\sigma_b \in st(v_2)$ and $\sigma' = \mu(\sigma_a) = \mu(\sigma_b)$. (See the examples of Figure 6 from bottom to top.)
3. σ' is transformed into one simplex σ incident at v_1 or in v_2. In this case, we map $\sigma' \in st(v)$ into $\sigma \in st(v_1) \cup st(v_2)$ such that $\sigma' = \mu(\sigma)$ by replacing v in σ with v_1 or v_2. (See Figure 4 from right to left.)

6. Performing an Edge Collapse

Performing the collapse of an edge $e = \{v_1, v_2\}$ into a vertex v in a complex Σ requires specifying just edge e and vertex v.

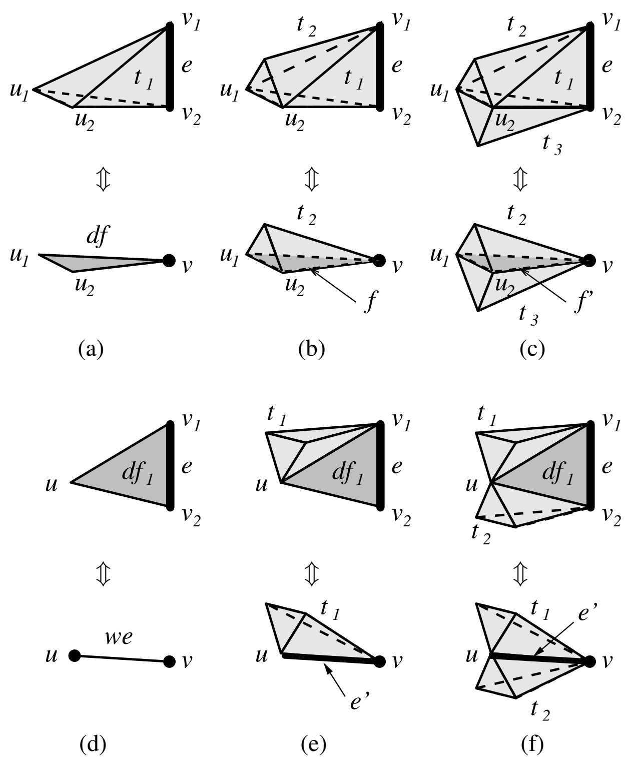

Figure 5: *Effect of edge collapse on the six examples shown in Figure 2. Edge $e = \{v_1, v_2\}$ is collapsed into vertex v. In (a), (b) and (c), tetrahedron t_1 becomes a new dangling face, df, an external face, f, of tetrahedron t_2, or an internal face, f', shared by tetrahedra t_2 and t_3, respectively. In (d), dangling face df_1 becomes a wire edge, we. In (e) and (f), df_1 becomes a boundary edge, e', of simplexes incident at both u and the new vertex v.*

R_{k0} relation is affected for all k-simplexes in $st(v_1) \cup st(v_2)$. R_{33}, $R_{3,clusters}$, $R_{2,clusters}$ and $R_{0,clusters}$ relations are affected for all simplexes belonging to $st(v_1) \cup st(v_2)$, or adjacent to simplexes in $st(v_1) \cup st(v_2)$. In this Section, we discuss in detail the different situations which can arise and show how the entities and the relations in the NMIA data structure have to be updated. Based on this analysis, we present an algorithm for performing edge collapse.

As discussed in Subsection 5.1, case 1, in which a k-simplex is reduced to a $(k-1)$-simplex and case 2, in which two k-simplexes may be merged into a single one, require more attention. We enumerate here the different situations which may arise in these two cases. This will also help us encoding the inverse of edge collapse, vertex split.

To this aim, we consider $L = link(v_1) \cap link(v_2)$. L is a collection of vertices and edges which can be ordered clockwise or counterclockwise around edge $e = \{v_1, v_2\}$. If e is a manifold edge, then L is homeomorphic to a circle or to a portion of a circle. If e is a non-manifold edge, then L is composed of several connected components, one for each edge-based cluster incident at e. Each component may consist of an isolated vertex or of a chain of edges. Figure 7 gives an example of $L = link(v_1) \cap link(v_2)$. Edge $e = \{v_1, v_2\}$ is the edge to be collapsed. $st(v_1)$ is composed of tetrahedra t_1, t_2, t_4 and of all their faces. $link(v_1)$ consists of the simplexes in $st(v_1)$

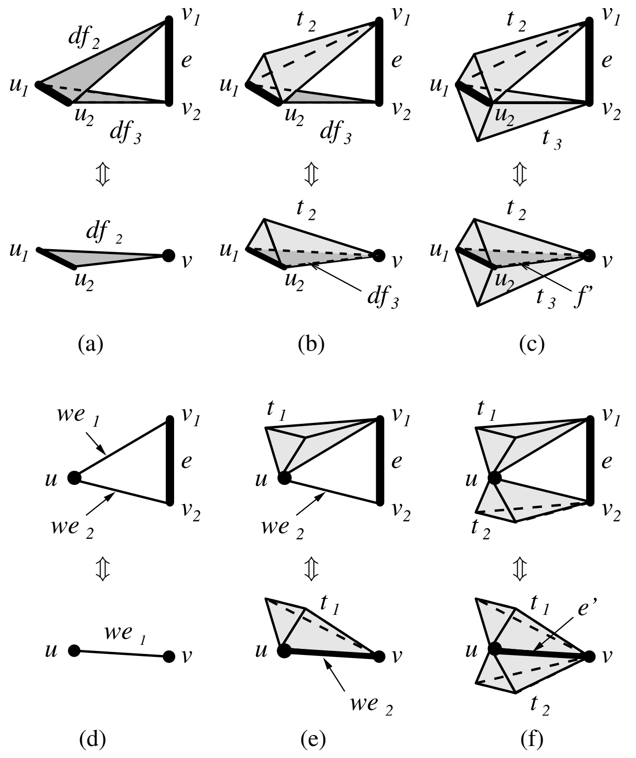

Figure 6: *Effect of edge collapse on the six examples shown in Figure 3. Edge $e = \{v_1, v_2\}$ is collapsed into a new vertex v. In (a), dangling faces df_2 and df_3, which originally intersect at edge $\{u_1, u_2\}$, become one face df_2. In (b), df_3 becomes an external face of tetrahedron t_2. In (c), the two tetrahedra t_2 and t_3 become face-adjacent at f'. In (d), wire edges we_1 and we_2 become a single wire edge, we_1. In (e), we_2 becomes a boundary edge of simplexes incident at both u and the new vertex v. In (f), the two set of simplexes that are incident at v_1 and v_2, respectively, become edge-adjacent at edge e' after edge collapse.*

that are not incident at v_1, namely faces f_1 and f_3 with all their boundaries. $st(v_2)$ is composed of tetrahedra t_1, t_3, t_5 and of all their faces. $link(v_2)$ is defined similarly to $link(v_1)$, and consists of faces f_2 and f_4 with all their boundaries. Thus L consists of one edge, namely, edge $\{u_1, u_2\}$.

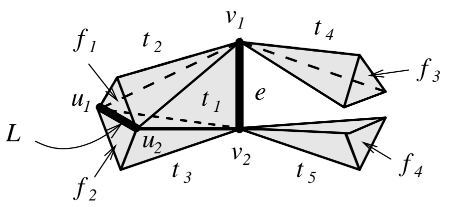

Figure 7: *An illustration for $L = link(v_1) \cap link(v_2)$: $link(v_1)$ is composed of faces f_1, f_3 and of all their boundaries. Similarly, $link(v_2)$ is composed of faces f_2, f_4, and of their boundaries. Therefore, L is composed of edge $\{u_1, u_2\}$. Let $\omega_0 = \{t_2, t_3\}$. Then, $l(\omega_0) = \{t_2, t_3\}$, and $c(\omega_0) = \{t_1\}$.*

Let us consider $\omega_i \in L$: ω_i can be a vertex, u, or an edge $\{u_1, u_2\}$. We denote with $l(\omega_i)$ the set of simplexes incident at ω_i, i.e. in $st(\omega_i)$, and incident at either v_1 or v_2 but not in both. In the example of Figure 7, L has only one component, namely edge $\{u_1, u_2\}$, which we call ω_0. Thus, $l(\omega_0) = \{t_2, t_3\}$. We denote with $c(\omega_i)$ the set of simplexes incident at ω_i, i.e., in $st(\omega_i)$, and in both v_1 and v_2. In the example of Figure 7, $c(\omega_0) = \{t_1\}$.

We consider a simplex $\sigma \in c(\omega_i)$, which can be either a tetrahedron, or a dangling face, and two simplexes σ_1 and σ_2, incident at v_1 and v_2, respectively, and belonging to $l(\omega_i)$. Note that σ_1 and σ_2 can be tetrahedra, dangling faces or wire edges. This latter case is possible only when ω_i is a vertex.

The possible combinations of σ_1, σ and σ_2 are reported in Table 1. The first eight cases apply when ω_i is an edge. The other eight cases apply when ω_i is a vertex. In the columns corresponding to σ_1, σ and σ_2, the symbol T, F, E or $\emptyset$, means that the corresponding simplex is a tetrahedron, a dangling face, a wire edge or is non-existent, respectively. When ω_i is an edge, σ is always bounded by four vertices $\{v_1, v_2, \omega_i\}$ and can be either a tetrahedron or a hole. In both cases, it shares face $\{v_1, \omega_i\}$ with σ_1 and face $\{v_2, \omega_i\}$ with σ_2. If σ is a tetrahedron, it undergoes a dimension-reduction transformation into face $\{v, \omega_i\}$. In all cases, simplexes $\{v_1, \omega_i\}$ and $\{v_2, \omega_i\}$ are merged into simplex $\{v, \omega_i\}$. In the NMIA data structure, we are only interested in the case in which $\{v, \omega_i\}$ is a dangling face, since we do not encode the faces bounding a tetrahedron explicitly.

When ω_i is a vertex, σ can be either a dangling face or a hole, but, in both cases, it is bounded by three vertices $\{v_1, v_2, \omega_i\}$. Simplexes σ_1 and σ_2 share edges $\{v_1, \omega_i\}$ and $\{v_2, \omega_i\}$ with σ, respectively. Note that, in general, there are several σ_1 and σ_2 in $l(\omega_i)$, except when σ_1 or σ_2 are wire edges: in this case, they are completely defined by $\{v_1, \omega_i\}$ or $\{v_2, \omega_i\}$. Moreover, either σ_1 or σ_2 can be a wire edge only if σ is empty.

ω_i is an edge $\{u_1, u_2\}$				ω_i is a vertex u			
Case	σ_1	σ	σ_2	Case	σ_1	σ	σ_2
1	$\emptyset$	T	$\emptyset$	9	$\emptyset$	F	$\emptyset$
2	T	T	$\emptyset$	10	T/F	F	$\emptyset$
3	$\emptyset$	T	T	11	$\emptyset$	F	T/F
4	T	T	T	12	T/F	F	T/F
5	F	$\emptyset$	F	13	E	$\emptyset$	E
6	T	$\emptyset$	F	14	T/F	$\emptyset$	E
7	F	$\emptyset$	T	15	E	$\emptyset$	T/F
8	T	$\emptyset$	T	16	T/F	$\emptyset$	T/F

Table 1: *Cases for simplex σ in $c(\omega_i)$ and simplexes σ_1 and σ_2 in $l(\omega_i)$.* T = *tetrahedron,* F = *dangling face,* E = *wire edge,* $\emptyset$ *stands for the absence of a simplex.*

Examples of cases 1, 2, 4, 9, 10 and 12 in Table 1 are shown in Figures 2(a) to 2(f). The examples in Figures 3(a) to 3(f) are instances of cases 5, 6, 8, 13, 14 and 16, respectively. Cases 3, 7, 11 and 15 are symmetric to cases 2, 6, 10 and 14 respectively.

In what follows, we examine how the entities and the relations are affected in the sixteen cases shown in Table 1. We consider L as an ordered sequence of elements ω_i, where ω_i is either a vertex or an edge. We denote as ω_{i-1} and ω_{i+1}, respectively, the predecessor and the successor of ω_i along L. Note that ω_i does not need to be connected to either ω_{i-1} or ω_{i+1}. Also, ω_i can be an isolated vertex, or a vertex which is common to the two edges ω_{i-1} and ω_{i+1}, or an extreme vertex of either ω_{i-1} or ω_{i+1}. These two latter cases occur when σ is a dangling face $\{u, v_1, v_2\}$ and $\{\omega_{i-1}, v_1, v_2\}$ or $\{\omega_{i+1}, v_1, v_2\}$, or both, define an empty tetrahedral hole. Thus, similarly to isolated vertices, we consider such a vertex as a separate element of L and not as an extreme vertex of an edge.

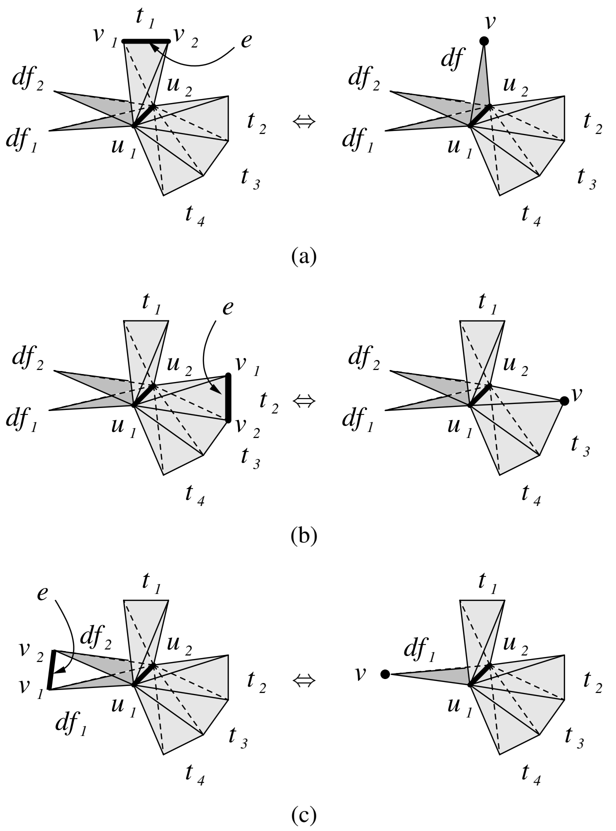

Figure 8: *Examples of the updates needed at edge $\{u_1, u_2\}$ after the collapse of edge e: In (a), tetrahedron t_1 becomes a dangling face df. $R_{2,cluster}(df)$ at edge $\{u_1, u_2\}$ consists of dangling face df_2 and tetrahedron t_2. The adjacency relations of the original neighbors of t_1 at edge $\{u_1, u_2\}$, namely df_2 and t_2, need to be updated. df replaces t_1 in $R_{2,cluster}(df_2)$ and $R_{3,cluster}(t_2)$. In (b), after edge e is collapsed, tetrahedron t_2 becomes a boundary face of tetrahedron t_3, and tetrahedra t_1 and t_3 become immediate neighbors of each other at edge $\{u_1, u_2\}$. Therefore, t_2 is removed from $R_{33}(t_3)$ relation and t_1 is added to $R_{3,cluster}(t_3)$. t_3 replaces t_2 in $R_{3,cluster}(t_1)$. In (c), dangling face df_2 is merged into dangling face df_1. df_1 and tetrahedron t_4 become immediate neighbors of each other. So t_4 replaces df_2 in $R_{2,cluster}(df_1)$, and df_1 replaces df_2 in $R_{3,cluster}(t_4)$.*

When $\omega_i = \{u_1, u_2\}$ is an edge, (which holds for cases 1 to 8 in Table 1,) the collapse of edge $e = \{v_1, v_2\}$ may cause the 1- and 2-adjacency (i.e., $R_{2,clusters}$, $R_{3,clusters}$ and R_{33}) relations to change for simplexes incident at edge $\{u_1, u_2\}$. These relations need to be updated. The three examples in Figure 8 illustrate how the update is done at $\{u_1, u_2\}$. In each example, the left part illustrates the situation before edge collapse, and the right part after. After edge collapse, adjacency relations ($R_{3,clusters}$ or $R_{2,clusters}$) at the new edges $\{u_1, v\}$ and $\{u_2, v\}$ are updated in a similar fashion. Special attention has to be given, however, when $\omega_i = \{u_1, u_2\}$ is connected to ω_{i-1} or to ω_{i+1} because neighboring simplexes may also be merged due to the collapse of edge e. Figure 9(a) illustrates a situation in which ω_i is not connected to its predessesor or successor, and Figure 9(b), a situation in which ω_i is connected to its predessesor. In Figure 9(a), tetrahedron t_1 is incident at the edge $e = \{v_1, v_2\}$ to be collapsed. Let ω_0 be edge $\{u_1, u_2\}$. ω_0 is not connected to any other components of L since it is the only component of L. Dangling face df_1 is the immediate neighbor of t_1 at edge $\{u_1, v_1\}$, and dangling face df_2 is the immediate neighbor of t_1 at edge $\{u_1, v_2\}$. After edge collapse, df_1 and df_2 become 1-adjacent at the new edge, $\{u_1, v\}$. In Figure 9(b), the edge to be collapsed is $e = \{v_1, v_2\}$. L has two components, namely $\{u_3, u_1\}$ and $\{u_1, u_2\}$, which we call ω_0 and ω_1. Similar to the example of 9(a), before edge collapse, dangling faces df_1 and df_2 are neighbors of t_1 at edges $\{u_1, v_1\}$ and $\{u_1, v_2\}$, respectively. After edge collapse, df_2 is merged to df_1.

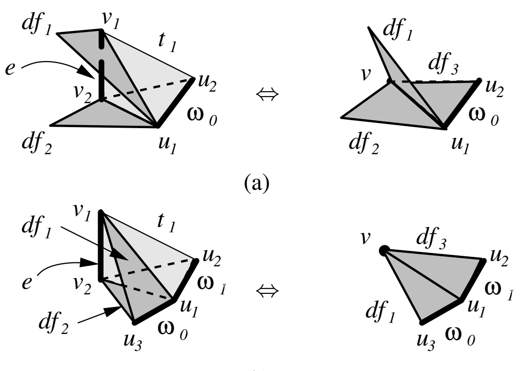

Figure 9: *Two examples to show the update of the edge-based clusters at edge $\{u_1, v\}$ after an edge collapse operation. In (a), tetrahedron t_1 becomes a dangling face df_3. Dangling faces df_1 and df_2, which are neighbors of t_1 at edge $\{u_1, v_1\}$ and edge $\{u_1, v_2\}$, respectively, become neighbors of each other at the new edge $\{u_1, v\}$ Therefore, $R_{2,clusters}(df_3)$ consists of df_1 and df_2. df_3 replaces t_1 in both $R_{2,clusters}(df_1)$ and $R_{2,clusters}(df_2)$. In (b), after edge collapse, tetrahedron t_1 becomes dangling face df_3, and dangling face df_1 is merged into dangling face df_2. So $R_{2,clusters}(df_3)$ consists of only df_1 and $R_{2,clusters}(df_1)$ consists of only df_3.*

When $\omega_i = u$ is a vertex, (cases 9 to 16 in Table 1,) the collapse of edge $e = \{v_1, v_2\}$ causes two 1-connect clusters to merge into one. $R_{0,clusters}$ at the new vertex v needs to be defined. We need to update $R_{3,clusters}$ or $R_{2,clusters}$ relations at edge $\{u, v\}$. These updates are completely similar to those done for simplexes incident at edge $\{u_1, v\}$ and $\{u_2, v\}$ for the case where ω_i is an edge.

We summarize now the various cases in an edge collapse algorithm. Let Σ be the given complex. Let $e = \{v_1, v_2\}$ be the edge to be collapsed into a vertex v. Recall that $L = link(v_1) \cap link(v_2)$. We denote with Ω_{L1} the set of top simplexes (tetrahedra, dangling faces and wire edge) σ such that σ is incident at v_1 and in some entity of $link(v_1) - L$, and with Ω_{L2} the set of simplexes incident at v_2 and in some entity of $link(v_2) - L$.

The *Edge Collapse algorithm* performs the following steps:

Step 1: Compute L, Ω_{L1}, Ω_{L2}, and, for each $\omega_i \in L$, $l(\omega_i)$ and $c(\omega_i)$ as follows:

1. Compute $st(v_1)$ and $st(v_2)$ by using $R_{0,clusters}(v_1)$ and $R_{0,clusters}(v_2)$, respectively.
2. Compute $link(v_1)$ and $link(v_2)$ from the boundary relations of the entities in $st(v_1)$ and $st(v_2)$.
3. Compute $L = link(v_1) \cap link(v_2)$. If a face exists in L, the edge collapse operation is invalid.
4. For each $\omega_i \in L$, compute $l(\omega_i)$ and $c(\omega_i)$ by using $R_{0,clusters}$, $R_{2,clusters}$ and $R_{3,clusters}$ relations.
5. Compute Ω_{L1}, Ω_{L2} from L, $st(v_1)$ and $st(v_2)$.

Step 2: For each simplex $\omega_i \in L$:
If $\omega_i = \{u_1, u_2\}$ is an edge,

1. Perform validity check to ensure that the region $\{v_1, v_2, u_1, u_2\}$ is either a tetrahedron or is empty.
2. **Case 1:** mark σ as deleted, and create a new dangling face σ' in Σ.
   Cases 2, 3 and 4: mark σ as deleted.
   Cases 5 and 6: mark σ_2 as deleted.
   Case 7: mark σ_1 as deleted.

If $\omega_i = u$ is a vertex,

1. Perform validity check to ensure that the region $\{v_1, v_2, u\}$ is either a dangling face or is empty.
2. **Cases 9 to 12:** mark σ as deleted.
   Case 9: create new a wire edge σ' in Σ,
   Cases 13 and 14: mark σ_2 as deleted.
   Case 15: mark σ_1 as deleted.

Step 3: For each simplex $\omega_i \in L$:
Update the relations affected at the neighborhood of ω_i, as described before.

Step 4: For each simplex $\sigma' \in \Omega_{L1}$, Update $R_{k0}(\sigma')$ by replacing v_1 with v.

Step 5: For each simplex $\sigma' \in \Omega_{L2}$, Update $R_{k0}(\sigma')$ by replacing v_2 with v.

Step 6: Update $R_{0,clusters}(v)$. Delete all the marked simplexes.

7. Performing a Vertex Split

7.1. An Encoding Scheme for Vertex Split

In this Subsection, we describe a compact encoding scheme for a vertex split. The encoding scheme is composed of two parts: a labeling of the entities in the restricted star of v and an encoding of $L = link(v_1) \cap link(v_2)$ together with the cases discussed in Section 6. The labeling of the entities in the star of v allows us to modify all the simplexes which become incident at v_1 or v_2 to generate the new k-simplexes obtained by expanding $(k-1)$-simplexes and to duplicate simplexes. The encoding of L is necessary for encoding boundary faces and edges which are duplicated and expanded (since they are not described in the NMIA data structure), and for updating topological relations locally.

For every k-simplex σ' in $st(v)$, we store a 2-bit code, $c_1(\sigma')$ for detecting whether σ' after the split becomes incident at v_1 ($c_1(\sigma') = 00$) or at v_2 ($c_1(\sigma') = 01$) or it is duplicated into two k-simplexes incident at v_1 and at v_2 respectively ($c_1(\sigma') = 10$), or it is expanded into a $(k+1)$-simplex incident at edge e ($c_1(\sigma') = 11$). A unique traversal of $st(v)$ is defined by following the order in which the representative simplexes are encoded in the $R_{0,clusters}$ relation and a predefined traversal inside each vertex-based cluster.

For each element $\omega_i \in L$, we store:

- a 4-bit code $c_2(\omega_i)$, which encodes the sixteen cases shown in Table 1;
- 1-bit code $c_3(\omega_i)$ which indicates whether ω_i is connected to ω_{i-1} through vertex u. If ω_i is connected to ω_{i-1}, then vertex u is not encoded;
- one or two indexes of the vertices which define ω_i (only one vertex is encoded when ω_i is a vertex or is connected to ω_{i-1});
- other information which depend on the specific case according to Table 1:

  - **Case 1:** index of the dangling face $\{u_1, u_2, v\}$, which becomes a tetrahedron $\{u_1, u_2, v_1, v_2\}$;
  - **Cases 2, 4, 6 and 8:** index of σ_1;
  - **Cases 3 and 7:** index of σ_2;
  - **Case 5:** index of the dangling face $\{u_1, u_2, v\}$, which becomes two faces $\{u_1, u_2, v_1\}$ and $\{u_1, u_2, v_2\}$;
  - **Case 9:** index of wire edge $\{u, v\}$, which becomes a dangling face $\{u, v_1, v_2\}$;
  - **Cases 10 and 14:** index of σ_1;
  - **Cases 11 and 15:** index of σ_2;
  - **Cases 12 and 16:** index of σ_1 and σ_2.
  - **Case 13:** index of wire edge $\{u, v\}$, which becomes two wire edges $\{u, v_1\}$ and $\{u, v_2\}$;

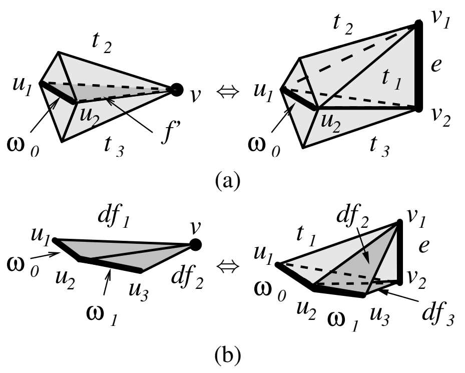

Figure 10: *Two examples of encoding a vertex split: In (a), the encoding of L is* [4 0 u_1 u_2 t_2]. *In (b), the encoding of L is* [1 0 u_1 u_2 df_1; 5 1 u_3 df_2].

Figure 10 shows two examples of the encoding scheme. In the example of Figure 10(a), L consists of just one element, namely $\{u_1, u_2\}$, which we call ω_0. The encoding of L relative to ω_0 is [4 0 u_1 u_2 t_2]. The first field is code c_2, meaning that we are in case 4. The second field indicates that the first vertex of ω_0 is not connected to the last vertex of ω_0. So, both vertices u_1 and u_2 are found in the next field, which is followed by tetrahedron t_2 incident at v_1. During vertex split, t_3 is retrieved from $R_{33}(t_2)$ relation, and, a new tetrahedron t_1 is created, which is incident at the new edge e.

In Figure 10(b), L consists of two elments, $\{u_1, u_2\}$ and $\{u_2, u_3\}$, that we call ω_0 and ω_1, respectively. The encoding of L is [1 0 u_1 u_2 df_1; 5 1 u_3 df_2]. ω_0 is in case 1. The value 0 in the next field indicates that the first vertex of ω_0 is not connected to the last vertex of ω_1. The simplex incident at ω_0 is df_1. During vertex split, df_1 is expanded into a tetrahedron incident at both v_1 and v_2. ω_1 is in case 5. The value 1 in the next field means that first vertex of ω_1 is the same as the last vertex of ω_0. The second vertex of ω_1 is u_3, which is given in the third field. The last field gives the simplex, df_2, which is incident at ω_1. During vertex split, df_2 becomes incident at v_1, and a new dangling face incident at v_2 is created.

7.2. An Algorithm for Performing Vertex Split

The algorithm for performing vertex split uses the encoding of the vertex split operation described in Section 7.1 and updates the simplicial complex Σ on which the vertex split is applied through the following steps:

Step 1: Compute $st(v)$ by retrieving $R_{01}(v)$ for wire edges incident at v, $R_{02}(v)$ for dangling faces incident at v and $R_{03}(v)$ relations from $R_{0,clusters}(v)$.

Step 2: For each $\sigma' \in st(v)$,

- **If** $c_1(\sigma') = 00$ (σ' becomes incident at v_1),
  then update $R_{k0}(\sigma'), k = 2$ or 3 by replacing v with v_1.
- **If** $c_1(\sigma') = 01$ (σ' becomes incident at v_2),
  then update $R_{k0}(\sigma'), k = 2$ or 3 by replacing v with v_2.
- **If** $c_1(\sigma') = 10$ (σ' is duplicated into two simplexes),
  then replace k-simplex $\sigma', k = 1$ or 2, with two new k-simplexes σ_a and σ_b such that $R_{k0}(\sigma_a)$ is obtained from $R_{k0}(\sigma')$ by replacing v with v_1 and $R_{k0}(\sigma_b)$ is obtained from $R_{k0}(\sigma')$ by replacing v with v_2.
- **If** $c_1(\sigma') = 11$ (σ' is expanded into a $(k+1)$-simplex),
  then replace k-simplex $\sigma', k = 1$ or 2, with a new $(k+1)$-simplex σ such that $R_{k0}(\sigma)$ is obtained from $R_{k0}(\sigma')$ by replacing v with $\{v_1, v_2\}$.

Step 3: For each $\omega_i \in L$:

Case 2: (σ_1 is already incident at v_1) A new tetrahedron $\sigma = \{u_1, u_2, v_1, v_2\}$ is created which shares face $\{u_1, u_2, v_1\}$ with σ_1.
Case 3: (σ_2 is already incident at v_2) A new tetrahedron $\sigma = \{u_1, u_2, v_1, v_2\}$ is created which shares face $\{u_1, u_2, v_2\}$ with σ_2.
Case 4: (σ_1 is already incident at v_1 and σ_2 is already incident at v_1) A new tetrahedron σ is created which shares faces $\{u_1, u_2, v_1\}$ and $\{u_1, u_2, v_2\}$ with σ_1 and σ_2, respectively.
Case 6: (σ_1 is already incident at v_1) A new dangling face $\sigma_2 = \{u_1, u_2, v_2\}$ is created.
Case 7: (σ_2 is already incident at v_2) A new dangling face $\sigma_1 = \{u_1, u_2, v_1\}$ is created.
Cases 10, 11 and 12: A new dangling face $\sigma = \{u, v_1, v_2\}$ is created.
Case 14: (σ_1 is already incident at v_1) A new wire edge $\sigma_2 = \{u, v_2\}$ is created.
Case 15: (σ_2 is already incident at v_2) A new wire edge $\sigma_1 = \{u, v_1\}$ is created.
All other cases: nothing to be done.

In all cases, define R_{k0} for each newly created k-simplex.

Step 4: Define adjacency and incidence relations for each newly created entity. Update relations for each simplex σ' incident at an element $\omega_i \in L$ and at v_1 or at v_2, and update relations for each neighbor of σ'. This step reverses the modifications to the adjacency and incidence relations encoded in the NMIA data structure performed in edge collapse (see Section 6).

Step 5: Compute $R_{0,clusters}(v_1)$ and $R_{0,clusters}(v_2)$.

8. A Non-Manifold Multi-Tessellation for 3D simplicial complexes

The basic ingredients in a *Non-Manifold Multi-Tessellation (NMT)* are modifications and a dependency relation among modifications. A *modification* of a complex Σ is an operation that replaces a set of simplexes from Σ with another set of simplexes, under the constraint that the result is still a simplicial complex [5]. A *dependency relation* among refinement modifications is defined as follows: a modification M depends on another modification M' if M deletes some simplexes introduced by M'. The transitive closure of the dependency relation defines a partial order among a set of refinement modifications applied to the complex at coarsest resolution, called the *base complex*.

A *Non-manifold Multi-Tessellation (NMT)* is a partially ordered set of *nodes* $\{\mathcal{M}\} = (\{M_0, M_1, \ldots, M_h\}, \prec)$, where each node M_i, $i = 1, 2, .., h$ represents both a refinement modification and its inverse coarsening modification and node M_0 is the base complex [5]. A subset S of the nodes of an NMT is called *closed* with respect to the partial order if and only if, for each node $M_j \in S$, all nodes M_i, such that M_i precedes M_j, are also in S. The modifications corresponding to a closed subset of nodes can be applied to the base complex Σ_0 in any total order extending the partial order. This produces an *extracted mesh* Σ_S at a level of resolution intermediate between the base complex and the complex at full resolution.

Here, we are interested in an NMT based on three-dimensional simplicial complexes and built through the iterative application of the edge collapse operation. Generating an NMT requires developing a simplification algorithm which applies edge collapse iteratively by starting from the most refined representation. The simplification algorithm requires a cost function to decide the order in which the edges must be collapsed [16]. The design and implementation of a simplification algorithm based on edge collapse is one of the objectives of our future work. The data structure we plan to develop for a 3D NMT is described below.

The root node of the NMT, M_0, is encoded through the NMIA data structure representing the base complex Σ_0, while each other node encodes all information needed to perform edge collapse and vertex split, The edge $e = \{v_1, v_2\}$ to be collapsed and the vertex v into which e collapses are sufficient to perform edge collapse, while performing a vertex split requires specifying how the simplexes in the star of v are modified by the vertex split operation, as described in Subsection 7.1. The dependency relation can be implicitly described as a binary tree which encodes only the dependencies between vertex v and the two extreme vertices of edge $e = \{v_1, v_2\}$ and by a vertex enumeration mechanism proposed by El-Sana and Varshney [13] for triangle meshes simplified through vertex-pair contraction, which we have used in the implementation of the 2D NMT [5].

The basis of any query on a multi-resolution model is *selective refinement*, which consists of extracting a complex, that satisfies some application-dependent requirements, such as approximating a spatial object with a certain accuracy which can be either uniform, or variable in space. The solution of a selective refinement query is the extracted complex Σ_S of minimum size associated with a closed set S of modifications applied to the base complex Σ_0. Selective refinement is performed by traversing the NMT and constructing a closed subset S of nodes, and its associated mesh Σ_S either by recursive top-down refinement applied to the base complex or by an incremental fashion, which finds a solution to a new query by applying refinement and coarsening modifications to the complex obtained as a solution to a previous query. The extracted complex is described through the NMIA data structure and the modifications are performed on the NMIA structure by applying the vertex split and edge collapse algorithms described in Sections 7 and 6.

9. Concluding Remarks

We have considered the problem of representing and updating a decomposition of a non-manifold object into a 3D simplicial complex as support to progressive and multi-resolution object representations.

We have described a compact and scalable implementation of the NMIA data structure, which exhibits a overhead of just one byte per vertex when applied to manifold objects. The implementation presented in [4] costs $1/3$ more when applied to manifolds. Moreover, the NMIA data structure requires $1/3$ of the space with respect to the incidence graph and about $2/3$ of the space required by a simplified version of the incidence graph discussed in [4]. We have also compared the NMIA data structure with a general, dimension-independent, data structure based on the decomposition of a d-dimensional simplicial complex into nearly manifold components, called *IQM data structure* [7]. The 3D instance of the IQM data structure requires about two more integers per manifold vertex and four more integers per non-manifold vertex with respect to the NMIA data structure. This is due to the fact that the IQM data structure explicitly encodes the object decomposition.

We have discussed the effect of edge collapse and vertex split operations in a 3D simplicial complex in a completely general setting, which allows changing the topological type of the complex. We have specified such operations based on the entities and relations stored in the NMIA data structure, and we have proposed an encoding for vertex splits to be used for progressive as well as for multi-resolution volumetric representations of non-manifold objects. The compactness of the NMIA data structure makes the updating task on such data structure more complex than on a complete, but verbose, representation such as the incidence graph.

We have implemented the NMIA data structure, an algorithm for constructing it from the collection of top simplexes describing the complex, navigation algorithms to retrieve adjacency and incidence relations efficiently, and the algorithms for performing the edge collapse and vertex split as described in this paper.

Figure 11 shows some experimental results we have obtained by applying edge collapse to a simple non-manifold and non-regular model of a wind-chime described by a 3D simplicial complex consisting of 360 tetrahedra, 9 dangling faces, and 7 wire edges, (see Figure 11(a)). We show in Figure 11(b-d) a progressively simplified sequence after 54, 114 and 138 edge collapse operations.

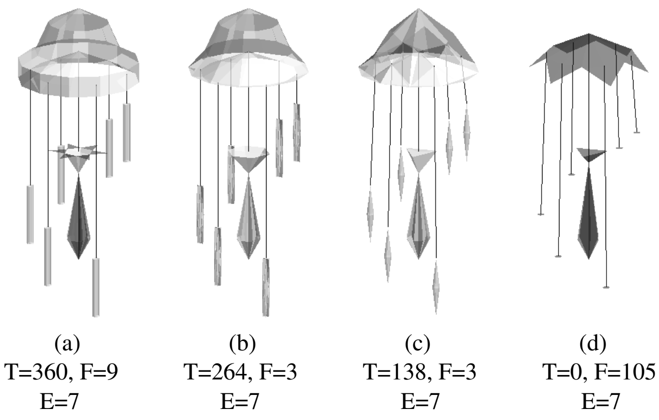

Figure 11: *Simplication of a wind-chime: (a) is the original object. (b) to (d) show the results after 54, 114 and 138 edge collapse operations, respectively.* T, F and E are the numbers of tetrahedra, dangling faces and wire edges in the model.

In our future work, we plan to design and develop a simplification algorithm for objects described through a 3D simplicial complex, based on the technique for edge collapse described here. To this aim, we need to design and implement suitable cost functions to define an order in which the collapses must be performed. Moreover, we plan to develop a data structure for the 3D NMT, and to implement selective refinement algorithms based on a representation of the extracted mesh as an NMIA data structure and on the techniques for performing vertex split and edge collapse described here.

10. Acknowledgments

This work has been partially supported by project MACROGeo, funded by the Italian Ministry of Education, University and Research under contract number RBAU01MZJ5.

References

[1] E. Brisson. Representing geometric structures in d dimensions: topology and order. In *Proceedings 5th ACM Symposium on Computational Geometry*, pages 218–227. ACM Press, 1989. 3

[2] P. Cignoni, D. Costanza, C. Montani, C. Rocchini, and R. Scopigno. Simplification of tetrahedral volume data with accurate error evaluation. In *Proceedings IEEE Visualization 2000*, pages 85–92. IEEE Computer Society, 2000. 2, 3

[3] P. Cignoni, L. De Floriani, P. Magillo, E. Puppo, and R. Scopigno. Selective refinement queries for volume visualization of unstructured tetrahedral meshes. *IEEE Transactions on Visualization and Computer Graphics*, 10(1):29–45, January-February 2004. 5

[4] L. De Floriani and A. Hui. A scalable data structure for three-dimensional non-manifold objects. In *Proceedings ACM/Eurographics Symposium on Geometry Processing, Aachen, June 22-25*, pages 73–83, 2003. 1, 11

[5] L. De Floriani, P. Magillo, E. Puppo, and D. Sobrero. A multi-resolution topological representation for non-manifold meshes. *Computer Aided Design Journal*, 36(2):141–159, February 2004. 1, 3, 10

[6] L. De Floriani, M.M. Mesmoudi, F. Morando, and E. Puppo. Non-manifold decomposition in arbitrary dimensions. *CVGIP: Graphical Models*, 65(1/3):2–22, 2003. 3

[7] L. De Floriani, F. Morando, and E.Puppo. Representation of non-manifold objects in arbitrary dimension through decomposition into nearly manifold parts. In *8th ACM Symposium on Solid Modeling and Applications*, pages 103–112. ACM Press, June 2003. 3, 11

[8] L. De Floriani, F. Morando, and E. Puppo. A representation for abstract simplicial complexes: an analysis and a comparison. In *Proceedings Symposium on Discrete Geometry for Computer Imagery*, Napoli (Italy), November 2003. 3

[9] H. Desaulnier and N. Stewart. An extension of manifold boundary representation to r-sets. *ACM Trans. on Graphics*, 11(1):40–60, 1992. 3

[10] T. Dey, H. Edelsbrunner, S. Guha, and D. Nekhayev. Topology preserving edge contraction. *Publications de l' Institut Mathematique (Beograd)*, 60(80), 1999. 1, 2, 3

[11] D. Dobkin and M. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. *Algorithmica*, 5(4):3–32, 1989. 3

[12] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer Verlag, Berlin, 1987. 3

[13] J. El-Sana and A. Varshney. Generalized view-dependent simplification. *Computer Graphics Forum*, 18(3):C83–C94, 1999. 10

[14] B. Falcidieno and O. Ratto. Two-manifold cell-decomposition of R-sets. In A. Kilgour and L. Kjelldahl, editors, *Proceedings Computer Graphics Forum (EUROGRAPHICS '92)*, volume 11, pages 391–404, September 1992. 3

[15] R. Farias, J.B.Mitchell, C. T. Silva, and B.Wylie. Time critical rendering of irregular grids. In *Proceedings Sibgrapi - XIII Brazilian Symposium on Computer Graphics and Image Processing*, pages 243–250, 2000. 3

[16] M. Garland. Multi-resolution modeling: Survey & future opportunities. In *Eurographics '99 – State of the Art Reports*, pages 111–131. Eurographics Association, 1999. 3, 10

[17] M.H. Gross and O.G. Staadt. Progressive tetrahedralizations. In *Proceedings IEEE Visualization'98*, pages 397–402, Research Triangle Park, NC, 1998. IEEE Computer Society. 3

[18] A. Gueziec, G. Taubin, F. Lazarus, and W. Horn. Converting sets of polygons to manifold surfaces by cutting and stitching. In *Conference abstracts and applications: SIGGRAPH 98*, Computer Graphics, pages 245–245. ACM Press, 1998. 3

[19] E. L. Gursoz, Y. Choi, and F. B. Prinz. Vertex-based representation of non-manifold boundaries. In M. J. Wozny, J. U. Turner, and K. Preiss, editors, *Geometric Modeling for Product Engineering*, pages 107–130. Elsevier Science Publishers B.V., North Holland, 1990. 3

[20] K.Weiler. The radial edge data structure: a topological representation for non-manifold geometric boundary modeling. In H.W. McLaughlin J.L. Encarnacao, M.J. Wozny, editor, *Geometric Modeling for CAD Applications*, pages 3–36. Elsevier Science Publishers B.V. (North–Holland), Amsterdam, 1988. 3

[21] S.H. Lee and 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*, pages 159–170. Ann Arbor, Michigan, June 2001. 3

[22] P. Lienhardt. Topological models for boundary representation: a comparison with *n*-dimensional generalized maps. *Computer Aided Design*, 23(1):59–82, 1991. 3

[23] H. Lopes and G. Tavares. Structural operators for modeling 3-manifolds. In *Proceedings Fourth ACM Symposium on Solid Modeling and Applications*, pages 10–18. ACM Press, May 1997. 3

[24] A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-independent modeling with simplicial complexes. *ACM Transactions on Graphics*, 12(1):56–102, January 1993. 3

[25] J. Popovic and H. Hoppe. Progressive simplicial complexes. In *ACM Computer Graphics Proceedings, Annual Conference Series, (SIGGRAPH '97)*, pages 217–224, 1997. 1, 2, 3

[26] J. Rossignac and D. Cardoze. Matchmaker: manifold BReps for non-manifold R-sets. In W. F. Bronsvoort and D. C. Anderson, editors, *Proceedings Fifth Symposium on Solid Modeling and Applications*, pages 31–41. ACM Press, June 9–11 1999. 3

[27] J.R. Rossignac and M.A. O'Connor. SGC: A dimension-independent model for point-sets with internal structures and incomplete boundaries. In M. J. Wozny, J.U. Turner, and K. Preiss, editors, *Geometric Modeling for Product Engineering*, pages 145–180. Elsevier Science Publishers B.V. (North–Holland), Amsterdam, 1990. 3

[28] I.J. Trotts, B. Hamann, and K.I. Joy. Simplification of tetrahedral meshes with error bounds. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):224–237, 1999. 3

[29] P. Véron and J-C. Léon. Using polyhedral models to authomatically sketch idealized geometry for structural analysis. *Engineering with Computers*, 17:373–385, 2001. 3

[30] Y. Yamaguchi and F. Kimura. Non-manifold topology based on coupling entities. *IEEE Computer Graphics and Applications*, 15(1):42–50, January 1995. 3