Eurographics Symposium on Geometry Processing (2005)
M. Desbrun, H. Pottmann (Editors)

Data structures for simplicial complexes: an analysis and a comparison

Leila De Floriani[1,2] and Annie Hui[2]

[1] Department of Computer and Information Sciences,
University of Genova, Via Dodecaneso, 35 - 16146 Genova (Italy).
[2] Department of Computer Science, University of Maryland, College Park, MD 20742 (USA).

Abstract

In this paper, we review, analyze and compare representations for simplicial complexes. We classify such representations, based on the dimension of the complexes they can encode, into dimension-independent structures, and data structures for three- and for two-dimensional simplicial complexes. We further classify the data structures in each group according to the basic kinds of the topological entities they represent. We present a description of each data structure in terms of the entities and topological relations encoded, and we evaluate it based on its expressive power, on its storage cost and on the efficiency in supporting navigation inside the complex, i.e., in retrieving topological relations not explicitly encoded. We compare the various data structures inside each category based on the above features.

1. Introduction

Cell and simplicial complexes are widely used representations for multi-dimensional geometric objects in geometric and solid modeling, in finite element analysis and in visualisation. In particular, simplicial complexes have received great attention, since their combinatorial properties make them easier to encode and manipulate. Here, we review data structures for general simplicial complexes, i.e., for simplicial complexes which describe non-manifold and non-regular objects in two, three and higher dimensions. A *manifold* (with boundary) is a subset of the Euclidean space for which the neighborhood of each internal point is homeomorphic to an open ball and the neighborhood of each boundary point to an open half-ball. Objects that do not fulfill such properties at one or more points are called *non-manifold* objects. Non-manifold objects, which are also non-regular, contain parts of different dimensionalities.

Several data structures have been proposed in the literature for representing manifold objects discretized as cell or simplicial complexes. Data structures for two-dimensional cell complexes include the *Winged-Edge* [Bau72], the *Half-Edge* [Man87], the *DCEL* [MP78], the *Quad edge* [GS85] and the *Lath* [JLM02]. The *Corner table* [RSS01] data structure has been proposed for two-dimensional simplicial complexes. The *Facet-Edge* [DL89] and the *Handle-Face* [LT97] data structures have been proposed for three-dimensional complexes. Dimension-independent data structures include the *Cell Tuple* [Bri89] and the *nG-map* [Lie91] for cell complexes, and the *Indexed data structure with adjacencies (IA)* [Nie97, PBCF93] for simplicial complexes. Data structures for two- and three-dimensional simplicial complexes have been reviewed in [DKP04].

Here, we focus on data structures for general simplicial complexes. We classify such data structures into three groups: *incidence-based*, *adjacency-based*, and *edge-based*. Incidence-based data structures encode only the incidence relations among simplexes. Such data structures provide a hierarchical perspective of the boundaries and co-boundaries of simplexes. Adjacency-based data structures encode only top simplexes (i.e., simplexes of maximal dimension or which are not on the boundary of any other simplex) with their vertices and their adjacent top simplexes. Usually, this leads to more compact representations. Edge-based data structures are specific for two-dimensional complexes embedded in the 3D Euclidean space. They encode the edge as basic entity and its relations with other simplexes (vertices and triangles). We classify the data structures also in terms of the dimensions of the complexes they describe.

The remainder of this paper is organized as follows. Section 2 provides some background notions. Section 3 reviews and compares three dimension-independent data structures, namely, the *Initial Quasi-Manifold* data structure [DMMP03], the *Incidence Graph*[Ede87] and the *Simplified Incidence Graph*[DGH04]. Section 4 describes a specialized data structure for three-dimensional simplicial complexes, namely the *Non-Manifold Indexed data structure* *with Adjacencies*[DH03], and compares it with the 3D instances of the dimension-independent data structures analyzed in Section 3. Section 5 reviews data structures for two-dimensional simplicial complexes embedded in the 3D Euclidean space. Specifically, we describe a specialization of the *Partial Entities* data structure[LL01] developed for cell complexes, the *Loop Edge-use* data structure[McM00] and the *Directed Edge* data structure[CKS98], which are edge-based. We also describe a compact adjacency-based data structure, the *Triangle Segment*[DMPS04], which explicitly encodes only vertices and top simplexes. Such data structures are compared also with the 2D instances of the dimension-independent data structures presented in Section 3. Finally, Section 6 presents some concluding remarks.

2. Background Notions

In this Section, we review some basic notions about Euclidean simplicial complexes in arbitrary dimensions, and about topological relations.

A Euclidean *simplex* σ of dimension k is the convex hull of $k+1$ linearly independent points in the n-dimensional Euclidean space E^n, $0 \leq k \leq n$. We simply call a *Euclidean simplex* of dimension k a *k-simplex*. k is called the *dimension* of σ and is denoted $dim(\sigma)$. Let V_σ be a set of vertices. Any Euclidean p-simplex σ', with $0 \leq p \leq k$, generated by a set of vertices $V_{\sigma'} \subseteq V_\sigma$ of cardinality $p+1 \leq d$, is called a *p-face* of σ. Whenever no ambiguity arises, the dimensionality of σ' can be omitted, and σ' is simply called a *face* of σ. Any face σ' of σ such that $\sigma' \neq \sigma$ is called a *proper face* of σ. The empty set is a (-1)-face of all simplexes. If σ' is a face of σ, then σ is called a *co-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'. If d is the maximum of the dimensions of the simplexes in Σ, we call Σ a *d-dimensional simplicial complex*, or a *simplicial d-complex*. The *domain*, or *carrier*, of a Euclidean simplicial d-complex Σ embedded in E^n, with $0 \leq d \leq n$, is the subset of E^n defined by the union, as point sets, of all the simplexes in Σ.

The *boundary* $b(\sigma)$ of a simplex σ is the set of all proper faces of σ in Σ. The *co-boundary*, or *star*, of a simplex σ is defined as $\star\sigma = \{\xi \in \Sigma \mid \sigma \text{ is a face of } \xi\}$. In the following, we will call *restricted star* of a simplex σ, $\star\sigma - \{\sigma\}$, and we denote it as $st(\sigma)$. The *link* of a simplex σ is defined as $lk(\sigma) = \{\tau \in \Sigma \mid \exists \xi \in st(\sigma) \text{ such that } \tau \text{ is a co-face of } \xi \text{ and } \tau \notin st(\sigma)\}$. A simplex σ is called a *top simplex* of Σ if $\star\sigma = \{\sigma\}$ A d-complex Σ, in which all top simplexes are d-simplexes, is called *regular* (or *uniformly d-dimensional*).

Two simplexes are called *k-adjacent* if they share a k-face. Two p-simplexes, $0 < p \leq d$, 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 $(h+1)$-simplexes $(\sigma_i)_{i=0}^{k}$ such that two consecutive simplexes σ_{i-1} and σ_i in the sequence are h-adjacent, $0 \leq h \leq d-1$. Two simplexes σ and σ^* are said to be *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 *sub-complex* of Σ is defined by any subset Σ^* of simplexes of Σ such that Σ^* is a simplicial complex. A sub-complex 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 0-connected component is called a *connected component*. We call an *h-cluster* an $(h-1)$-connected component in which all top simplexes have dimension h.

An h-simplex σ in a regular d-complex Σ, $0 \leq h \leq d-1$ is a *manifold* h-simplex if and only if there are at most two $(h+1)$-simplexes incident at σ. An h-path such that any two consecutive simplexes in the path are adjacent through a manifold h-simplex is called a *manifold path*. Two $(h+1)$-simplexes are *h-manifold connected* if and only if there exists a *manifold h-path* connecting them. A regular $(d-1)$-connected d-complex in which all $(d-1)$-simplexes are manifold is called a *(combinatorial) pseudo-manifold* (possibly with boundary). Figure 1(a) shows a complex, which is a 2-cluster but not a pseudo-manifold. A regular d-complex Σ is called an *initial quasi-manifold* if and only if every pair of d-simplexes in the restricted star of every vertex of Σ are $(d-1)$-manifold-connected within the restricted star. Figure 1(b) shows an example of a pseudo-manifold that is not an initial quasi-manifold (because of the star of vertex v).

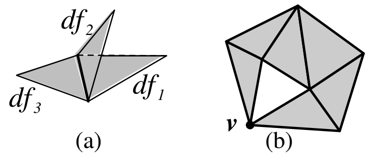


Figure 1: *(a) A 2-cluster which is not a pseudo-manifold, (b) a pseudo-manifold which is not an initial quasi-manifold.*

A simplicial d-complex Σ embedded in the Euclidean d-dimensional space E^d has the following properties:

1. Σ is a pseudo-manifold;
2. The top simplexes incident at any $(d-2)$-simplex σ can be ordered around σ.

If a simplicial d-complex is embedded in the E^n, where $n > d$, then Σ is not necessarily a pseudo-manifold, i.e., we can have several d-simplexes in the star of a $(d-1)$-simplex. The second property holds for all $(n-2)$-simplexes, if $d \leq n-1$.

We call a k-simplex σ in a simplical d-complex a *non-manifold simplex* if and only if $lk(\sigma)$ consists of more than one connected component. Figure 2 shows examples of non-manifold simplexes in a simplicial 3-complex. Figure 2(a) shows a non-manifold edge e, the link of e is highlighted in Figure 2(b). Figure 2(c) shows an example of a non-manifold vertex v.

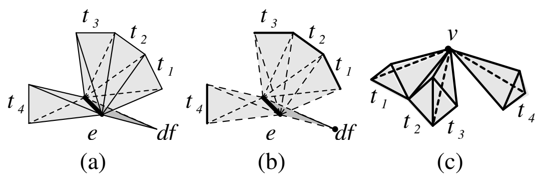


Figure 2: *Singularities in 3D simplicial complexes*

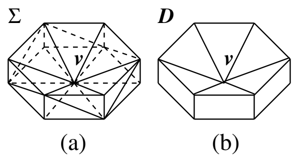


Figure 3: *A 3D simplicial complex* Σ *(a) and a hexagonal pinched pie subdivided in* Σ *(b)*

There is a distinction between the non-manifold properties of a simplicial complex and the non-manifold properties of its domain. For instance, vertex v in the pinched pie in Figure 3(a) does not satisfy our definition of a non-manifold vertex, but it is a non-manifold singularity in the domain of the complex, see Figure 3(b).

Let Σ be a simplicial d-complex and let $\sigma \in \Sigma$, with $0 \leq p \leq d$. We define the following *topological relations*:

- For $0 \leq q \leq p-1$, *boundary relation* $R_{p,q}(\sigma)$ consists of the set of q-simplexes in the set of faces of σ.
- For $p+1 \leq q \leq d$, *co-boundary relation* $R_{p,q}(\sigma)$ consists of the set of q-simplexes in the star of σ.
- For $p > 0$, *adjacency relation* $R_{p,p}(\sigma)$ is the set of p-simplexes in Σ that are $(p-1)$-adjacent to σ.
- *Adjacency relation* $R_{0,0}(\sigma)$, where σ is a vertex, consists of the set of vertices σ' such that $\{\sigma, \sigma'\}$ is an edge of Σ.

We call *constant* any relation which involves a constant number of entities. Note that boundary relations are constant in a simplicial complex. Relations, which involve a variable number of entities, are called *variable*. Co-boundary and adjacency relations are variable relations. We call an algorithm which retrieves a topological relation R *optimal* if and only if it retrieves relation R in time linear in the number of entities involved in R.

3. Data structures for d-dimensional simplicial complexes

A widely-used dimension-independent data structure for simplicial d-complexes is the *Indexed data structure*, which encodes, for each top k-simplex σ, relation $R_{k,0}(\sigma)$, i.e., the indexes to its $(k+1)$-vertices. Only boundary relations of type $R_{k,j}(\sigma)$, $j < k$, for any top k-simplex σ, can be extracted in optimal time from it. Note that the j-simplexes on the boundary of σ are described through $j+1$ vertex indexes. Here, however, we are interested in so-called *topological data structures*, which also encode adjacency and incidence information among simplexes.

A very common data structure used for simplicial pseudo-manifolds is the *Indexed data structure with Adjacencies (IA)*[Nie97] (also called *winged representation* [PBCF93]), which encodes, for each d-simplex σ in a complex, relations $R_{d,0}(\sigma)$ and $R_{d,d}(\sigma)$ (which is a constant relation for pseudo-manifolds). The IA data structure can be extended, when restricted to initial quasi-manifolds, by encoding, for each vertex v, relation $R_{0,d}(v)$, i.e., one d-simplex in the star of v. This extension allows extracting all simplexes in the star of a vertex in time linear in the number of such simplexes, i.e., all $R_{0,k}(v)$ relations, where $0 \leq k \leq d$, in time linear in the number of k-simplexes in $R_{0,k}(v)$, which is optimal.

To the extent of our knowledge, only three dimension-independent topological data structures have been proposed in the literature for d-dimensional simplicial complexes with a completely general domain, namely, the *Initial Quasi-Manifold (IQM) data structure* [DMMP03], the *Incidence Graph* [Ede87] and the *Simplified Incidence Graph (SIG)* [DGH04].

3.1. The Initial Quasi-Manifold data structure

The *Initial Quasi-Manifold (IQM)* data structure [DMMP03] describes the decomposition of a simplicial complex into k-dimensional initial quasi-manifold components, which are nearly manifold in their properties. Intuitively, the decomposition of Σ is obtained by cutting the complex at non-manifold simplexes. Figure 4(b) shows an example of a decomposition of the complex shown in Figure 4(a) into three initial quasi-manifold components. In [DMMP03], an algorithm has been proposed, which computes a unique decomposition, called *standard decomposition*, of a simplicial d-complex into initial quasi-manifold components.

The basis of the IQM data structure are an extended indexed data structure with adjacencies to encode each IQM component and a hypergraph describing how the components are connected together in the decomposition. An h-dimensional IQM can be effectively described by an extended indexed data structure with adjacencies, since the star of each vertex in the IQM can be traversed by using relations $R^*_{0,h}$ plus $R_{h,h}$.

The connection among components is described through the vertices bounding the k-simplexes, which are shared by more than one IQM component. A vertex v of Σ, which is shared by several IQM components, is called a *split vertex*. The copy of split vertex v in a component C_i, to which vertex v belongs, is denoted as v_i and it is called a *vertex copy*. The relations among the components in an IQM decomposition of a complex described by the split vertices is represented as a hypergraph H, in which the nodes correspond to IQM components and each hyperarc corresponds to a split vertex v and it connects all components C_i sharing v. In the example shown in Figure 4, vertex v in Figure 4(a) is split into vertices v_1, v_2 and v_3 in the decomposition shown in Figure 4(b). In the hypergraph shown in Figure 4(c), a hyperarc associates v with the three components C_1, C_2 and C_3 through the three vertex copies.

The hypergraph is encoded in the following data structure:

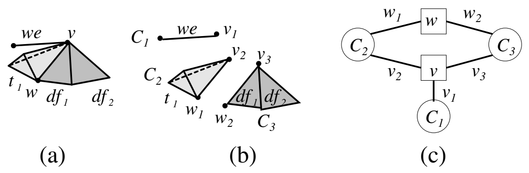


Figure 4: *IQM decomposition of a complex*

- for each component C_i: a reference to the extended IA data structure describing C_i;
- for each hyperarc: the corresponding split vertex v and the vertex copies of v;
- for every vertex copy v_i corresponding to split vertex v:
  - the component containing v_i;
  - a reference to its hyperarc, i.e., v.

The hypergraph supports a vertex-based traversal among components connected through the same hyperarc. Given a vertex copy v_i from any component C_i, we can follow the reference to its hyperarc and find all other vertex copies v_j connected with v, as well as all other components sharing v.

3.2. The Incidence Graph

The *Incidence Graph (IG)* [Ede87] encodes all the simplexes of any given simplicial d-complex Σ, and the following topological relations:

- for each p-simplex σ, where $0 < p \leq d$, boundary relations $R_{p,p-1}(\sigma)$,
- for each p-simplex σ, where $0 \leq p < d$, co-boundary relations $R_{p,p+1}(\sigma)$

Thus, for each p-simplex σ, the IG encodes its immediate boundary, and its immediate co-boundary.

3.3. The Simplified Incidence Graph

The *Simplified Incidence Graph (SIG)* [DGH04] is based on the concept of *h-cluster* (see Section 2. The SIG encodes all simplexes in a simplicial complex Σ as well as the following topological relations:

- for each p-simplex σ, where $0 < p \leq d$, boundary relations $R_{p,p-1}(\sigma)$,
- for each p-simplex σ, where $0 \leq p < d$, partial co-boundary relations $R^*_{p,q}(\sigma)$ (where $q > p$), which is defined as follows: $R^*_{p,q}(\sigma)$ consists of one arbitrarily-selected q-simplex, for each q-cluster in the restricted star $st(\sigma)$ of σ. In the example of Figure 2(c), t_2 and t_3 form a 3-cluster. $R^*_{0,3}(v) = \{t_1, t_2, t_4\}$.

Note that partial co-boundary relation $R^*_{d-1,d}(\sigma)$ is the same as co-boundary relation $R_{d-1,d}(\sigma)$. If the domain of Σ is manifold, all partial co-boundary relations are empty with the exception of $R^*_{p,d}(\sigma)$, which consists of at most two d-simplexes.

3.4. Comparison

All three data structures have the same expressive power. While the IQM data structure is obviously decomposition-based, the SIG encodes somehow a decomposition of the star of a simplex. Note that the two decompositions are not the same. A manifold-connected h-component is generally more tightly connected than an h-cluster. The star of vertex v in the complex in Figure 5(a) consists of four IQM components as shown in Figure 5(b), but it has only two 3-clusters and one 2-cluster, as shown in Figure 5(c).

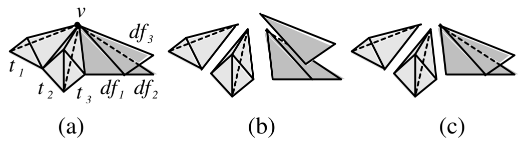


Figure 5: *(a) Star of v; (b) IQM components at v; and (c) Clusters at v*

The IQM data structure encodes only the top simplexes and the vertices, while both the IG and the SIG encode all simplexes. The IQM data structure is adjacency-based, since it encodes the adjacency relation among top simplexes just like the IA data structure and its extensions. The other two are incidence-based, since they encode only incidence relations. In the IG, the number of co-boundary relations encoded is the same as the boundary relations encoded. The number of partial co-boundary relations encoded in the SIG depends on the number of q-clusters incident at each simplex. The IG is conceptually simpler, but definitely less compact than the other two data structures.

A p-simplex σ, that is not explicitly encoded in the IQM data structure, is represented either through a subset of vertices in $R_{k,0}(\sigma')$, where σ' is a top k-simplex incident at σ, or by addressing σ as a p-face of σ'. Since it generalizes the extended indexed data structure with adjacencies, the IQM structure most efficiently answers queries on top h-simplex σ regarding its vertices (i.e., $R_{h,0}(\sigma)$ relation) and its neighbours of dimension h (i.e., $R_{h,h}(\sigma)$ relation). For extracting co-boundary relation for a simplex σ, the general strategy is to retrieve a vertex v of σ, traverse all the top simplexes incident at v and extract from them the relevant simplexes that are incident at σ.

The IG most efficiently supports queries about the immediate boundary $R_{h,h-1}(\sigma)$, or co-boundary $R_{h,h+1}(\sigma)$ of an h-simplex σ. It supports a simple recursive strategy to retrieve all other topological boundary and co-boundary relations. Adjacencies relations are retrieved in two steps: first, the immediate boundary of the query p-simplex σ is retrieved; and then, for each $(p-1)$-face of σ, its immediate co-boundary simplexes are extracted.

Boundary relations are retrieved from the SIG in the same way as from the IG. The general strategy for retrieving co-boundary relations at a simplex σ consists of performing a traversal of the star of σ, and then retrieving the boundary relations of the top simplexes in the star σ. The strategy for retrieving adjacency relations at σ consists of retrieving the co-boundary relations for the simplexes that are in the boundary relation of σ [DGH04].

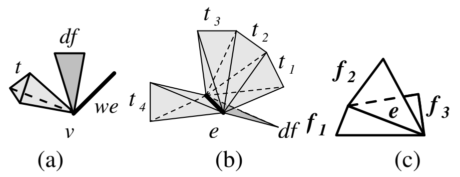


Figure 6: *Properties of simplicial complexes in 3D space*

In summary, all three data structures can support the retrieval of boundary relations $R_{p,q}, p > q$ and of co-boundary relation $R_{d-1,d}$ and $R_{0,q}$ in optimal time. The other co-boundary relations $R_{p,q}(\sigma), 0 < p < q$ can be retrieved from the IG and the SIG in optimal time but not from the IQM data structure. Similarly, in retrieving adjacency relations, the IG and the SIG are optimal, but the IQM data structure is not.[DH05]

4. Data structure for 3D simplicial complexes

In this Section, we discuss a data structures specific for three-dimensional simplicial complexes, the *Non-Manifold Indexed data structure with Adjacencies (NMIA)* [DH03]. We compare it with the three-dimensional instances of the data structures described in Section 3.

We consider simplicial 3-complexes embedded in E^3. In such complexes, we call a top 2-simplex a *dangling face* (for instance, triangle df in Figure 6(a)), and a top 1-simplex a *wire edge* (for instance, edge we in Figure 6(a)). Each sub-complex of $st(e)$ bounded by a connected component of $lk(e)$ at an edge is called an *edge-based cluster*. An edge-based cluster may consist of just one single dangling face (such as df in Figure 6(b) and the three faces in Figure 6(c)), one single tetrahedron, or a fan of tetrahedra (such as t_1, t_2 and t_3 in Figure 6(b)).

4.1. The Non-Manifold Indexed data structure with Adjacencies

The *Non-Manifold Indexed data structure with Adjacencies (NMIA)* [DH03] encodes the vertices, all the top simplexes of a simplicial 3-complex and the following topological relations:

- For each tetrahedron t:
  - relation $R_{3,0}(t)$;
  - relation $R_{3,3}(t)$;
  - for each non-manifold edge e of t, relation $R_{3,clusters}(t)$, which encodes the preceeding and the succeeding top simplexes, around e, when that top simplex is not in the same edge-based cluster as t. For example, in Figure 6(b), $R_{3,clusters}(t_3)$ at edge e consists of t_4, but not t_2.
- For each dangling face f:
  - relation $R_{2,0}(f)$;
  - for each non-manifold edge e of f, relation $R_{2,clusters}(f)$, which encodes the preceeding and the succeeding top simplexes, around e. In Figure 6(b), $R_{2,clusters}(df)$ at edge e consists of t_4 and t_1
- For each wire edge w: relation $R_{1,0}(w)$;
- For each vertex v: relation $R_{0,clusters}(v)$ which encodes one top simplex from each connected component in the restricted star of v

The NMIA data structure is a non-manifold extension of the extended IA data structure (IA) and thus it is an adjacency-based data structure. The multiple connected components at non-manifold vertices and non-manifold edges through relations $R_{2,clusters}$ for dangling faces, $R_{3,clusters}$ for tetrahedron, and $R_{0,clusters}$ at vertices.

The storage cost of the NMIA data structure is as follows: $8n_3 + 3n_2^t + 2n_1^t + C_e + C_v$, where n_3 is the total number of tetrahedra, n_2^t the total number of dangling faces, n_1^t the total number of wire edges, C_e the total number of edge-based clusters at non-manifold edges and C_v the total number of connected-components at all vertices. When the domain of the complex is manifold, $C_e = n_2^t = n_1^t = 0$ and $C_v = n_0$. Thus the NMIA structure encodes $8n_3 + n_0$, exactly the same amount of information as the extended IA structure for simplicial 3-complexes.

4.2. Comparison

Both the NMIA and the IQM data structures encode only top simplexes, while the IG and the SIG 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 SIG encodes non-manifold information at the non-manifold simplexes, namely, through the presence of several clusters in its star. The IG makes no explicit distinction between manifold and non-manifold simplexes.

We compare the storage cost of the NMIA, of the 3D instances of the IG, the SIG and the IQM data structures on the five data sets shown in Table 1(a). The bunny data set is manifold. The spider contains comparable numbers of tetrahedra and dangling faces. The other data sets have small number of singularities. Table 1(b) shows the storage cost of the four data structures. We can see the the IG uses about 1.38 times as much storage as the SIG 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.

All four data structure are able to support the retrieval of all topological relations. The NMIA data structure behaves like the IQM structure in extracting boundary relations. For co-boundary relations, top simplexes incident at an edge e are ordered as clusters. Thus, they can be retrieved by using relations $R_{3,clusters}$ and $R_{2,clusters}$ at the top simplexes incident at e. Top simplexes incident at a vertex v must be retrieved by traversing the star of v through relation $R_{0,clusters}$.

Table 2 summarizes the time complexity of the algorithms

Data set	V	WE	DF	T	C_e	$C_v - n_0$	n_{iqm}
bunny	443	0	0	1996	0	0	1
dragon	1485	0	0	4996	0	8	1
hand	3139	1	1	9986	4	4	2
chime	246	7	9	360	18	30	27
spider	1250	0	1989	503	44	0	2

(a)

Data set	NMIA	IQM	IG	SIG
bunny	16.4k	16.4k	51.4k	36.8k
dragon	41.46k	41.48k	137.2k	97.70k
hand	83.04k	83.05k	276.8k	197.1k
chime	3.20k	3.32k	11.90k	8.51k
spider	17.22k	17.36k	39.56k	30.06k

(b)

Table 1: *(a) Five data sets: V=# vertices, WE=# wire edges, DF=# dangling faces, T=# tetrahedra; (b) Storage cost of the four data structures for the data sets in (a).*

for retrieving topological relations for the NMIA data structure and the 3D instances of the IQM data structure, of the SIG and of the IG. Relations $R_{1,3}$ and $R_{0,3}$ can be retrieved from the NMIA data structure in sub-optimal time. [DH03]

Relations	NMIA	IQM	IG & SIG
$R_{p,q}, p<q$	optimal	optimal	optimal
$R_{2,3}$	optimal	optimal	optimal
$R_{1,3}$	$O(n_\sigma)$	$O(n_v)$ for $v \in R_{1,0}(\sigma)$	optimal
$R_{1,2}$	optimal	$O(n_v)$ for $v \in R_{1,0}(\sigma)$	optimal
$R_{0,3}$	$O(n_\sigma)$	optimal	optimal
$R_{0,2}$ & $R_{0,1}$	optimal	optimal	optimal
$R_{3,3}$	optimal	optimal	optimal
$R_{2,2}$ & $R_{1,1}$	optimal	$O(n_v)$ for $v \in R_{k,0}(\sigma), k=1,2$	optimal
$R_{0,0}$	optimal	optimal	optimal

Table 2: *Retrieving topological relations of a simplex σ in 3-complexes: n_σ denotes the number of top simplexes in the star of σ, v is a vertex of σ, and n_v denotes the number of top simplexes in the star of v.*

Because of its compactness, the implementation of the NMIA data structure is more complex than the SIG or the IG. A non-optimized implementation of the IQM data structure is not difficult. The major problem is that the decomposition algorithm needs to be applied to any given simplicial complex for computing it [DMMP03]. Algorithm for updating an NMIA data structure through edge collapse and its reverse, vertex split, are described in [DH04]. We have developed algorithms for performing vertex pair collapse on the SIG [DH05]. Similar algorithms have been proposed for the IG [PH97]. No algorithm has been developed to update the decomposition on which the IQM data structure is based when the underlying simplicial complex is modified.

5. Data structures for 2D simplicial complexes

In this Section, we discuss data structures for two-dimensional simplicial complexes embedded in E^3. We review and analyze an adjacency-based data structure called the *Triangle-Segment (TS)* data structure [DMPS04], which extends the IA data structure to general simplicial complexes, and three edge-based data structures, namely:

- a specialization we have performed of the *Partial Edge (PE)* data structure [LL01] originally designed for cell 2-complexes embedded in E^3,
- the *Loop Edge-use (LE)* data structure [McM00], which specializes the Radial Edge (RE) data structure proposed by Weiler [Wei88] for cell 2-complexes to regular simplicial 2-complexes, in which the star of a vertex consists of a single connected component,
- the *Directed Edge (DE) data structure* [CKS98], which can encode any simplicial complexes embedded in E^3.

The specializations of the PE and the LE are detailed in [DH05]. We have also defined a specialization of the RE data structure to general simplicial complexes, which we do not present here, since the PE data structure is a more compact representation with the same expressive power as the RE data structure [LL01].

5.1. The Triangle-Segment data structure

The *Triangle-Segment (TS)* data structure [DMPS04] encodes all vertices, and top simplexes, i.e., triangles and wire edges in a simplicial complex toegether with the following topological relations:

- For each triangle t:
  - boundary relation $R_{2,0}(t)$;
  - relation $R^*_{2,2}(t)$ that, for each edge e of t, encodes the triangle(s) that are immediately preceding and succeeding t in counter-clockwise order around edge e. In the example of Figure 6(c), $R^*_{2,2}(f_2) = \{f_1, f_3\}$.
- For each vertex v:
  - relation $R^*_{0,2}(v)$ which encodes one triangle for each connected component of $lk(v)$
  - relation $R^*_{0,0}(v)$, which is the $R_{0,0}(v)$ relation restricted to wire edges. Thus, w is in $R^*_{0,0}(v)$ if and only if $\{v,w\}$ is a wire edge.

In the TS data structure, edges are not encoded. Wire edges are implicitly represented through $R^*_{0,0}$ relations. Notice that relation $R^*_{2,2}(t)$ in the TS is equivalent to the $R_{2,clusters}(t)$ relation in the NMIA data structure. Relations $R^*_{0,2}(v)$ and $R^*_{0,0}(v)$ together are equivalent to $R_{0,clusters}(v)$

relation. The only difference is that wire edges are implicitly described in the TS data structure.

The storage cost of the TS data structure is equal to $6n_2 + C_e + C_v$, where n_2 is the total number of triangles, C_e the total number of edge-based clusters at non-manifold edges and C_v the total number of connected-components at all vertices. For a manifold domain, $C_e = 0$ and $C_v = n_0$, so the TS encodes $6n_2 + n_v$ pieces of information, which is the same as the space required by the 2D instance of the extended IA data structure.

Algorithms for retrieving topological relations are described in [DH05]. In [DMPS04], algorithms for performing vertex pair contraction and vertex split on a simplicial complex encoded in the TS data structure are presented. All topological relations can be retrieved in optimal time from the TS data structure [DH05].

5.2. Edge-based data structures

5.2.1. The Partial Entities data structure

The *Partial Entities (PE) data structure* [LL01] has been proposed for cell complexes. It encodes: *edges*, and *vertices*, plus two oriented entities, i.e., *faces* and *partial-edges*. Each face is bounded by one oriented loop, which consists of a circle of partial-edges. Each partial-edge corresponds to the appearance of an edge on the oriented loop bounding a face. Thus, if there are m faces incident at edge e, the PE data structure stores m partial-edges corresponding to e. A wire edge *we* has a loop that consists of two partial-edges of *we*.

We have simplified the original PE data structure, by dropping the loops and entities like the *shell* and the *region*, which have been introduced in the PE structure to encode objects with several boundaries and several connected components. The PE data structure for simplicial complexes encodes the following information (we refer to Figure 7 to illustrate it):

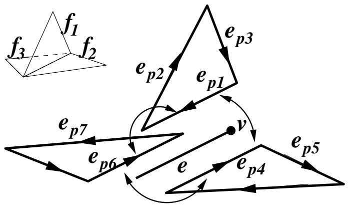


Figure 7: *Elements of the PE structure: relations at a non-manifold edge e shared by three faces: f_1, f_2, f_3*

- For each face f: a reference to a partial-edge on its boundary. (In Figure 7, f_1 has a reference to e_{p1});
- For each edge e, a reference to a partial-edge that describes e. (In Figure 7, e has a reference to e_{p1});
- For each partial-edge e_p bounding face f, there is a reference to each of the following: (see e_{p1} in Figure 7 as an example)
  - the corresponding edge e (e in the example);
  - the face f (f_1 in the example);
  - the previous adjacent partial-edge ordered in counter-clockwise direction around e (e_{p4} in the example);
  - the next adjacent partial-edge ordered around e (e_{p6} in the example);
  - the previous partial-edge in counter-clockwise direction on the boundary of f (e_{p3} in the example);
  - the next partial-edge on the boundary of f (e_{p2} in the example);
  - the start vertex of e_p (v in the example);
- For each vertex v: the list of partial-edges e_p that start at v (In Figure 7, v has references to e_{p1}, e_{p5} and e_{p7}.)

We can express the information encoded in the specialized PE data structure in terms of topological relations. The formalization of the PE data structure can be summarized as follows:

- For each triangle f: relation $R^*_{2,1}(f)$, which encodes one edge on the boundary of f,
- For each edge e:
  - Relation $R_{1,2}(e)$, ordered around edge e, so that the i-th element in $R_{1,2}(e)$ is the i-th triangle incident at e;
  - Partial relation $R^*_{1,1}(e)$ which is defined as follows: $R^*_{1,1}(e)$ consists of the edges on the boundary of the triangles incident at e and sharing one extreme vertex with e. The elements in relation $R^*_{1,1}(e)$ are ordered so that both the $2i$-th and the $(2i+1)$-th elements in $R^*_{1,1}(e)$ are on the i-th triangle in $R_{1,2}(e)$.
  - Relation $R_{1,0}(e)$;
- For each vertex v: relation $R_{0,1}(v)$, unordered.

Relations $R_{1,2}(e)$, $R^*_{1,1}(e)$ and $R_{1,0}(e)$ for edge e describe the information encoded at edges. $R_{1,2}(e)$ describes the relation between an edge and a triangle defined by a partial-edge. Relation $R^*_{1,1}(e)$ captures the association between a partial-edge e_p and the edges following and preceeding e_p in the boundary of the triangle f with which e_p is associated. The adjacency of partial-edges at the same edge e is implicitly expressed through the order in $R^*_{1,1}(e)$.

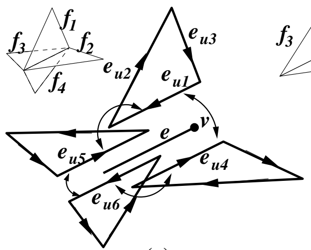
5.2.2. The Loop Edge-use data structure

The *Loop Edge-use (LE) data structure* [McM00] simplifies the RE data structure [Wei88] developed for cell 2-complexes embedded in E^3 to regular simplicial complexes in which non-manifold singularities occur only at edges. Thus, the LE data structure does not represent wire edges, and is based on the assumption that the link of each vertex consists of a single connected component.

In this representation, each 2-simplex has a single orientation. The entities encoded by the LE data structure are: faces, edges, vertices and *edge-uses*, where an edge-use is similar to the partial-edge in the PE data structure and represents the association between an edge and a triangle incident at it. The LE structure encodes the following information: (we refer to the examples in Figure 8 to illustrate it)

- For each face f, a reference to one edge-use on the boundary of f (In Figure 8(a), f_1 has a reference to e_{u1});

- For each edge e, a reference to one edge-use that is associated with e (In Figure 8(a), e has a reference to e_{u1});
- For each edge-use e_u on the boundary of face f, there is a reference to each of the following: (We illustrate the fields of e_u with e_{u1} in Figure 8(a). For clarity, we illustrate the last of field of e_u with e_{u1} in Figure 8(b).)
  - the face f (f_1 in the example);
  - the corresponding unoriented edge e (e in the example);
  - the next adjacent edge-use ordered around e in counter-clockwise direction (e_{u5} in the example);
  - the edge-use following e_u in counter-clockwise direction on the boundary of f (e_{u2} in the example);
  - the start vertex v of e_u (v in the example);
  - the next edge-use that starts at v (In the example of Figure 8(b), the three edge-uses starting at v are e_{u1}, e_{u5} and e_{u7} in counter-clockwise order. e_{u1} has a reference to e_{u5});
- For each vertex v, a reference to an edge-use that starts at v (In Figure 8(b), v has a reference to e_{u1});



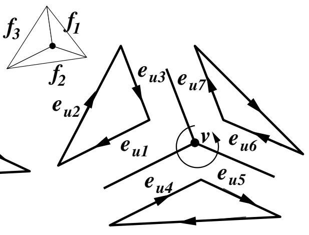


Figure 8: *Elements of the LE structure: (a) Edge-uses of a non-manifold edge e shared by four faces: f_1, f_2, f_3, f_4; (b) Edge-uses starting at a vertex v*

The formalization of the LE data structure in terms of topological relations can be expressed as follows:

- For each triangle f: Relation $R^*_{2,1}(f)$,
- For each edge e:
  - Relation $R_{1,2}(e)$, ordered around edge e, so that the i-th element is the i-th triangle at e;
  - Partial relation $R^*_{1,1}(e)$, as defined for the PE data structure (see Subsection 5.2.1);
  - Relation $R_{1,0}(e)$;
- For each vertex v: Partial relation $R^*_{0,1}(v)$, that associates v with one edge incident at v.

Note that it is almost identical to that of the PE data structure. The only difference is that $R_{0,1}$ is encoded as a partial relation since the LE data structure does not represent non-manifold singularities at vertices.

5.2.3. The Directed Edge data structure

The *Directed Edge (DE)* data structure [CKS98] is an extension of the *Half-Edge* data structure [Man87] proposed for two-dimensional cell complexes with a manifold domain. It is based on the concept of directed edge. A *directed edge* e_d of an edge e in a 2-complex is an occurence of e on the boundary a triangle incident at e. A directed edge is similar to the *edge-use* in the LE data structure and to the *partial-edge* in the PE data structure.

In the DE data structure, the entities are directed edges and vertices. Triangles and unoriented edges are not explicitly encoded. Triangles are implicitly encoded by the edges on their boundary. The association between a triangle and its three edges is through indexing. The i-th triangle, f_i is described by the $3i$-th, $(3i+1)$-th and $(3i+2)$-th directed edges, which form the oriented boundary of f_i. Wire edges are represented as directed edges. The DE data structure encodes the following information:

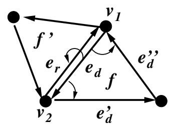


Figure 9: *An illustration of the relations encoded at a directed edge e_d*

- For each triangle f, the three directed edges on the boundary of f;
- For each directed edge e_d on the boundary of face f, there is a reference to each of the following entities: (see Figure 9 for the illustration of the symbols)
  - the start vertex v_1;
  - the end vertex v_2;
  - the adjacent directed edge e_r that is incident at v_1 and v_2;
  - the previous directed edge e''_d bounding f in counter-clockwise order;
  - the next directed edge e'_d bounding f in counter-clockwise order;
- For each vertex v, one directed edge from each connected component of the link of v.

We can formalize the topological relations encoded in the DE data structure as follows:

- For each face f: $R_{2,1}(f)$, which is encoded implicitly (the i-th directed edge belongs to the $(i/3)$-th triangle);
- For each edge e:
  - Relation $R_{1,0}(e)$;
  - Partial relation $R^*_{1,1}(e)$, as defined for the PE data structure (see Subsection 5.2.1);
- For each vertex v: Partial relation $R^*_{0,1}(v)$, which consists of one edge for each connected component of the link of v

In our formalization, we have encoded the unoriented edge e, instead of its oriented edges and, thus, the information in the directed edges in the DE data structure has been transferred to the unoriented edge and described by partial relation $R^*_{1,1}(e)$, and in its ordering.

Thus, the DE structure encodes almost the same relations as the PE data structure except for relation $R_{0,1}(v)$ at vertex

v which is partially encoded in the DE data structure, but completely encoded in the PE data structure.

5.2.4. Discussion

The three edge-based structures are very similar. The LE data structure has a storage cost that is half of that of the PE data structure. In the PE data structure, each partial-edge has a reference to its previous and next partial-edge in ccw order on the same face, and a reference to its previous and next adjacent partial-edge associated with the same edge e in ccw order. In the LE data structure, only the edge-uses following in ccw order are referenced in both cases. Figure 10 illustrates the differences through the arrows which show the references encoded in each structure. The DE data structure is slightly more compact than the LE data structure because faces and unoriented edges are not encoded explicitly.

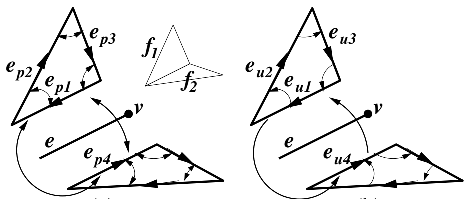


Figure 10: *Comparison between (a) the PE and (b) the LE data structures*

All topological relations can be retrieved from the edge-based data structures in optimal time, and the retrieval algorithms are straightforward [DH05]. For example, retrieving face-based boundary relations $R_{2,1}$, $R_{2,0}$ involves following the oriented edges on the loop bounding the face. Retrieval of $R_{1,2}(e)$ for an edge e involves extracting all the oriented edges ordered around the unoriented edge e.

5.3. Comparison

The TS data structure and the 2D instance of the IQM data structure encode only top simplexes. The IG, the SIG, the PE and the LE data structures encode all simplexes. The DE data structure encodes only oriented edges and vertices. The LE data structure represents only a subset of regular simplicial complexes while all others represent arbitrary non-manifold complexes.

The TS and the IQM structures are the most compact ones because they encode only top simplexes. We compare the various data structures on the data sets in Table 3(a). The duck and the head are manifold data sets. The grid is a regular data set. The others contain non-manifold vertices and edges. Generally, edge-based data structures require more space than the incidence-based structures. The DE data structure is 1.3 to 1.5 times the size of the IG. It is about 1.25 the size of the SIG. The IQM is slightly larger than the TS.

All edge-based data structures (the PE, the LE and the DE data structures), all the incidence-based data structures (the IG and the SIG) and the TS data structure support the retrieval of all topological relations in optimal time.

Data set	V	WE	F	C_e	$C_v - n_0$
duck	93	0	182	0	0
head	426	0	829	0	0
grid	200	0	492	256	0
gull	33	1	56	26	2
finch	629	12	1220	6	24
butterfly	68	11	90	49	21

(a)

Data set	PE	LE	DE	TS	IQM	SIG	IG
duck	6918	3278	2823	1185	1186	1731	2184
head	31.5k	14.9k	13.0k	5379	5380	7908	9990
grid	18.9k	8778	7644	3280	3728	4628	5776
gull	2139	-	876	366	379	534	664
finch	46.5k	-	19.0k	7981	8069	11.6k	14.7k
butterfly	3539	-	1451	628	714	905	1112

(b)

Table 3: *(a) Seven data sets: V=# vertices, WE=# wire edges, F=# triangles; Storage cost of data structures (b) for 2D data sets in (a).*

Aspect	PE	LE	DE	TS	IQM	SIG	IG
Domain	NM	REG	NM	NM	NM	NM	NM
Entities	all	all	all	top	top	all	all
Storage cost	high	high	med	low	low	med	med
Retrieval of relations	opt	opt	opt	opt	cobound sub-opt	opt	opt

Table 4: *Summary of comparisons for data structures for 2-complexes. For domain: NM=non-manifold, REG=regular. For entities: top=top simplexes, all=all simplexes. For retrieval of relations: opt=optimal, sub-opt=sub-optimal.*

In the IQM data structure, relations $R_{1,2}$ and $R_{2,2}$ can be retrieved in sub-optimal time because retrieving these relations from the IQM structure involves visiting all components at a vertex of the edge, and these components may consist of wire edges. Table 4 summarizes the comparisons among the data structures for 2D simplicial complexes.

6. Conclusion

We have reviewed data structures for simplicial complexes. The IQM data structure, the IG and the SIG are dimension-independent. The NMIA data structure is specialized for complexes in E^3. We have reviewed the TS data structure and three edge-based data structures, namely, the PE, the LE and the DE data structures, for 2D simplicial complexes.

The most compact data structures are adjacency-based representations, while edge-based ones are the most spaceconsuming. Incidence-based structures are in the middle range for storage cost.

Almost all these data structures support the retrieval of topological relations in optimal time, except for the IQM and the NMIA data structures for selected relations. The IQM data structure is more difficult to update than the other ones because the decomposition of the complex, on which it is based, must be maintained at each update.

7. Acknowledgement

This work has been partially supported by project funded by the Italian Ministry of Education, University, and Research (MIUR) on *Representation and Management of Spatial Data in the WEB* (SPADA-WEB) and by the European Network of Excellence AIM@SHAPE under contract number 506766.

References

[Bau72] BAUMGART B. G.: *Winged-edge polyhedron representation*. Technical Report CS-TR-72-320, Stanford University, Dept. of Computer Science, October 1972. 1

[Bri89] BRISSON E.: Representing geometric structures in D dimensions: topology and order. In *Proc. 5th ACM Symp. on Computational Geometry* (1989), ACM Press, pp. 218–227. 1

[CKS98] CAMPAGNA S., KOBBELT L., SEIDEL H.-P.: Directed edges - a scalable representation for triangle meshes. *J. of Graphics Tools 3*, 4 (1998), 1–12. 2, 6, 8

[DGH04] DE FLORIANI L., GREENFIELDBOYCE D., HUI A.: A data structure for non-manifold simplicial d-complexes. In *Proc. ACM/Eurographics Symp. on Geometry Processing* (Nice (France), 8–10 July 2004), Kobbelt L., Schroder P.,, Hoppe H., (Eds.), ACM Press. 1, 3, 4

[DH03] DE FLORIANI L., HUI A.: A scalable data structure for three-dimensional non-manifold objects. In *Proc. ACM/Eurographics Symp. on Geometry Processing* (Aachen (Germany), 23–25 June 2003), Kobbelt L., Schroder P.,, Hoppe H., (Eds.), pp. 73–83. 2, 5, 6

[DH04] DE FLORIANI L., HUI A.: Update operations on 3d simplicial decompositions of non-manifold objects. In *9th ACM Symp. on Solid Modeling and Appl.* (Genova (Italy), 9–11 June 2004), Fellner D., (Ed.), ACM Press, pp. 169–180. 6

[DH05] DE FLORIANI L., HUI A.: *A survey on data structures for simplicial complexes*. Tech. rep., Computer and Information Science Dept. (DISI), University of Genova, Italy, May 2005. 5, 6, 7, 9

[DKP04] DE FLORIANI L., KOBBELT L., PUPPO E.: A survey on data structures for level-of-detail models. In *Multi-resolution in Geometric Modeling*, Dodgson N., Floater M.,, Sabin M., (Eds.). Springer Verlag, 2004, pp. 49–74. 1

[DL89] DOBKIN D., LASZLO M.: Primitives for the manipulation of three-dimensional subdivisions. *Algorithmica 5*, 4 (1989), 3–32. 1

[DMMP03] DE FLORIANI L., MESMOUDI M. M., MORANDO F., PUPPO E.: Non-manifold decompositions in arbitrary dimensions. *CVGIP: Graphical Models 65*, 1/3 (2003), 2–22. 1, 3, 6

[DMPS04] DE FLORIANI L., MAGILLO P., PUPPO E., SOBRERO D.: A multi-resolution topological representation for non-manifold meshes. *Computer-Aided Design Jour. 36*, 2 (February 2004), 141–159. 2, 6, 7

[Ede87] EDELSBRUNNER H.: *Algorithms in Combinatorial Geometry*. Springer Verlag, Berlin, 1987. 1, 3, 4

[GS85] GUIBAS L., STOLFI J.: Primitives for the manipulation of general subdivisions and computation of Voronoi diagrams. *ACM Trans. on Graphics 4*, 2 (April 1985), 74–123. 1

[JLM02] JOY K. I., LEGAKIS J., MACCRACKEN: Data structures for multi-resolution representation of unstructured meshes. In *Hierarchical Approximation and Geometric Methods for Scientific Visualization*, G. Farin H. Hagen B. H., (Ed.). Springer Verlag, Heidelberg, 2002. 1

[Lie91] LIENHARDT P.: Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Computer Aided Design 23*, 1 (1991), 59–82. 1

[LL01] LEE S. H., LEE K.: Partial-entity structure: a fast and compact non-manifold boundary representation based on partial topological entities. In *Proc. Sixth ACM Symp. on Solid Modeling and Appl.* (Ann Arbor, Michigan, June 2001), ACM Press, pp. 159–170. 2, 6, 7

[LT97] LOPES H., TAVARES G.: Structural operators for modeling 3-manifolds. In *Proc. Fourth ACM Symp. on Solid Modeling and Appl.* (May 1997), ACM Press, pp. 10–18. 1

[Man87] MANTYLA M.: *An Introduction to Solid Modeling*. Computer Science Press, 1987. 1, 8

[McM00] MCMAINS S.: *Geometric Algorithms and Data Representation for Solid Freeform Fabrication*. PhD thesis, University of California at Berkeley, 2000. 2, 6, 7

[MP78] MULLER D. E., PREPARATA F. P.: Finding the intersection of two convex polyhedra. *Theoretical Computer Science 7* (1978), 217–236. 1

[Nie97] NIELSON G. M.: Tools for triangulations and tetrahedralizations and constructing functions defined over them. In *Scientific Visualization: overviews, Methodologies and Techniques*, Nielson G. M., Hagen H.,, Müller H., (Eds.). IEEE Computer Society, Silver Spring, MD, 1997, ch. 20, pp. 429–525. 1, 3

[PBCF93] PAOLUZZI A., BERNARDINI F., CATTANI C., FERRUCCI V.: Dimension-independent modeling with simplicial complexes. *ACM Trans. on Graphics 12*, 1 (January 1993), 56–102. 1, 3

[PH97] POPOVIC J., HOPPE H.: Progressive simplicial complexes. In *ACM Computer Graphics Proc. Annual Conference Series, (SIGGRAPH '97)* (1997), ACM Press, pp. 217–224. 6

[RSS01] ROSSIGNAC J., SAFONOVA A., SZYMCZAK A.: 3D compression made simple: Edge-Breaker on a Corner Table. In *Proc. Shape Modeling International 2001* (Genova, Italy, May 2001), IEEE Computer Society. 1

[Wei88] WEILER K.: The radial-edge data structure: a topological representation for non-manifold geometric boundary modeling. In *Geometric Modeling for CAD Appl.*, J. L. Encarnacao M. J. Wozny H. W. M., (Ed.). Elsevier Science Publishers B. V. (North–Holland), Amsterdam, 1988, pp. 3–36. 6, 7