A dimension-independent simplicial data structure for non-manifold shapes

Annie Hui
Dept of Computer Science,
Univerity of Maryland, College Park, USA
e-mail:huiannie@cs.umd.edu

Leila De Floriani
Dept of Computer Science, University of Genova, Italy
Dept of Computer Science and UMIACS,
University of Maryland, College Park, USA
e-mail: deflo@umiacs.unige.it

April 7, 2006

Abstract

We consider the problem of representing and manipulating non-manifold multi-dimensional shapes, discretized as d-dimensional simplicial Euclidean complexes, for modeling finite element meshes derived from CAD models. We propose a dimension-independent data structure for simplicial complexes, that we call the *Incidence Simplicial (IS)* data structure. The IS data structure is scalable to manifold complexes, and supports efficient traversal and update algorithms for performing topological modifications, such as hole removal or dimension reduction. It has the same expressive power and performances as the incidence graph, commonly used for dimension-independent representation of simplicial and cell complexes, but it is much more compact. We present efficient algorithms for traversing, generating and updating a simplicial complex described as an IS data structure. We compare the IS data structure with dimension-independent and dimension-specific representations for simplicial complexes. Finally, we briefly discuss two applications that the IS data structure supports, namely decomposition of non-manifold objects for effective geometric reasoning, and multi-resolution modeling of non-manifold multi-dimensional shapes.

1 Introduction

Simplicial complexes are commonly used to represent multi-dimensional geometric objects in a variety of application domains, including finite element analysis, solid

modeling, animation, terrain modeling, and visualization of scalar and vector fields, because of their attractive combinatorial properties. One of the driving applications of our work is developing data structures, traversal and manipulation algorithms, for modeling finite element meshes generated from CAD models. When generating a finite element mesh from a polyhedral model to meet simulation requirements, several idealization operations need to be performed on the mesh, such as removal of details, topology modification, e.g. hole removal, or reduction in the dimensionality of some parts, which produce non-manifold geometries [18]. A further objective is the identification of form features, which help in the mesh idealization process. For such application, it is necessary to have a data structure for non-manifold and non-regular simplicial complexes, that supports topological navigation, and mesh update efficiently.

In the literature, both dimension-specific (2D and 3D) and dimension-independent representations have been developed for cell and simplicial complexes. Many such data structures have domains that are limited to manifolds or pseudo-manifolds. A widely used data structure is the incidence graph [15], which has been developed for cell complexes in arbitrary dimensions. However, when restricted to simplicial complexes, it results in a verbose representation, which also does not scale well to the manifold case. Scalability is an important property for data structures for cell and simplicial complexes, since non-manifold objects often present few non-manifold singularities.

Here, we propose a dimension-independent data structure for d-dimensional simplicial complexes, that we call the *Incidence Simplicial (IS) data structure*. The IS data structure encodes all simplexes of a simplicial complex, the relation between a simplex of dimension p and its simplicial faces of dimension $p-1$, but only one index pointing from each simplex σ of dimension p to a simplex of dimension $p+1$ that has σ on its boundary. We show that this is sufficient to retrieve efficiently all topological relations in a simplicial d-complex by exploiting the fact that any simplex has a bounded number of faces. The algorithms for retrieving topological relations are the basic traversal tools for navigating in a complex as required by modeling and update operations. We also show how to generate the IS representation for a complex starting from the simplexes forming it, and how to perform topological modifications on a simplicial complex described as an IS data structure. The basic operator for performing topological modifications, such as hole removal or reduction in the dimensions of a part, is vertex-pair contraction, which consists of contracting a pair of vertices into a single vertex. We describe an efficient algorithm for performing it on the IS data structure.

The IS data structure also scales well to the manifold case, since it requires only a small amount of extra storage compared with an IS data structure specific for representing manifold simplicial complexes. It is more compact than the incidence graph even if it has the same representation power and the same performances. It requires more space than data structures which encode only top simplexes, but has the advantage of representing all simplexes explicitly and uniquely, as required, when we want to attach attributes to them (as, for instance, in finite element mesh analysis). It supports update operations more efficiently.

Thus, novel contributions of this paper are:

- a dimension-independent and scalable data structure, the IS data structure, for representing non-manifold, non-regular, d-dimensional objects described by simplicial complexes;
- efficient navigation algorithms for retrieving topological relations from an IS data structure;
- an algorithm for generating an IS data structure from a simple non-topological representation of the complex;
- an algorithm for efficiently performing vertex-pair contraction on a simplicial complex described as an IS data structure;
- a comparison with dimension-independent data structures for simplicial complexes in arbitrary dimensions and with dimension-specific ones for two- and three-dimensional simplicial complexes.

The remainder of this paper is organized as follows. In Section 2, we summarize some background notions. In Section 3, we review related work. In Section 4, we present the IS data structure, and discuss its implementation, its storage cost and its scalability. In Section 5, we describe an algorithm for generating an IS data structure from a collection of simplexes. In Section 6, we present navigation algorithms for efficiently retrieving topological relations among the simplexes of a complex. In Section 7, we discuss a dimension-independent multi-resolution model which can be built on the IS. In Section 8, we present an algorithm for performing vertex-pair contraction on the IS data structure. In Section 9, we compare the IS data structure with both dimension-independent and dimension-specific data structures for simplicial complexes. In Section 10, we draw some concluding remarks by discussing application of the IS data structure to support decomposition of non-manifold objects for geometric reasoning, and multi-resolution modeling of non-manifold multi-dimensional shapes.

2 Background

In this section, we review some basic notions about Euclidean simplicial complexes in arbitrary dimensions, and about topological relations among the cells of a simplicial complex. We use simplicial complexes as discretization for non-manifold objects. Informally, we consider a *manifold* (with boundary) as a compact and connected subset of the Euclidean space for which the neighborhood of each of its points is homeomorphic to an open ball or to an open half-ball. Objects, that do not fulfill this property at one or more points, are called *non-manifold* objects, and if they also contain parts of different dimensionalities, are called *non-regular*.

A Euclidean simplex σ of dimension d is the convex hull of $d+1$ linearly independent points in the n-dimensional Euclidean space E^n, with $d \leq n$. 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*, and a 3-simplex a *tetrahedron*. d is called the *dimension* of σ and is denoted $dim(\sigma)$. Any Euclidean k-simplex σ', with $k < d$, generated by a set $V_{\sigma'} \subseteq V_\sigma$ of

cardinality $k+1 \leq d$, is called a *k-face* of σ. Where no ambiguity arises, the dimension of σ' can be omitted and σ' is simply called a *face* of σ.

A finite collection Σ of Euclidean simplexes forms a *Euclidean simplicial complex* if and only if (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 maximal dimension d of the simplexes in Σ is called the *order*, or the *dimension* of complex Σ. The *domain*, or *carrier*, of a d-dimensional Euclidean simplicial complex (also called a simplicial d-complex) Σ embedded in E^n, with $d \leq n$, is the subset of E^n defined by the union, as point sets, of all the simplexes in Σ.

The *boundary* of a simplex σ is the set of all faces of σ in Σ, different from σ itself. The *star* of a simplex σ is the set of simplexes in Σ that have σ as a face. Any simplex σ such that the star of σ contains only σ is called a *top* simplex. The *link* of a simplex σ is the set of all the faces of the simplexes in the star of σ which are not incident in σ.

Two distinct simplexes are *incident* if one of them is a face of the other. Two simplexes are *k-adjacent* if they share a k-face. 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 $(\sigma_i)_{i=0}^k$ in a simplicial complex Σ such that two consecutive simplexes σ_{i-1} and σ_i in the sequence are h-adjacent. Two simplexes σ and σ^* are *h-connected* when 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 any two simplexes of Σ^* are h-connected. Any maximal h-connected sub-complex of a complex Σ is called an *h-connected component* of Σ.

A d-complex Σ in which all top simplexes are d-simplexes is called *regular*. A regular $(d-1)$-connected d-complex in which the star of all $(d-1)$-simplexes consists of one or two simplexes is called a *(combinatorial) pseudo-manifold*. Note that a carrier of a pseudo-manifold is not necessarily a manifold object. In what follows, we will call manifold simplicial complexes those pseudo-manifold complexes that have a manifold carrier.

Topological relations provide an effective framework for defining, analyzing and comparing data structures for simplicial and cell complexes. Let Σ be a simplicial d-complex and let $\sigma \in \Sigma$ be a p- simplex, with $0 \leq p \leq d$. For g and q, $0 \leq g, q \leq d$, we define the following *topological relations*:

- For $p > q$, the *boundary relation* $B_{p,q}(\sigma)$ consists of the set of simplexes of order q in the set of faces of σ.
- For $p < g$, the *co-boundary relation* $C_{p,g}(\sigma)$ consists of the set of simplexes of order g in the star of σ.
- For $p > 0$, the *adjacency relation* $A_p(\sigma)$ is the set of p-simplexes in Σ that are $(p-1)$-adjacent to σ.
- The *adjacency relation* $A_0(\sigma)$, where σ is a vertex, consists of the set of vertices σ' such that $\{\sigma, \sigma'\}$ is a 1-simplex of Σ.

3 Related Work

Dimension-independent data structures have been proposed for d-dimensional manifold cell complexes, which include the *cell-tuple* [2], and the *n-G-map* [25]. This latter can describe a sub-class of pseudo-manifolds introduced in [25], but not arbitrary non-manifold complexes. *Selective Geometric Complexes (SGCs)* [35] can describe non-manifold and non-regular objects through cell complexes whose cells can be either open, or not simply connected. In SGCs, cells and their mutual adjacencies are encoded in an incidence graph [15]. The *incidence graph* is a data structure for arbitrary cell complexes, which encodes all the cells of the complex, and a large subset of its boundary and co-boundary relations. Thus, it provides a complete, but verbose, description of the complex.

The *Indexed data structure with Adjacencies (IA)* is a dimension-independent data structure for pseudo-manifold simplicial d-complexes embedded in the d-dimensional Euclidean space [31]. It encodes only the d-simplexes together with boundary relation $B_{d,0}$ and adjacency relation A_d. The Simplified Incidence Graph (SIG) [4] is a specialization of the incidence graph for simplicial complexes, which is more compact than an incidence graph. Update operations, however, are somehow complicated on a SIG. The Incidence Simplicial data structure proposed here not only is more compact than a SIG and, thus, than an incidence graph, but, as we will show, it supports traversal and update operations efficiently.

Several dimension-specific data structures have been designed for manifold two-dimensional cell complexes [1, 30, 20, 27] and some which are specific for triangle meshes, e.g., the Corner Table [34], and the Star Vertex [22] data structures. Data structures for non-manifold, non-regular two-dimensional cell complexes have been proposed for modeling non-manifold solids [21, 37, 38]. The *partial entity structure* [24] is more scalable to the manifold case, but still verbose when applied to simplicial 2-complexes. Dimension-specific data structures have been also proposed for encoding simplicial 2-complexes [3, 28, 4]. The Loop edge-use data structure proposed in [28] is for regular simplicial complexes in which non-manifold singularities occur only at edges. A comparison among such data structures presented in [8] shows that the *Triangle-Segment (TS)* data structure requires almost half of the space with respect to the other two, but it encodes only vertices and top simplexes explicitly.

Two representations have been proposed in the literature for three-dimensional manifold complexes, i.e., the *facet-edge* [14] and the *handle-face* data structures [26]. Both of them describe three-dimensional cells implicitly by encoding the manifold complexes that form their boundary. In [23], a scalable data structure for manifold tetrahedral complexes has been proposed, which extends the Corner Table [34] to the 3D case. In [6], a compact and scalable data structure for arbitrary three-dimensional simplicial complexes embedded in the three-dimensional Euclidean space is proposed, called the *Non-Manifold Indexed data structure with Adjacencies (NMIA)*, which extends the IA data structure to arbitrary complexes.

An alternative approach to the design of non-manifold data structures consists of de-

composing a non-manifold object into simpler and more manageable parts [13, 17, 19, 32, 33]. Such techniques deal with the decomposition of the boundary of a regular object into two-manifold parts, with the exception of [32], which decomposes also non-manifold objects with parts of different dimensionalities. In [10], a sound decomposition for d-dimensional non-manifold and non-regular objects described through simplicial complexes is defined, which is unique and produces a description of a simplicial d-complex as a combination of nearly manifold components. A dimension-independent data structure for simplicial d-complexes has been presented in [11], which is based on encoding the decomposition as a graph and the components with an extension of the IA data structure.

4 The Incidence Simplicial Data Structure (IS)

In this Section, we introduce a new dimension-independent data structure for representing Euclidean simplicial complexes in arbitrary dimensions, called the *Incidence Simplicial (IS)* data structure. The IS data structure has the same representation power as the *Incidence Graph (IG)* [15], a data structure widely used for representing d-dimensional Euclidean cell complexes embedded in n-dimensional Euclidean space (with $d \leq n$).

The IG encodes all simplexes of a given d-dimensional simplicial complex Σ, and boundary relations $B_{p,p-1}(\sigma)$, for each p-simplex σ (with $0 < p \leq d$), and co-boundary relations $C_{p,p+1}(\sigma)$, for each p-simplex σ (with $0 \leq p < d$). The IG is easy to manipulate, but it is a verbose representation. The incidence relation between two simplexes is encoded twice—once as a boundary relation and once as a co-boundary relation. The IG can be simplified in the case of simplicial complexes by observing that the co-boundary relations only need to be stored as partial relations. This because all boundary relations in a simplicial complex involve a constant number of entities (e.g., any simplex has a bounded number of faces of any dimension).

Given a d-dimensional simplicial complex Σ embedded in n-dimensional Euclidean space (with $d \leq n$), the *Incidence Simplicial (IS) data structure* encodes all simplexes of Σ and

- for each p-simplex σ, where $0 < p \leq d$, boundary relation $B_{p,p-1}(\sigma)$,
- for each p-simplex σ, where $0 \leq p < d$, a partial version of partial co-boundary relation $C_{p,p+1}(\sigma)$, denoted as $C^*_{p,p+1}(\sigma)$, that consists of one arbitrarily-selected $(p+1)$-simplex for each connected component of the link of σ.

In the example of Figure 1(a), the link of vertex v (shown in Figure 1(b)) consists of two connected components. There are in total five edges incident at v, of which four come from the same component. Thus, partial co-boundary relation $C^*_{0,1}(v)$ consists of $\{we, e\}$, where e is an edge of triangle df. In the example of Figure 1(c) the link of edge e (shown in Figure 1(d)) is composed of three connected components corresponding to triangle df, and to tetrahedra t_1, t_2 and their faces which are incident at e. Thus, partial

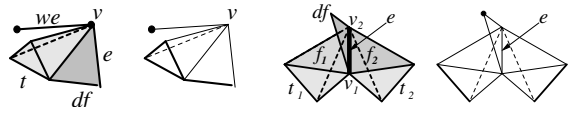


Figure 1: Two examples of 3-complexes: (a) shows an example of a non-manifold vertex v, whose link is shown in thick lines and a vertex in (b); (c) shows an example of a non-manifold edge e; (d) shows the link of e

co-boundary relation $C^*_{1,2}(e)$ consists of just three elements, namely $\{df, f_1, f_2\}$, where f_1 is a face of t_1 and f_2 is a face of t_2.

In general, for each $(d-1)$-simplex σ, partial co-boundary relation $C^*_{d-1,d}(\sigma)$ is the same as co-boundary relation $C_{d-1,d}(\sigma)$. If the domain of Σ is a manifold, then $C^*_{p,p+1}(\sigma)$ contains just one $(p+1)$-simplex since the link of σ consists of one single connected component. Also, when $d = n$, every $(d-1)$-simplex is shared by at most two d-simplexes, since any d-dimensional simplicial complex embedded in the d-dimensional Euclidean space is a pseudo-manifold.

The IS data structure encodes the same boundary relations as the IG. Encoding co-boundary relations as partial relations reduces the storage requirements of the IS data structure with respect to IG (see Section 9.1), especially when the object contains a limited number of non-manifold singularities. Non-manifold singularities are defined by simplexes of dimension lower than the dimension n of the embedding space whose link consists of more than one connected component. Such singularities are made explicit by the encoding of the partial co-boundary relations, in which one simplex for each connected components in the link of each simplex is considered.

In the following, we describe our implementation of the IS data structure, and discuss the storage cost for the IS data structure. Simplexes are stored in ascending order of their dimensions. Each simplex has a unique index and attributes can be associated to it. For each simplex, we also assign a one-bit flag that is used by the navigation algorithms to mark a simplex as visited during traversal, and is reset after traversal is completed.

For each p-simplex σ, with $0 < p \leq d$, boundary relation $B_{p,p-1}(\sigma)$ is stored in a fix-sized array, each element of which is an index to a $(p-1)$-simplex on the boundary of σ.

For each p-simplex σ, with $0 \leq p < d$, we encode partial co-boundary relation $C^*_{p,p+1}(\sigma)$ in a variable-sized array. The first entry of the array stores its length. Each of the remaining entries of the array consists of the index of a $(p+1)$-simplex to which σ is a boundary. Simplex σ has an integer pointer that points to the beginning of this variable-sized array. In the manifold case, relation $C^*_{p,p+1}(\sigma)$ consists of just one element. Thus, the integer pointer for the partial co-boundary relations of σ directly stores the index of the $(p+1)$-simplex in $C^*_{p,p+1}(\sigma)$ relation. A bit-flag is used to indicate whether the manifold condition holds at a p-simplex, when $p < d-1$.

We evaluate the storage cost of the IS data structure assuming that indices and pointers both have the size of an integer. We denote the number of p-simplexes in a simplicial complex Σ as n_p, for $0 \leq p \leq d$. We denote the total number of connected components in the link of a simplex σ as $\kappa(\sigma)$ (when $dim(\sigma) < d$), and the total number of connected components summed over the links of all p-simplexes in Σ as $\kappa_p = \sum_{dim(\sigma)=p} \kappa(\sigma)$, for $0 \leq p < d$.

Each p-simplex has a pointer to a record in which all its attributes are stored. The total number of such pointers is $\sum_{0\leq p\leq d} n_p$ integers. As each p-simplex has $(p+1)$ $(p-1)$-faces, the encoding of boundary relations $B_{p,p-1}$ for $0<p\leq d$ requires $\sum_{0<p\leq d}(p+1)n_p$ integers. The encoding of the partial co-boundary relations requires $\sum_{0\leq p<d}(\kappa_p + 2)$ integers (for the variable-sized arrays and the pointers that points to them) and $\sum_{0\leq p<d-1} n_p$ bits (for the bit flags). Thus, the total storage cost of the IS data structure for a general simplicial d-complex cab be summarized as follows:

- For entities: $\sum_{0\leq p\leq d} n_p$ integers;
- For boundary relations: $\sum_{0<p\leq d}(p+1)n_p$ integers;
- For co-boundary relations: $\sum_{0\leq p<d}(\kappa_p+2)$ integers and $\sum_{0\leq p<d-1} n_p$ bits;

If Σ is a manifold complex, $\kappa(\sigma) = 1$ for $dim(\sigma) < d-1$ and $\kappa(\sigma) \leq 2$ for $dim(\sigma) = d-1$. So $\kappa_p = n_p$ for $p < d-1$. The variable-sized arrays are not needed for $0 \leq p < d-1$ because the pointer to the array can directly store the index of one simplex. For the $(d-1)$-simplexes, fix-sized arrays are used because each $(d-1)$-dimensional simplex can be shared by either one or two d-simplexes. The storage cost of the IS data structure for a manifold d-complex reduces to:

- For entities and boundary relations: no change
- For co-boundary relations: $\sum_{0\leq p<d-1} n_p + 2n_{d-1}$ integers and $\sum_{0\leq p<d-1} n_p$ bits;

The overhead of the IS data structure with respect to a data structure that encodes the same topological relations but specific for a manifold d-complex is thus equal to $\sum_{0\leq p<d-1} n_p$ bits. This is just the cost of encoding the bit flags that indicate whether a simplex depicts a non-manifold singularity. This means a overhead of one byte plus one bit per vertex for manifold 2-complexes, and of four bytes plus four bits per vertex for manifold 3-complexes. Thus, the IS data structure scales very well to the manifold case.

5 Building an Incidence Simplicial Data Structure

The most common exchange format for a simplicial complex consists of a collection of top simplexes described by their vertices. This representation is known as a *soup of top simplexes*. In this section, we discuss how to generate the IS data structure from a soup of top simplexes. As a soup representation does not explicitly describe any simplex that is on the boundary of other simplexes, we need to generate all the simplexes of the input complex first and then establish the topological relations among them. This

is performed in four steps (we perform the computation in decreasing order of simplex dimension):

1. For each p-simplex, we generate its $(p+1)$ $(p-1)$-faces. Each $(p-1)$-simplex is represented by its vertices sorted in lexicographic order.

2. All $(p-1)$-simplexes are sorted by the lexicographic order of their vertices and duplicate simplexes are removed. In this way, each simplex is given a unique index.

3. Boundary relations $B_{p,p-1}$ and complete co-boundary relations $C_{p-1,p}$ for all simplexes are computed as follows. For each p-simplex σ of index i, we simply consider all its $(p-1)$-faces γ. A$(p-1)$-face γ is defined as $V_\sigma - \{u\}$, where V_σ denotes the set of vertices of σ and u the vertex of σ not belonging to γ. We locate the index j of γ by binary search on the lexicographical order of its vertices, and we add γ to $B_{p,p-1}(\sigma)$ and σ to $C_{p-1,p}(\gamma)$.

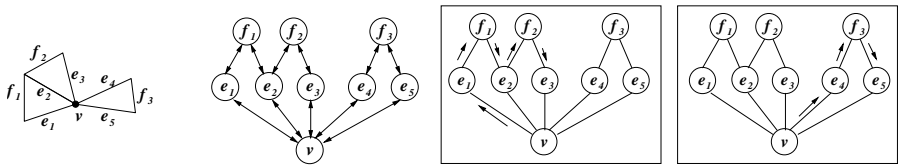
4. For each $(p-1)$-simplex, its partial co-boundary relation $C^*_{p-1,p}$ (with $p < d$) is computed from the corresponding complete relations as described below. Recall that $C^*_{d-1,d}$ is the same as $C_{d-1,d}$

To explain the computation of the partial co-boundary relation of p-simplex σ, let us consider the topological relations among simplexes in the star of σ. Figure 2(a) shows an example of the star of a vertex v. Figure 2(b) illustrates the boundary $B_{p,p-1}$ and co-boundary $C_{p,p+1}$ relations at vertex v through a graph. In this graph, the nodes represent simplexes incident at v and the arcs going downwards depict boundary relations, while those going upwards depict co-boundary relations. This graph depicting the star of v is a subgraph of the complete incidence graph that describes the entire complex.

For each p-simplex σ (with $p < d-1$), we need to identify the $(p+1)$-simplexes that belong to the same connected component of its star. This is performed by a connected component labeling algorithm. The simplest situation arises when a connected component is formed only by one $(p+1)$-simplex τ. In this case, τ will be labeled as belonging one component and added to $C^*_{p,p+1}(\sigma)$. Otherwise, a $(p+1)$-simplex in the star of σ will be on the boundary of $(p+2)$-simplexes belonging to the same connected component. Referring to the graph of Figure 2 we need to traverse the simplexes belonging to levels $(p+1)$ and $(p+2)$ in such graph, by using relations $C_{p,p+1}$, $C_{p+1,p+2}$, and $B_{p+2,p+1}$. We start from an unlabeled $(p+1)$-simplex τ incident at σ, i.e., in $C_{p,p+1}(\sigma)$, and we mark τ with a new label. For each $(p+2)$-simplex θ in $C_{p+1,p+2}(\tau)$, we retrieve the $(p+1)$-simplexes in its boundary, i.e., belonging to $B_{p+2,p+1}(\theta)$. All $(p+1)$-simplexes μ in $B_{p+2,p+1}(\theta)$ that are incident at σ are marked with the same label. This traversal is repeated recursively for all the simplexes in $C_{p+1,p+2}(\mu)$ until all $(p+1)$-simplexes incident at σ and belonging to the same connected component are visited. Then, for each connected component in the star of σ, one $(p+1)$-simplex is selected as an element of $C^*_{p,p+1}(\sigma)$.

As an example, we consider computing $C^*_{0,1}(\sigma)$ in the complex of Figure 2(a) in terms of a graph traversal. Figure 2(c) shows the traversal of one component: starting at vertex v, we move one level up to edge e_1. By visiting all the nodes through the arcs

between levels 1 and 2, we find all the edges, namely e_1, e_2 and e_3, that are in the same connected component. Edge e_1 is then selected as a representative for this component. Figure 2 (d) shows the traversal of the other connected component in $st(v)$. Edge e_4 is selected to represent the same component. The partial $C^*_{0,1}(\sigma)$ relation thus consists of e_1 and e_4.



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

Figure 2: Example of the topological relations of simplexes in the star of a vertex expressed as a graph: (a) The star, $st(v)$, of vertex v; (b) the boundary $B_{1,0}$, $B_{2,1}$ relations and co-boundary $C_{1,0}$, $B_{2,1}$ relations; (c) a traversal of one component in $st(v)$ using relations $C_{0,1}, C_{1,2}$ and $B_{2,1}$ only; (d) a traversal of the other component in $st(v)$

It should be pointed out that it is not necessary to store the complete co-boundary relations of all simplexes throughout the construction process. At any level p, only the boundary and co-boundary relations for levels $(p+1)$ and $(p+2)$ are needed.

Let us now evaluate the time complexity of the various steps in the IS construction algorithm. In step 1, at any level p, the total number of $(p-1)$-simplexes created equals to $(p+1)\cdot n_p$, where n_p is the number of p-simplexes in the whole complex. Thus, creation of $(p-1)$-simplexes take linear time with respec to the number of p-simplexes. In step 2, the time required for sorting all $(p-1)$-simplexes is thus bounded by $O(n_p log(n_p))$. The computation of boundary and co-boundary relations at Step 3 takes linear time with respect to the number of number of p-simplexes, since boundary relations are constant relations.

At step 4, the traversal of the star of a simplex σ, visits every arc and every node of the sub-graph of $st(\sigma)$ once. In the entire complex, each q-simplex is in the stars of $(q+1)$ $(q-1)$-simplexes and in the stars of $\frac{(q+2)\cdot(q+1)}{2}$ $(q-2)$-simplexes. Thus the computation of partial relations for all simplexes in level $(q-2)$ has time complexity of $\frac{(q+2)\cdot(q+1)}{2} n_q + (q+1)n_q$ which is $O(n_q)$.

6 Retrieving Topological Relations

Basic geometric modeling operations, such as Boolean operations, as well as algorithm for manipulating and updating an object described by a simplicial complexes (e.g, simplification algorithms) require being able to extract the simplexes on the boundary of a given simplex, or belonging to the star of a simplex, or those adjacent to a given simplex. This requires efficient algorithms for retrieving the simplexes with are in some topological relation with a given simplex from the data structure encoding a simplicial

complex. The objective is to have algorithms which exhibit a time complexity linear in the number of simplexes in the neighborhood of a given simplex. In this section, we present algorithms for retrieving the simplexes which are in a boundary, co-boundary or adjacency relation with a given simplex.

6.1 Retrieving Boundary Relations

Boundary relations of type $B_{p,p-1}$ are directly encoded, while boundary relation of type $B_{p,q}$, with $q < p$ can be easily retrieved through relations $B_{p,p-1}$, $B_{p-1,p-2}$,$\cdots$, $B_{q+1,q}$. For instance, the vertices of a tetrahedron σ, i.e., in $B_{3,0}(\sigma)$, are retrieved by applying $B_{3,2}(\sigma)$, then $B_{2,1}(\tau)$, for each triangle τ in $B_{3,2}(\sigma)$, and then $B_{1,0}(\gamma)$, for each edge γ in $B_{2,1}(\tau)$. The time complexity of this process is equal to $\Pi_{r=q+1,p+1} r$, where c is a constant. This quantity is bounded by a constant which depends on the dimension p of the simplex and on the dimension q of its faces. For instance, retrieving for $B_{d,0}$, requires $O((d+1)!)$ time.

6.2 Retrieving Co-boundary Relations

Co-boundary relation $C_{d-1,d}(\sigma)$ for any $(d-1)$-simplex σ is directly encoded in the IS data structure, because $C^*_{d-1,d}(\sigma)$ is the same as $C_{d-1,d}(\sigma)$. Since only partial co-boundary relations are encoded in the IS, the challenge is to retrieve all complete co-boundary relations efficiently. Recall that relation $C_{p,q}(\sigma)$ consists of all q-simplexes in the star of σ. We will show that we can retrieve such relations in time linear in the number of top simplexes incident in simplex σ.

Observe that the q-simplexes incident at σ are either top simplexes or faces of top simplexes in the star of σ of dimension greater than q. Thus, in order to compute the general $C_{p,q}(\sigma)$ relation, we need to retrieve the top simplexes of dimensions q and above from each connected component of the star of σ, since all q-simplexes that are faces of higher-dimensional simplexes incident at σ can be only retrieved from boundaries of the top simplexes incident in σ.

To illustrate the algorithm, we consider a subgraph of the incidence graph describing only the star of σ. In such subgraph, the nodes describe the simplexes in the star of σ, the arcs from $(q+1)$-simplexes to q-simplexes describe boundary relation $B_{q+1,q}$, and the arcs from q-simplexes to $(q+1)$-simplexes describe the partial co-boundary relation $C^*_{q,q+1}$. Figure 3(a) shows an example of the star of a vertex v. The arcs representing boundary and partial co-boundary relations encoded in the IS data structure for this complex are shown in Figures 3(b) and (c), respectively. Note that the vertices bounding simplexes in the star of v are not shown for clarity.

The algorithm for retrieving co-boundary relation $C_{p,q}(\sigma)$ for a p-simplex consists of a breadth-first traversal of the incidence subgraph associated with σ, starting at σ, as described in Algorithm 1. For each p-simplex τ in the star of σ, the traversal algorithm visits every simplex θ in its partial co-boundary relation, and visits every

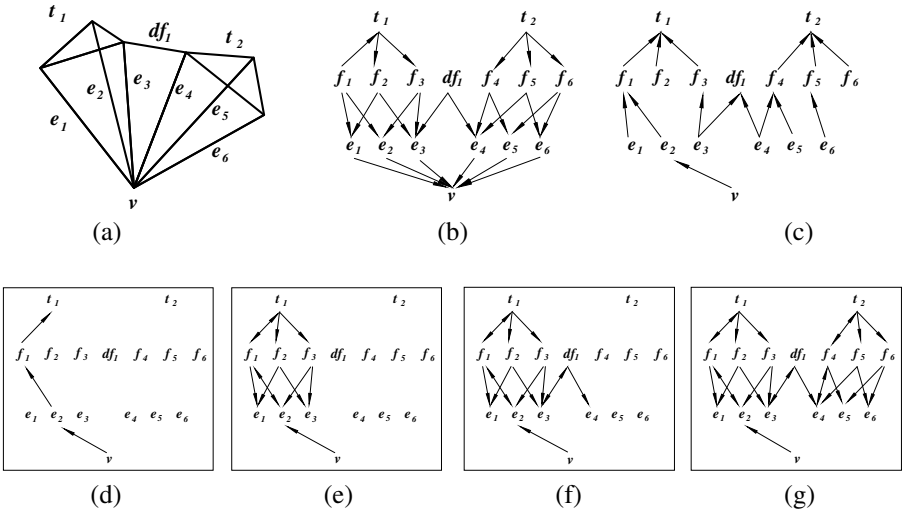


Figure 3: Example of retrieving $C_{0,1}(v)$ through a traversal of the star of vertex v using boundary and partial co-boundary relations: (a) the star $st(v)$ of a vertex v; (b) boundary relations among simplexes in $st(v)$; (c) partial co-boundary relations among simplexes in $st(v)$; (d) to (g): four stages of the traversal of $st(v)$

simplex γ in its boundary relation provided that γ is incident at τ. The traversal of the arcs representing partial co-boundary relations is described in lines 11-18 of Algorithm 1, while the traversal of the arcs representing boundary relations is described in lines 20-29. Note that simplexes of dimension p or lower are not visited because they are not in the star of σ.

Figures 3(d) to (g) show four stages of the traversal of the star of vertex v (from the example of Figure 3(a)) for retrieving $C_{0,1}(v)$. The traversal starts from v and it initialized by using $C^*_{0,1}(v)$. This leads us to visit edge e_2. Through partial co-boundary relations, we get to visit tetrahedron t_1 (as shown in Figure 3(d)). Through the boundary relations, all faces of t_1, and all the edges of the faces f_1, f_2 and f_3 are visited (see Figure 3(e)). The partial co-boundary relation of edge e_3 leads us to visit dangling face df_1. The boundary relation of df_1 leads to edge e_4 (as shown in Figure 3(f)). From e_4, we visit all the edges of tetrahedron t_2 through boundary and partial coboundary relations again (see Figure 3(g)).

In Algorithm 1, a simplex is inserted and deleted from the queue exactly once and each arc in the incidence subgraph associated with a simplex σ is traversed exactly once. Note that the number of arcs is linear in the number of nodes in the incidence subgraph since each simplex is bounded by a constant number of simplexes. Moreover, the total number of simplexes in the star of a simplex is linear in the number of top simplexes in the star. The time complexity of the algorithm for retrieving $C_{p,q}(\sigma)$ is linear with respect to the number of top simplexes in the star of σ.

For simplicial 2-complexes and for simplicial 3-complexes embedded in the three-

dimensional Euclidean space, any $C_{p,q}(\sigma)$ can be retrieved in time linear in the number of q-simplexes in the star of σ. For instance, in a simplicial 3-complex, $C_{0,1}(v)$ for a vertex v can be retrieved in time linear in the number of edges incident at v, since the number of tetrahedra, triangles and edges incident at a vertex v are all linear in each other (from Euler's formula). In a simplicial 4-complex, for instence, $C_{0,1}(v)$ for a vertex v cannot be retrieved in time linear in the number of edges incident at v since the number of 4-simplexes incident at a vertex v can be quadratic in the number of such edges [29]. In general, when $p \leq d-3$, the number of q-simplexes in $C_{p,q}(\sigma)$ is linear with respect to the number of top simplexes, because the link of σ is homeomorphic in this case to a triangulated sphere and thus the vertices, edges and faces in the link are related by Euler's Formula.

Algorithm 1 Coboundary(q, σ)

Require: $q > dim(\sigma)$
1: $S \leftarrow \emptyset$
2: Mark σ as visited
3: Enqueue(Q, σ)
4: **while** not Empty(Q) **do**
5: $\tau \leftarrow$ Dequeue(Q)
6: $r \leftarrow dim(\tau)$
7: **if** $r = q$ **then**
8: Add τ to S
9: **end if**
10: { *Visit co-boundary of* τ }
11: **if** $r < d$ **and** $C^*_{r,r+1}(\tau) \neq \emptyset$ **then**
12: **for** each $\theta \in C^*_{r,r+1}(\tau)$ **do**
13: **if** θ is not visited **then**
14: Mark θ as visited
15: Enqueue(Q, θ)
16: **end if**
17: **end for**
18: **end if**
19: { *Visit boundary of* τ }
20: **if** $r > dim(\sigma)+1$ **then**
21: **for** each $\gamma \in B_{r,r-1}(\tau)$ **do**
22: **if** σ is on the boundary of γ **then**
23: **if** γ is not visited **then**
24: Mark γ as visited
25: Enqueue(Q, γ)
26: **end if**
27: **end if**
28: **end for**
29: **end if**
30: **end while**
31: **return** S

6.3 Retrieving Adjacency Relations

Adjacency relation $A_p(\sigma)$ for a p-simplex σ, when $p > 0$ is retrieved by first extracting the $p+1$ $(p-1)$-faces of σ, and then retrieving, for each such face τ of σ, co-boundary relation $C_{p-1,p}(\tau)$. When $p = 0$, adjacency relation $A_0(v)$ for a vertex v is obtained by first retrieving the set of edges in co-boundary relation $C_{0,1}(v)$, and then retrieving the other extreme vertex of each edge e in $C_{0,1}(v)$ through boundary relation $B_{1,0}(e)$.

For $p > 0$, the running time of the algorithm for retrieving $A_p(\sigma)$ is dominated by the time required to retrieve the co-boundary relations at the $(p-1)$-faces of σ. Thus, it is linear in the total number of top simplexes incident at the $(p-1)$-faces of σ. Similarly, the time complexity of the algorithm for retrieving $A_0(v)$ is linear in the number of top simplexes incident at vertex v.

7 A Multi-resolution Approach to Non-manifold Modeling

In this Section, we discuss a dimension-independent, multi-resolution model for non-manifold shapes described by simplicial complexes. The model provides a compact representation of the topological modifications performed by a simplification algorithm applied to a simplicial complex, organized according to a dependency relation. A virtually continuous set of representations of a shape at different resolutions can be interactively, and dynamically, obtained from such model.

A *Non-Manifold Multi Tessellation (NMT)* is a generalization of the Multi-Tessellation proposed in [12] for manifold simplicial complexes to d-dimensional simplicial complexes. The basic ingredients in an NMT are updates and a dependency relation among updates. An *update* of a complex Σ is an operation that replaces a set of simplexes of Σ with another set of simplexes, under the constraint that the result is still a simplicial complex. Here, we focus on updates that change the size of a mesh by either increasing it (*refinement*), or decreasing it (*coarsening*). The *dependency relation* among refinement updates is defined as follows: an update u depends on another update u' if u deletes some simplex introduced by u'. Under certain assumptions [12], the transitive closure of the dependency relation defines a partial order among a set of refinement updates applied to the complex at coarsest resolution. This latter is called the *base complex*. A *Non-manifold Multi-Tessellation (NMT)* is, thus, defined as the base complex plus a partially ordered set of *updates* $\{\mathscr{U}\} = (\{u_0, u_1, \ldots, u_h\}, \prec)$, where each update u_i, $i = 1, 2, .., h$ represents both a refinement update and its inverse coarsening update.

A subset S of the updates of an NMT is called *closed* with respect to the partial order if, for each update $u_j \in S$, all updates u_i, such that u_i precedes u_j, are also in S. The refinement updates 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*

complex Σ_S at a level of resolution intermediate between the base complex and the complex at full resolution.

Vertex-pair contraction is the basic operation used for modifying the topological type of a mesh, used for closing holes or performing a reduction in dimension for certain parts of a mesh. It is fundamental in the idealization of finite meshes [36], or for performing drastic simplifications of CAD objects consisting of several millions of triangles. Thus, we focus on an NMT built through vertex-pair contraction, that we call a *Vertex-based Non-Manifold Multi-Tessellation (VNMT)*. An update in a VNMT, corresponds to a contraction of a pair of vertices to a new vertex v and to its inverse, vertex split, and it is described procedurally. The procedural representation consists of encoding the two collapsed vertices and the new vertex v, plus a compact encoding of the effect of splitting v on the simplexes in the star of v [7]. The *dependency relation* is encoded as a forest of binary trees of vertices by using a modification of the mechanism proposed in [16] for triangle meshes.

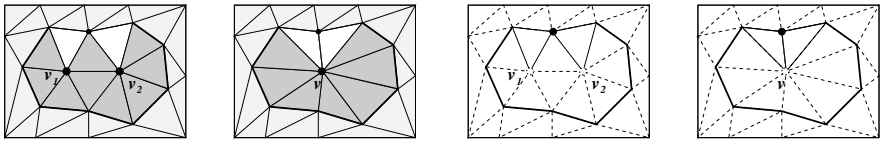
The basis of any query on a multi-resolution model is *selective refinement*. It consists of extracting a complex which satisfies some application-dependent requirements, such as approximating a spatial object with a certain accuracy, which can be either uniform or variable in space. The solution of 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 updates, and its associated mesh Σ_S either by recursive *top-down* refinement applied to the base complex, or by an *incremental* refining and coarsening an already extracted complex.

We encode the extracted complex as an IS data structure, on which vertex-pair contraction and vertex split, which are the basic building blocks of selective refinement algorithms, can be performed efficiently. In the following section, we discuss these building blocks.

8 Update Operations on the IS Data Structure

In this Section, we describe an algorithm for performing vertex-pair contraction on a simplicial complex described as an IS data structure. Vertex-pair contraction, which consists of contracting a pair of vertices to a single one, is the basic operation used for modifying the topological type of a complex, such as closing holes or performing a reduction in dimension for portions of the complex. Thus, it is fundamental in the idealization of finite meshes [36], for performing drastic simplifications of CAD objects consisting of several millions of triangles [16], for building multi-resolution representation for non-manifold objects.

We define first the effect of vertex-pair contraction. To this aim, we introduce some notations. For a vertex w, we denote as $lk(w)$ its link and with $st(w)$ the set of simplexes in its star. Let v_1 and v_2 be two vertices in a simplicial d-complex Σ. A vertex-pair contraction applied to pair v_1 and v_2 consists of contracting vertices v_1 and v_2 to a new



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

Figure 4: In a vertex-pair contraction, vertices v_1 and v_2 become one new vertex v: (a) shows $st(v_1) \cup st(v_2)$ in dark gray; (b) shows $st(v)$ in dark gray which replaces $st(v_1) \cup st(v_2)$ in the complex; (c) shows $lk(v_1) \cup lk(v_2)$ in thick black lines and a vertex. It remains the same as $lk(v)$ which is shown in (d).

vertex v. Thus, all simplexes that are in $st(v_1)$ or in $st(v_2)$ become incident at v. This can be described through a map F which maps simplexes in $st(v_1) \cup st(v_2)$ onto $st(v)$, in such a way that for each simplex $\sigma \in st(v_1) \cup st(v_2)$, $F(\sigma) = \sigma - \{v_1, v_2\} \cup \{v\}$. Note that if a p-simplex also belongs to $st(v_1) \cap st(v_2)$, map F transforms σ into a $(p-1)$-simplex. Figure 4 shows the effect of a vertex-pair contraction. Map F is a surjective function. Its effect can be characterized by *four* possible cases that it produces for a p-simplex σ in the star of the new vertex v:

Case 1: There exists *one* p-simplex σ_1 in the star of v_1 such that $\sigma = F(\sigma_1)$, and σ_1 has an empty intersection with every simplex in the star of v_2 (see Figure 5). In this case, σ is obtained from σ_1 just by replacing v_1 with v.

Case 2: There exists *one* p-simplex σ_2 in the star of v_2 which has an empty intersection with every simplex in the star of v_1, such that $\sigma = F(\sigma_2)$. This case is completely symmetric with respect to case 1.

Case 3: There exist *two* p-simplexes σ_1 and σ_2, belonging to the star of v_1 and of v_2, but not to the intersection of the two stars, such that $\sigma = F(\sigma_1)$ and $\sigma = F(\sigma_2)$ (see Figure 6).

Case 4: There exist *three* simplexes, a $(p+1)$-simplex σ' belonging to the intersection of the stars of v_1 and v_2 and two p-simplexes σ_1, and σ_2, belonging to the stars of v_1 and v_2, respectively, such that $\sigma = F(\sigma')$, $\sigma = F(\sigma_1)$ and $\sigma = F(\sigma_2)$. In this case, σ results from contracting σ' incident at edge $e = \{v_1, v_2\}$, and from transforming σ_1 and σ_2 into σ through map F (see Figure 7).

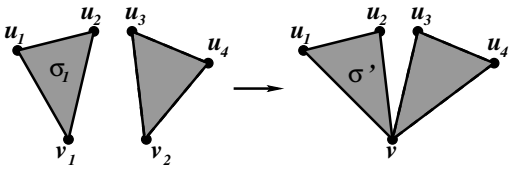


Figure 5: Case 1: simplex σ is obtained from exactly one simplex, σ_1 because σ_1 does not intersect with $st(v_2)$.

The algorithm for performing a vertex-pair contraction first retrieves the stars and the

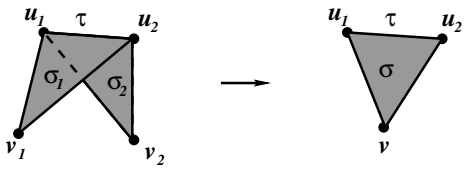


Figure 6: Case 3: simplex σ is obtained from two simplexes, σ_1 and σ_2 because σ_1 and σ_2 intersect at τ. $\sigma = F(\sigma_1) = F(\sigma_2) = \tau - \{v_1, v_2\} \cup \{v\}$.

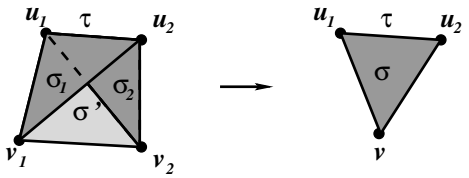


Figure 7: Case 4: simplex σ is obtained from three simplexes, σ', σ_1 and σ_2, the intersection of all three of which is at τ. $\sigma = F(\sigma_1) = F(\sigma_2) = \tau - \{v_1, v_2\} \cup \{v\}$.

links of v_1 and v_2. Next, it computes map F for all the simplexes in the stars of v_1 and v_2, thus defining the simplexes in $st(v)$. Then it updates the boundary and the partial co-boundary relations of the simplexes in $st(v)$ as well as the partial co-boundary relations of the simplexes in $lk(v)$. The link of v is affected since any simplex τ incident in the link of v_1 or v_2 and belonging to the star of v_1 or v_2 has to be replaced with $F(\tau)$ in $lk(v)$. Moreover, connected components formed by simplexes incident in a simplex μ belonging the two links of v_1 and v_2 may merge as an effect of the vertex-pair contraction (such as at vertex u_2 in Figure 8). This also affects the partial co-boundary relations of the simplexes in the link.

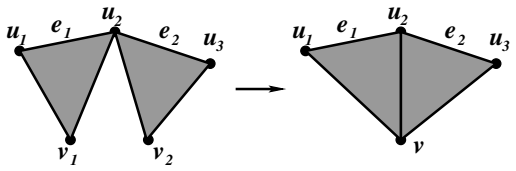


Figure 8: An example showing the effect of vertex-pair contraction on simplexes in the intersection of $lk(v_1)$ and $lk(v_2)$. Vertex u_2 is in $lk(v_1) \cup lk(v_2)$. Contraction of v_1 and v_2 causes two connected components at u_2 to merge into one single component.

A high-level description of the vertex-contraction algorithm is shown in Algorithm 2. Boundary relation $B_{p,p-1}(\gamma)$ for a p-simplex γ in the star of v is computed by considering just one simplex σ such that $\gamma = F(\sigma)$ and applying map F to the $(p-1)$-simplexes belonging to the boundary of σ and incident in v_1 or v_2 (see lines $9-16$ in Algorithm 2).

Complete co-boundary relation $C_{p,p+1}(\sigma)$ are computed for each p-simplex incident

in v_1 or v_2. Given the stars of v_1 and of v_2, complete co-boundary relations for all p-simplexes in each star can be computed based on the boundary relations of all $(p+1)$-simplexes in the same star. This strategy is similar to that used in the construction algorithm in Section 5. Then, co-boundary relation $C_{p,p+1}(\gamma)$ for a p-simplex γ in the star of v is computed by considering the p-simplexes σ incident in v_1 or v_2 such that $\gamma = F(\sigma)$, and applying map F to the $(p+1)$-simplexes θ belonging to the star of simplexes σ (see lines $18-20$ in Algorithm 2). Note that that we do not need to consider any $(p+1)$-simplex incident in both v_1 and v_2 and mapped by F into γ, if there exists one. Then, the partial co-boundary relations of each simplex in $st(v)$ can be found by a traversal of the star of each simplex simply using relations $C_{p,p+1}, C_{p+1,p+2}$ and $B_{p+1,p}$.

Co-boundary relations are updated for the simplexes in the links of v_1 and v_2. For simplexes not in the intersection of $lk(v_1)$ and $lk(v_2)$, update is simply based on the boundary relations of their incident simplexes. But for the simplexes in the intersection of $lk(v_1)$ and $lk(v_2)$, complete co-boundary relations are computed and then the complete co-boundary relations for the simplexes in the link of v are retrieved in a similar way as for the simplexes in the star of v (see lines $24-30$ in Algorithm 2). As mentioned above, we need to retrieve the complete co-boundary relations since connected components formed by simplexes incident in the link may merge as an effect of vertex-pair contraction. Partial co-boundary relations are then obtained from complete ones as in the case of simplexes belonging to the star of v.

Note that the algorithm for the retrieval of the star, $st(\sigma)$, of a simplex σ is very similar to that for retrieving the co-boundary relations of σ. The only difference is that the star of σ consists of every simplex encountered in the traversal while co-boundary relation $C_{p,q}(\sigma)$ consists of only the q-simplexes encountered. Thus, Algorithm **Star(σ)** to retrieve $st(\sigma)$ can be obtained by changing the test condition of line 7 of Algorithm 1 from $(r = q)$ to $(\tau \neq \sigma)$. In a similar manner, $lk(\sigma)$ can be computed by collecting all simplexes visited during the traversal that are not incident at σ. Algorithm **Link(σ)** to compute $lk(\sigma)$ can be obtained by adding an else-statement to Algorithm 1 when the test condition in line 22 fails. Function F can be implemented as a hash table.

Let $\mathscr{S}(\sigma)$ be the size of $st(\sigma)$. The retrieval of the stars and links of v_1 and v_2 is of the order of $\mathscr{S}(v_1)$ and $\mathscr{S}(v_2)$ respectively. The retrieval of the co-boundary relations of simplexes in $lk(v_1) \cap lk(v_2)$ takes linear time with respect to the star of each simplex σ in $lk(v_1) \cap lk(v_2)$. The update of all boundary and co-boundary relations is linear with respect to $\mathscr{S}(v_1) + \mathscr{S}(v_2)$ The complexity of vertex-pair contraction on an IS data structure is thus $\mathscr{S}(v_1) + \mathscr{S}(v_2) + \sum_{\sigma \in lk(v_1) \cap lk(v_2)} \mathscr{S}(\sigma)$.

Some examples of vertex-pair contractions performed on solid models, represented through their triangulated boundaries, are shown in Figures 9.

Vertex Split is the reverse of vertex-pair contraction. In vertex split, a vertex v is split into two new vertices v_1 and v_2. For each p-simplex σ in the star of v, σ may become one of the followings:

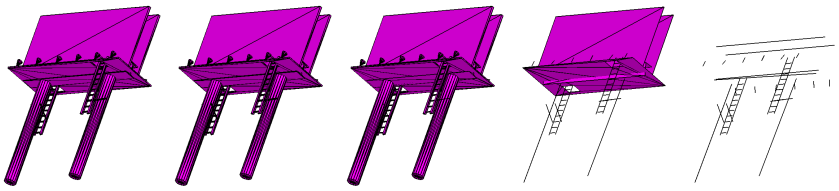
- a new p-simplex incident only at v_1,

- a new p-simplex incident only at v_2,
- two new p-simplexes, one each at v_1 and v_2,
- a new $(p+1)$-simplex incident at v_1 and v_2

A vertex split operation can be fully encoded by encoding the change to simplexes in star of v. The algorithm for performing a vertex split is conceptually the reverse of vertex-pair contraction and is thus not elaborated here.

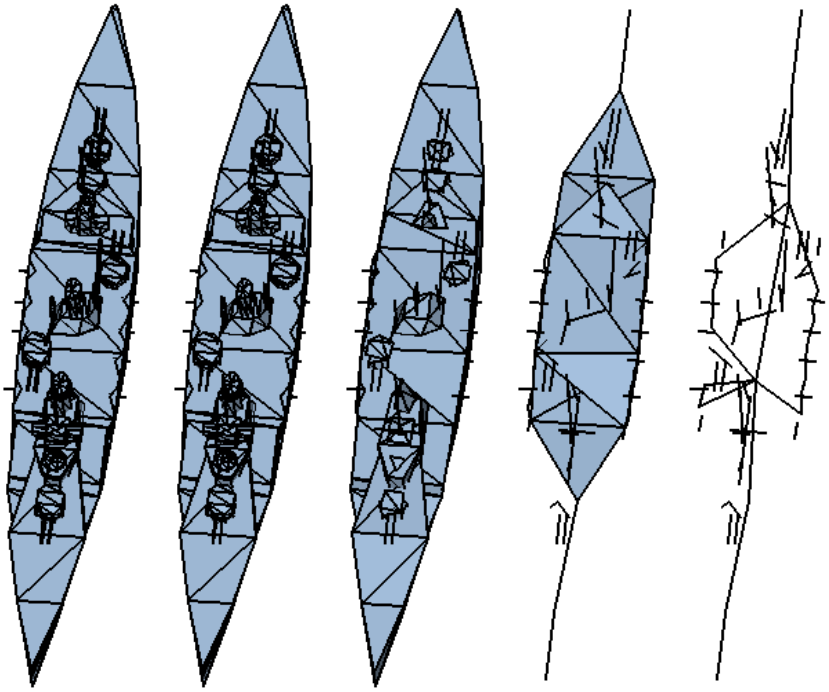
Algorithm 2 VertexPairContract(v_1, v_2)

$S \leftarrow$ **Star**(v_1) $\cup$ **Star**(v_2)
$L \leftarrow$ **Link**(v_1) $\cup$ **Link**(v_2)
Compute and store $F(\sigma)$ for each p-simplex σ in S
Retrieve $C_{p,p+1}(\sigma)$ for each p-simplex σ in S
for each p-simplex σ in S **do**
  $\gamma \leftarrow F(\sigma)$
  if $dim(\gamma) = dim(\sigma)$ **then**
    $\{$*Update boundary relations of* $\gamma\}$
    for each $(p-1)$-simplex τ in $B_{p,p-1}(\sigma)$ **do**
      if τ is in S **then**
        Add $F(\tau)$ to $B_{p,p-1}(\gamma)$
      else
        $\{$ τ *is in* $L\}$
        Add τ to $B_{p,p-1}(\gamma)$
      end if
    end for
    $\{$*Update co-boundary relations of* $\gamma\}$
    for each $(p+1)$-simplex θ in $C_{p,p+1}(\sigma)$ **do**
      Add $F(\theta)$ to $C_{p,p+1}(\gamma)$
    end for
  end if
end for
$\{$*Update co-boundary relations of the links* $L\}$
for each p-simplex σ in L **do**
  for each $(p+1)$-simplex θ in $C_{p,p+1}(\sigma)$ **do**
    if θ is in S **then**
      Replace θ by $F(\theta)$ in $C_{p,p+1}(\sigma)$
    end if
  end for
end for
Compute $C^*_{p,q}(\sigma)$ for each p-simplex σ in $S \cup L$



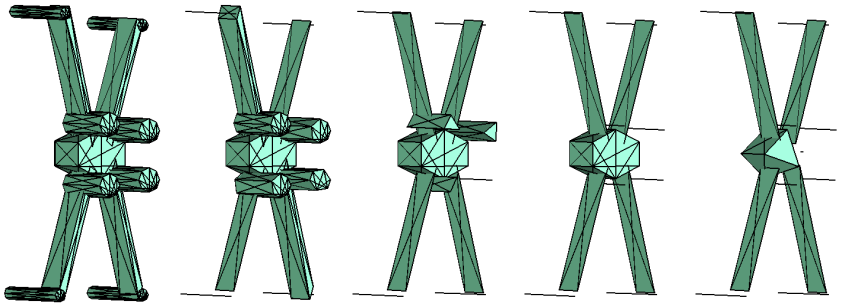
original 100 VPC 300 VPC 650 VPC 670 VPC

(a)



original 100 VPC 500 VPC 620 VPC 640 VPC

(b)



original 150 VPC 200 VPC 210 VPC 215 VPC

(c)

Figure 9: Vertex-Pair Contraction (VPC) on solid models represented as simplicial 2-complexes in E^3

9 Comparison

In this Section, we present comparisons of the IS data structure with data structures for simplicial complexes. We classify the data structures as dimension-independent and dimension-specific, and we compare them in terms of storage costs and efficiency in retrieving topological relations.

9.1 Dimension-independent Data Structures

In this Subsection, we compare the IS data structure with dimension-independent data structures proposed in the literature which have the same representation power, i.e., they can describe arbitrary-dimensional simplicial d-complexes embedded in n-dimensional Euclidean space (with $d \leq n$). One of them is the *Incidence Graph* [15], described in Section 4, and the other is the *Initial Quasi-Manifold (IQM)* data structure we have proposed in [11]. The IS data structure is a further development of the Simplified Incidence Graph (SIG), that we have described in [4] as an attempt to reduce the storage requirements of the IG. The IS data structure has a lower storage cost with respect to the SIG, and it supports more efficient update algorithms.

In [10], we have defined a decomposition for d-dimensional non-manifold objects described through simplicial complexes, which is unique and removes singularities by splitting the complex at non-manifold simplexes only, i.e., at simplexes whose link consists of several connected components. We have called such a decomposition the *standard* decomposition of the original complex, and shown that the components of such decomposition, that we called *Initial Quasi-Manifolds (IQMs)*, admit a local characterization in terms of combinatorial properties around each vertex. A d-dimensional IQM is a simplicial d-complex Γ in which all top simplexes have dimension d and such that the star of each vertex of Γ is $(d-1)$-connected. Up to dimension two, the class of initial quasi-manifolds coincides with that of manifolds. In general, in three or higher dimensions, an IQM is not always a manifold and not even a pseudo-manifold. However, in dimension $d \geq 3$, if an IQM is embeddable in E^d, it must be a pseudo-manifold complex. Figure 10(b) shows an example of a decomposition of the complex depicted in Figure 10(a) into three initial quasi-manifold components. The connection among components is described through the vertices bounding the k-simplexes, which are shared by more than one component. A vertex u of Σ, which is shared by several IQM components, is called a *split vertex*. The copy of split vertex u in a component D_i, to which vertex u belongs, is denoted as u_i and it is called a *vertex copy*. In the example shown in Figure 10, vertex u is split into vertices u_1, u_2 and u_3 in the decomposition.

The IQM data structure proposed in [11] is a two-level data structure encoding the decomposition. The first level represents the relations among the components in the decomposition, which are encoded as a hypergraph H. The nodes in H correspond to IQM components and each hyperarc corresponds to a split vertex u, and it connects all components D_i sharing u. In the hypergraph shown in Figure 10(c), a hyperarc associates u with the three components D_1, D_2 and D_3 through the three vertex copies

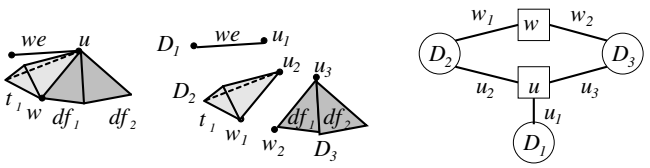


Figure 10: IQM decomposition of a complex

of u shown in Figure 10(b). For every vertex copy u_i corresponding to a split vertex u, the data structure encodes references to the component containing u_i, and to the hyperarc corresponding to u. Each IQM component is encoded with an extension of the Incidence data structure with Adjacencies (IA data structure0, since the star of each vertex in the IQM can be traversed by using $C^*_{0,h}$ plus A_h relations. In the extended IA data structure for an h-dimensional IQM component, only the vertices and h-simplexes are encoded, plus $B_{h,0}$, and A_h at h-simplexes and $C^*_{0,h}$ at vertices. On the other hand, the IA data structure has to be extended since, for instance, a three-dimensional IQM component embedded in 4D space might not be a pseudo-manifold (see [10] for an example) and, thus, a tetrahedron may have more than four face-adjacent tetrahedra.

The IG and IS data structures encode all simplexes of a complex uniquely and explicitly, while the IQM data structure encodes only the vertices and top simplexes explicitly. Thus, If we denote with n_p the number of p-simplexes in a complex Σ, and with n^t_p the number of top simplexes of dimension p, the IQM requires $(d+1)n_d + dn^t_{d-1} + \cdots + 2n^t_1$ integers for encoding the vertex indexes of the top simplexes, while both IG and IS data structure require $(d+1)n_d + dn_{d-1} + \cdots + 2n_1$ integers since they encode the $(p-1)$-faces of all p-simplexes. The IQM stores only one integer for each vertex inside each IQM component, which encodes $C^*_{0,h}$ partial co-boundary relation. This is because of the properties of initial quasi-manifolds. On the other hand, it encodes also the structure of the decomposition and several vertex copies for any non-manifold vertex. Both the IG and IS data structure encode co-boundary relations for every simplex, and thus necessarily require more space. A comparison of the storage costs of IQM, IS data structures and IG can only be done experimentally (see Section 9.2).

The IS data structure and the IG store the same entities, and boundary relations, but the IG encodes the complete co-boundary relations of type $C_{p,p+1}$. Thus, the IG thus occupies $\sum_{p=1}^{d-1}(p+1)n_p - \sum_{q=0}^{d-2}(\kappa_q)$ integers more than the IS. Recall that we denote the total number of connected components in the link of a simplex σ as $\kappa(\sigma)$ (when $dim(\sigma) < d$), and the total number of connected components summed over the links of all p-simplexes in Σ as $\kappa_p = \sum_{dim(\sigma)=p} \kappa(\sigma)$, for $0 \leq p < d$. The above difference is maximum when encoding a manifold complex. In this case, $\kappa_q = n_q$ and thus the difference is $(d+1)n_d + \sum_{q=1}^{d-1} qn_q - n_0$. Figure 11 provides an example at vertex v whose star is a manifold 3-complex. The IG encodes all the seven edges incident at v while the IS encodes only edge e. The difference between the IG and the IS data is minimum when only $(q+1)$-simplexes are incident at all q-simplexes, in which case, $\kappa_q = (q+2)n_{q+1}$. An example for this case is when the whole complex consists of

isolated edges. Only relations $C_{0,1}$ and $C_{1,0}$ are encoded. Then the IG and the IS have the same storage cost.

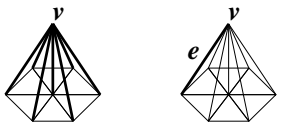


Figure 11: A comparison between $C_{0,1}(v)$ and $C^*_{0,1}(v)$ in the manifold case. In (a) IG encodes all the edges that are incident at v. In (b), IS encodes only edge e in the star of v

Retrieving boundary relations is performed in the same way for IG and IS data structures, and requires constant time. Retrieving co-boundary relation $C_{p,q}(\sigma)$, with $p < q+1$ from the IS data structure requires time linear in the number of top simplexes in the star of σ, while for regular objects, it can be shown that it requires time linear only in the number of q-simplexes in the star of σ from the IG. This also reflects on the time required for retrieving adjacency relations. Boundary relations can be retrieved in constant time from the IQM data structure, co-boundary relations at the vertices can be retrieved in time linear in the number of top simplexes at such vertices, while retrieving $C_{p,q}(\sigma)$, with $p < q$ and $p > 0$ requires time linear in the number of top simplexes in the star of one of the vertices of σ [11].

Update operations can be done efficiently on both the IS data structure and on the IG, while no algorithm has been yet been proposed for updating an IQM data structure. All three data structures exhibit a very low overhead when encoding manifold complexes. In the IQM data structure, non-manifold simplexes are explicit, but they are encoded only through their extreme vertices. The IG does not distinguish between manifold and non-manifold simplexes, while these latter and their incident components are explicit in the IS data structure.

9.2 Dimension-specific Data Structures

In this Subsection, we compare the IS data structure with dimension-specific data structures proposed for two-dimensional and three-dimensional simplicial complexes. We also compare it with two- and three-dimensional instances of the IQM data structure and of the Incidence Graph (IG).

In the two-dimensional case, we consider the *Directed Edge (DE)* data structure [3] and the *Triangle-Segment (TS)* data structure [9]. The DE data structure is an edge-based data structure extending to the non- manifold case the *Half-Edge* data structure proposed for two-dimensional cell complexes with a manifold domain [27]. We have shown in [8] that it is the most space-efficient data structures among edge-based data structures for encoding simplicial 2-complexes. The TS data structure is an adjacency-based compact data structure which extends the IA data structure to arbitrary simplicial

2-complexes embedded in the three-dimensional Euclidean space.

The TS data structure and the 2D instance of the IQM data structure encode only vertices and top simplexes, the IG, and the IS a data structure encode all simplexes, while the DE data structure encodes only edges and vertices explicitly. The TS and the IQM structures are the most compact. We have performed experimental comparisons of the various data structures on manifold and non-manifold complexes [7]. Generally, edge-based data structures require more space than the data structures which encode boundary and co- boundary relations, like IG or IS data structure. The DE data structure is 1.3 to 1.5 times larger then the IG, which is about 1.25 the size of the IS data structure. The IQM data structure is slightly larger than the TS data structure.

Topological relations can be retrieved in optimal time, i.e., in time linear in the number of output simplexes for all data structures, with the exception of the IQM data structure. $C_{1,2}(\sigma)$ can be retrieved in time linear in the number of triangles incident at one of the two extreme vertices of edge σ from the IQM data structure, and the same happens for A_2 relations. All other topological relations can be retrieved in optimal time.

In the three-dimensional case, i.e., for simplicial 3-complexes embedded in the three-dimensional Euclidean space, we compare the IS data structure with the *Non-Manifold Indexed data structure with Adjacencies (NMIA)* [5], that is, to the extent of our knowledge, the only data structure proposed in the literature specifically for simplicial 3-complexes. The NMIA data structure is an adjacency-based data structure which extends to the non- manifold domain the IA data structure. The extension is performed by encoding the multiple connected components at non-manifold vertices and non-manifold edges implicitly. We also perform comparisons with instances of the IG and of the IQM data structure for simplicial 3-complexes embedded in 3D space.

Both the NMIA and the IQM data structures encode only vertices and top simplexes, while the IG and the IS data structure encode all simplexes. We can observe that the NMIA structure encodes information on singularities mainly from the perspective of the top simplexes, that is, given a top simplex, we can tell whether it is a non-manifold singularity, or its boundary is a non-manifold singularity. The IQM structure encodes non-manifold information at the vertices. The IS data structure encodes non-manifold information at all non-manifold simplexes, namely, through the presence of several clusters in its star, while the IG does not distinguish between manifold and non-manifold simplexes.

We have shown experimentally on manifold and non-manifold data sets [7] that IG uses about 1.38 times as much storage as the IS data structure and at least 3 times the storage size of the NMIA. The IQM data structure is only slightly larger than the NMIA data structure and in the manifold case they are almost equivalent.

Table 1 summarizes the time complexity of the algorithms for retrieving topological relations for the NMIA data structure and the 3D instances of the IQM data structure, of the IS data structure and of the IG. Relations $C_{1,3}$ and $C_{0,3}$ can be retrieved from the NMIA data structure in time linear in the number of top simplexes incident at an edge or at a vertex, respectively [5]. Relations $C_{1,3}(\sigma)$ and $C_{1,2}(\sigma)$ can be retrieved from the

IQM data structure in time linear in the number of top simplexes in the star of one of the vertices of σ. A similar behavior occurs for adjacency relations A_1 and A_2.

Relations	NMIA	IQM	IG & IS
$C_{p,q}, p < q$	optimal	optimal	optimal
$C_{2,3}$	optimal	optimal	optimal
$C_{1,3}$	$O(n_\sigma)$	$O(n_v)$ for $v \in B_{1,0}(\sigma)$	optimal
$C_{1,2}$	optimal	$O(n_v)$ for $v \in B_{1,0}(\sigma)$	optimal
$C_{0,3}$	$O(n_\sigma)$	optimal	optimal
$C_{0,2}$ & $C_{0,1}$	optimal	optimal	optimal
A_3	optimal	optimal	optimal
A_2 & A_1	optimal	$O(n_v)$ for $v \in B_{k,0}(\sigma), k=1,2$	optimal
A_0	optimal	optimal	optimal

Table 1: Retrieving topological relations of a simplex σ in 3-complexes: n_σ denotes the number of top simplexes in the star of σ, v denotes a vertex of σ, and n_v denotes the number of top simplexes in the star of v.

10 Concluding Remarks

We have presented a new dimension-independent data structure, the Incidence Simplicial (IS) data structure for representing d-dimensional simplicial complexes in nD Euclidean space. A dimension-independent implementation of the data structure has been presented as well as algorithms for retrieving topological relations and for performing topological updates through vertex-pair contraction. The IS data structure has the same representation power as the widely-used Incidence Graph (IG), but it is more compact, and has the same performances in traversal and manipulation algorithms. We have also compared the IS data structure with another dimension-independent data structure, the IQM data structure, as well as with dimension-specific data structures, also on the basis of an experimental evaluation of their storage costs. We are currently working on applying the IS data structure and its topological manipulation algorithms to the idealization of 2D and 3D finite element meshes generated from CAD models.

A common issue in representing and manipulating non-manifold objects is the availability of large-size simplicial representations for describing such objects. Their complexity can easily exceed the capability of computational tools for analyzing them (for instance, in finite element simulations). In these cases, adaptively simplified meshes, i.e., simplicial complexes in which the level of detail varies in different parts of the object they describe, are often required. On the other hand, accurate mesh simplification algorithms are too time consuming to be performed on-line. Thus, a multi-resolution model, which encodes the modifications performed by a simplification algorithm in a

compact representation, is an effective solution. Dimension-independent non-manifold representations can be integrated with a multi-resolution framework, giving rise to a powerful tool for modeling non-manifold objects at variable resolutions.

The decomposition approach which is at the basis of the IQM data structure defines a general approach to modeling non-manifold multi-dimensional shapes. First, other data structures can be defined for encoding the decomposition and the IQM components. We are currently working on a new incidence-based IQM representation in which the IS data structure is used to encode the IQM components and all non-manifold simplexes are described as hyperarcs in the decomposition graph. We have developed dimension-specific implementations for simplicial 2-complexes and 3-complexes embedded in 3D space. Since no update algorithms are available on the decomposition, our current work is on developing efficient algorithms for updating a standard decomposition when a topological local operator, such as vertex pair contraction, is applied to the underlying complex.

Moreover, the standard decomposition is unique, but there exist admissible decompositions of a complex which have a lower number of components. The uniqueness property is achieved by cutting at all possible non-manifold singularities and thus it tends to over-decompose the complex. In the case of 2-complexes, we are currently developing a decomposition algorithm which computes a more compact decomposition which is also, in the specific 2D case, unique. We are planning to use the resulting decomposition as the basis for performing geometric reasoning on non-manifold 3D shapes. In particular, we are interested in computing topological invariants from the decomposition as signatures for efficient shape analysis and retrieval, and in identifying non-manifold form features based on the structure of the decomposition graph.

11 Acknowledgement

This work has been partially supported by the European Network of Excellence AIM@SHAPE under contract number 506766.

References

[1] B. G. Baumgart. A polyhedron representation for computer vision. In *Proceedings AFIPS National Computer Conference*, volume 44, pages 589–596, 1975.

[2] 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] S. Campagna, L. Kobbelt, and H.-P. Seidel. Directed edges - a scalable representation for triangle meshes. *Journal of Graphics Tools*, 3(4):1–12, 1998.

[4] L. De Floriani, D. Greenfieldboyce, and A. Hui. A data structure for non-manifold simplicial d-complexes. In L. Kobbelt, P. Schroder, and H. Hoppe, editors, *Proceedings ACM/Eurographics Symposium on Geometry Processing*, Nice (France), 8–10 July 2004. ACM Press.

[5] L. De Floriani and A. Hui. A scalable data structure for three-dimensional non-manifold objects. In L. Kobbelt, P. Schroder, and H. Hoppe, editors, *Proceedings ACM/Eurographics Symposium on Geometry Processing*, pages 73–83, Aachen (Germany), 23–25 June 2003.

[6] L. De Floriani and A. Hui. Update operations on 3D simplicial decompositions of non-manifold objects. In D. Fellner, editor, *9th ACM Symposium on Solid Modeling and Applications*, pages 169–180, Genova (Italy), 9–11 June 2004. ACM Press.

[7] L. De Floriani and A. Hui. A compact dimension-independent data structure for non-manifold simplicial complex. Technical report, University of Maryland, College Park, October 2005.

[8] L. De Floriani and A. Hui. Data structures for simplicial complexes: an analysis and a comparison. In M. Desbrun and H. Pottmann, editors, *Third Eurographics Symposium on Geometry Processing*, pages 119–128, Vienna, Austria, July 4-6 2005.

[9] 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.

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

[11] 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.

[12] L. De Floriani, E. Puppo, and P. Magillo. A formal approach to multi-resolution modeling. In W. Strasser, R. Klein, and R. Rau, editors, *Geometric Modeling: Theory and Practice*, pages 302–323. Springer-Verlag, 1997.

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

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

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

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

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

[18] L. Fine, L. Remondini, and J.-C. Léon. Automated generation of FEA models through idealization operators. *International Journal for Numerical Methods in Engineering*, 49:83–108, 2000.

[19] 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.

[20] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, April 1985.

[21] 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.

[22] M. Kallmann and D. Thalmann. Star vertices: a compact representation for planar meshes with adjacency information. *Journal of Graphics Tools*, 6(1):7–18, 2001.

[23] M. Lage, T. Lewiner, H. Lopes, and L. Velho. Chf: a scalable topological data structure for tetrahedral teshes. In *18th Brazilian Symposium on Computer Graphics and Image Processing (Sibgrapi 2005)*, pages 349–356, Oct 2005.

[24] 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. ACM Press.

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

[26] 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.

[27] M. Mantyla. *An Introduction to Solid Modeling*. Computer Science Press, 1987.

[28] S. McMains. *Geometric Algorithms and Data Representation for Solid Freeform Fabrication*. PhD thesis, University of California at Berkeley, 2000.

[29] F. Morando. *Decomposition and Modeling in the Non-Manifold domain*. PhD thesis, February 2003.

[30] D. E. Muller and F. P. Preparata. Finding the intersection of two convex polyhedra. *Theoretical Computer Science*, 7:217–236, 1978.

[31] 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.

[32] S. Pesco, G. Tavares, and H. Lopes. A stratification approach for modeling two-dimensional cell complexes. *Computers and Graphics*, 28:235–247, 2004.

[33] 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, 9–11 June 1999.

[34] J. Rossignac, A. Safonova, and A. Szymczak. 3D compression made simple: Edge-Breaker on a Corner Table. In *Proceedings Shape Modeling International 2001*, Genova, Italy, May 2001. IEEE Computer Society.

[35] 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.

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

[37] 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: Selected and Expanded Papers from the IFIP WG5.2 Working Conference, Rensselaerville, NY, USA, 12-16 May 1986*, pages 3–36. Elsevier Science Publishers B. V. (North–Holland), Amsterdam, 1988. ISBN: 0444704167.

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