Shape Representations Based on Simplicial and Cell Complexes

L. De Floriani[†1,2] and A. Hui[‡2]

[1] Dept. of Computer Science, University of Genova (Italy)
[2] Dept of Computer Science, University of Maryland at College Park (USA)

Abstract

Simplicial and cell complexes are the most common way to discretize 3D shapes and two-, three and higher-dimensional scalar fields. In this state-of-the-art report, we review, analyze and compare data structures for simplicial and cell complexes. We first classify such representations, based on the dimension of the complexes they can encode, into dimension-independent, and dimension-specific ones. We further classify the data structures in each group according to the basic types of topological entities they represent. We present a description of each data structure in terms of the entities and topological relations it encodes, and we evaluate it based on its expressive power, on its storage cost, on the efficiency in supporting navigation inside the complex (i.e., in retrieving topological relations not explicitly encoded in the data structure). We also discuss a decomposition approach to modeling non-manifold shapes, which has led to powerful and scalable representations.

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

1. Introduction

Simplicial and cell complexes are the most common way to discretize geometric shapes, such as static and dynamic 3D objects, or surfaces and hyper-surfaces describing the behavior of scalar or vector fields. Representations for simplicial and cell complexes are at the heart of modeling and simulation tools in a variety of application domains, including computer graphics, Computer Aided Design (CAD), Computer Aided Engineering (CAE), finite element analysis, animation, scientific visualization, and geographic data processing.

Historically, data structures for representing 3D shapes have been developed in solid modeling. The most common representations for 3D objects are boundary representations. A *boundary representation* consists of a description of a 3D object in terms of its bounding surfaces, which are decomposed into a collection of faces, edges and vertices forming a cell complex. The first data structure for boundary representation is the *Winged-Edge data structure*, proposed by Baumgardt in 1972 [Bau72]. This has been only the starting point for the development of a variety of representations, which are at the basis of current solid modeling systems [Man87].

Triangle meshes have been used for a long time in finite element analysis and in terrain modeling. Specific data structures for triangle meshes have been developed as well as a variety of algorithms for generating and updating such meshes. Triangles are also the basic primitive handled by graphics hardware, and, thus, triangle meshes have become the most common way of discretizing 3D surfaces and scenes, as well as one of the most common exchange format for such shapes.

Both boundary representations and triangle meshes discretize surfaces which can either be closed (without boundary) or be the graphs of functions of two variables (i.e., 2D scalar fields) as in the case of terrain models. Such surfaces have a simple topological structure. On the other hand, several applications, such as CAD/CAE, finite element simulation, require modeling objects with a more complex topol-

† e-mail:deflo@disi.unige.it
‡ e-mail:huiannie@cs.umd.edu

© The Eurographics Association 2007.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

ogy, including singularities and parts of different dimensionalities, the so-called *non-manifold* and *non-regular* objects. Recall that 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 contain parts of different dimensionalities, are called *non-regular*. Non-manifold and non-regular objects are discretized as two-dimensional cell and simplicial complexes. Data structures have been developed for both kinds of complexes as well as operators for manipulating them. Some of such data structures are based on a decomposition of a non-manifold shape into simple manifold or nearly manifold components.

Several applications, like solid modeling, simulation and scientific visualization, require a shape, or the domain of a scalar field to be discretized as a three-dimensional simplicial complex. Thus, representations and update operators have been developed for such complexes. Finally, data structures for encoding arbitrary dimensional simplicial complexes have been proposed, mainly in the computational geometry literature, as the basis for a dimension-oriented approach to the design and implementation of a data structure for a cell or a simplicial complex.

This state-of-the-art report reviews, analyzes and compares data structures for simplicial and cell complexes used for modeling 3D shapes and scalar fields. The comparison is performed in terms of their expressive power, of their storage cost, and of the efficiency and effectiveness of navigation operations on them.

The remainder of this paper is organized as follows. Section 2 provides some background notions on cell and simplicial complexes. Section 3 gives a formalization of topological relations. Section 4 presents a taxonomy of data structures for cell and simplicial complexes. Section 5 reviews and compares dimension-independent data structures. Section 6 reviews and compares data structures specific for two-dimensional complexes. Due to the large number of data structures in this category, we distinguish between data structures for manifold and non-manifold cell complexes, addressed respectively in Subsections 6.1 and 6.2. Section 7 reviews and compares data structures for three-dimensional cell and simplicial complexes. Section 8 addresses shape representations that are based on the decomposition of a 3D shape into parts of lower complexity. Finally, section 9 draws some concluding remarks.

2. Background Notions

In this Section, we review some notions on cell and simplicial complexes, that we will use throughout this report (see [Ago05] for more details).

Intuitively, a Euclidean cell complex is a collection of basic elements, called *cells*, which cover a domain in the Euclidean space. A *k-dimensional cell* (or simply a *k-cell*) γ in the Euclidean space E^n, $1 \leq k \leq n$, is a subset of E^n homeomorphic to a closed k-dimensional ball $B^k = \{x \in R^k : ||x|| \leq 1\}$. ($||x||$ denotes the norm of vector x.) A 0-cell is a point in R^n. k is called the *order*, or *dimension*, of k-cell γ.

A *(Euclidean) cell complex* is a finite set Γ of cells of dimension at most d in E^n, $0 \leq d \leq n$, such that the interiors of the cells of Γ are disjoint, and if $\gamma, \gamma_1 \in \Gamma$, such that $\gamma \cap \gamma_1 \neq \emptyset$, then $\gamma \cap \gamma_1$ is the disjoint union of interiors of cells of Γ. A cell complex Γ such the maximum dimension of its cells is equal to d is called a *d-dimensional complex*, or simply a *d-complex*. The *domain*, or *carrier*, of a Euclidean cell d-complex Γ 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 cells in Γ.

The *(combinatorial) boundary* of a cell γ in a cell complex Γ is the set of all cells in Γ, which are subsets of the boundary of cell γ (considered as a point set). Every cell in $b(\gamma)$ is called a *face* of γ. If k is the dimension of the face, it is called a *k-face*. The *co-boundary*, or *star*, of a cell γ, is the collection of all the cells in Γ containing γ in its boundary. The *link* of a cell γ is defined as the collection of the cells bounding the cells in the star of γ, which do not contain γ. A cell is called a *top cell* if it is not contained in the boundary of any other cell in Γ.

Two cells are called *k-adjacent* if they share a k-face. Two p-cells, $0 < p \leq d$, 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*, $0 \leq h \leq d-1$, is a sequence of $(h+1)$-cells $(\gamma_i)_{i=0}^k$ such that two consecutive cells γ_{i-1} and γ_i in the sequence are h-adjacent. Two cells γ and γ^* are said to be *h-connected* if there exists an h-path $(\gamma_i)_{i=0}^k$ such that γ is a face of γ_0 and γ^* is a face of σ_k. A complex Γ^* is called *h-connected* if and only if any two cells of Γ^* are h-connected.

A d-complex Γ, in which all top cells are d-cells, is called *regular* (or *uniformly d-dimensional*). A regular $(d-1)$-connected d-complex in which each $(d-1)$-cell is shared by one or two d-cell is called a *(combinatorial) pseudo-manifold* (possibly with boundary). A pseudo-manifold complex whose domain is a manifold is called a *manifold complex*. Figure 1(a) shows an example of a regular complex, which is not a pseudo-manifold, while Figure 1(b) and (c) show an example of a pseudo-manifold complex which does not have a manifold domain.

Simplicial complexes can be seen as a subclass of the cell complexes. Their cells, called *simplexes*, are defined by the convex combination of points in the Euclidean space. 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. Any Euclidean p-simplex σ', with $0 \leq p < k$, generated by a set $V_{\sigma'} \subseteq V_\sigma$ of cardinality $p+1 \leq d$, is called a *p-face* of σ.

© The Eurographics Association 2007.

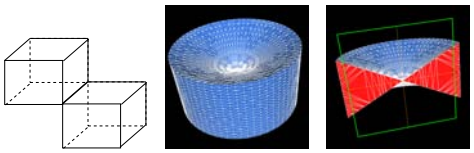


Figure 1: *(a) A regular cell complex that is not manifold; (b) A pseudo-manifold with a non-manifold domain (a 3D pinched pie) and (c) shows the cross-section of the pinched pie at the non-manifold point*

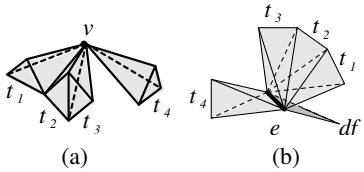


Figure 2: *(a) A non-manifold vertex v; (b) A non-manifold edge e*

Whenever no ambiguity arises, the dimension of σ' will 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.

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 complex is defined in the same way as for a cell complex. Since a simplicial complex is a cell complex, all the properties of cell complexes are inherited by simplicial complexes.

We characterize the non-manifold singularities in the combinatorial representation of a non-manifold shape by defining non-manifold vertices and edges in its discretization as a cell complex. A vertex (0-cell) v in a cell (simplicial) d-complex Γ (with $d \geq 1$) is a *manifold* vertex if and only if the link of v in Γ is homeomorphic to a triangulation of the $(d-1)$-sphere S^{d-1}, or of the $(d-1)$-disk B^{d-1}. A vertex is called *non-manifold* otherwise. Figure 2 (a) shows an example of a non-manifold vertex. An edge (1-cell) e in a d-complex Γ (with $d \geq 2$) is a *manifold* edge if and only if the link of e in Γ is homeomorphic to a triangulation of the $(d-2)$-sphere S^{d-2}, or of the $(d-2)$-disk B^{d-2}. An edge is called *non-manifold* otherwise. Figure 2 (b) shows an example of a non-manifold edge. In general, a k-cell γ is a d-complex Γ (with $d \geq k+1$) is a *manifold* k-cell if and only if the link of γ in Γ is homeomorphic to a triangulation of the $(d-k)$-sphere S^{d-k} or of the $(d-2)$-disk B^{d-k}. It is called *non-manifold* otherwise.

3. Topological Relations

The connectivity information among the entities in a cell or in a simplicial complex are expressed through *topological relations*, which provide an effective framework for defining, analyzing and comparing the wide spectrum of existing data structures. Data structures for cell and simplicial complexes can be described formally in terms of the topological entities and relations they encode. We define topological relations for the case of a cell complex (since a simplicial complex can be seen as a special case of a cell complex).

We consider a cell d-complex Γ and a cell $\gamma \in \Gamma$, with $0 \leq p \leq d$. We can define *topological relations* as follows:

- *Boundary relation* $R_{p,q}(\gamma)$, with $0 \leq q \leq p-1$, consists of the set of q-cells which are faces of γ.
- *Co-boundary relation* $R_{p,q}(\gamma)$, with $p+1 \leq q \leq d$, consists of the set of q-cells incident in γ.
- For $p>0$, *adjacency relation* $R_{p,p}(\gamma)$ consists of the set of p-cells in Γ that are $(p-1)$-adjacent to γ.
- Relation $R_{0,0}(\gamma)$, where γ is a vertex, consists of the set of vertices that are adjacent to γ through a 1-cell.

Note that both boundary and co-boundary relations are called *incidence relations*.

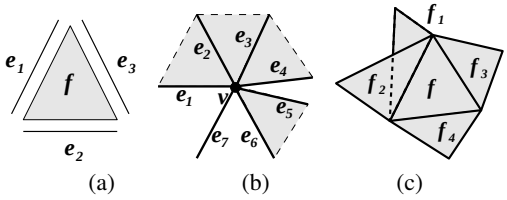


Figure 3: *Example of topological relations (a) boundary relation $R_{2,1}(f) = \{e_1, e_2, e_3\}$ for face f, (b) co-boundary relation $R_{0,1}(v) = \{e_1, \cdots, e_7\}$ for vertex v, and (c) adjacency relation $R_{2,2}(f) = \{f_1, \cdots, f_4\}$ for face f*

We call *constant* any relation which involves a constant number of entities. Relations which involve a variable number of entities are called *variable*. Co-boundary and adjacency relations are variable relations in general. Boundary relations are constant in simplicial complexes. Thus, we consider an algorithm for retrieving a topological relation R to be *optimal* if it retrieves a given relation R in time linear in the number of entities involved in R.

Depending on the amount of information encoded, data structures for cell and simplicial complexes may support the retrieval of topological relations according to various degree of efficiency. If the retrieval of a relation requires examining the star of all the cells adjacent to or on the boundary of the query cell, we say that the data structure offers a *sub-optimal* support for the retrieval of that relation. If the retrieval a relation from a data structure requires examining all cells of a

© The Eurographics Association 2007.

specific dimension, then the data structure does not support efficient retrieval of that relation.

4. A Taxonomy for Data Structures for Cell and Simplicial Complexes

We can first classify the data structures for cell and simplicial complexes in terms of:

1. *the domain* of the complexes represented: manifold, pseudo-manifold, regular, etc..
2. *the dimension*: *dimension-independent* data structures can describe cell and simplicial complexes in any dimension, while *dimension-specific* data structures are for 2D and 3D cell and simplicial complexes embedded in the three-dimensional Euclidean space.
3. *the type of topological information encoded*: in a cell (or simplicial) complex, the basic topological elements are the cells (simplexes). A data structure may encode all the cells of a complex, or only a subset of it.
4. *the way topological information is encoded*: some data structures encode the cells and their topological relations *explicitly*. In such data structures, the cells are *entities* and the relations are associated with the entities. *Implicit* data structures encode the relations among cells indirectly, through tuples of cells in the same relation.

Explicit data structures can be further classified into *incidence-based*, and *adjacency-based* representations. Incidence-based data structures encode all cells in a complex and a suitable subset of incidence relations. Adjacency-based data structures generally encode only top cells (i.e., cells which are not on the boundary of other cells) and vertices, and adjacency relations among them plus possibly a suitable subset of co-boundary relations. A further category exists for data structures for simplicial and cell 2- and 3-complexes and consists of *edge-based* data structures in the 2D case, and *face-based* data structures in the 3D case.

Data structures that are designed for cell complexes can be used for simplicial complexes. In some cases, specializations of such data structures have been developed by taking advantage of the properties of simplicial complexes.

**** ANNIE: if it is too long, we can cut the paragraph below

In what follows, we organize the description of the various representations on the basis of the dimension of the complex 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 space requirements, and on the efficiency in supporting navigation inside the complex (i.e., in retrieving topological relations not explicitly encoded). The space requirements are expressed throughout this report in terms only of number of items of topological information encoded, since we assume that all the data structures encode the same geometrical information. This is also gives an evaluation which is independent of the specific implementation. We compare the various data structures inside each category based on the above features and, for representations for non-manifold shapes, also based on their scalability to the manifold case. We emphasize data structures for simplicial complexes since these are the most common mesh-based models in a variety of applications.

5. Dimension-independent Representations

In this Section, we discuss dimension-independent representations for cell complexes first, and then for simplicial complexes. We review first two dimension-independent implicit representations, namely the *Cell-Tuple* [Bri89] and the *N-G-map* [Lie94] representations, and an explicit incidence-based representation, the *Incidence Graph (IG)* [Ede87]. The two implicit representations are for manifold shapes, while the latter is for non-manifold shapes as well. We then discuss data structures specific for simplicial complexes, namely the *Indexed data structure with Adjacencies (IA)* [PBCF93] (which is a d-dimensional extension of the representation discussed in [Nie97]), and the *Incidence Simplicial data structure (IS)* [DH06]. The former is an adjacency-based representation, while the latter is a simplified version of the Incidence Graph specific for simplicial complexes. The IA data structure is for pseudo-manifolds, while the IS is for arbitrary simplicial complexes.

5.1. Cell-Tuple and N-G-map

A cell-tuple [Bri89] is a representation for Euclidean cell complexes with a manifold domain, while the n-G-map [Lie94] has been developed for abstract cell complexes belonging to the class of quasi-manifolds, which is a superclass of combinatorial manifolds defined in [Lie94]. In essence, however, the cell-tuples and the n-G-maps are equivalent. Here, we describe, for brevity, only Cell-Tuple data structure.

Given a Euclidean d-dimensional cell complex, a *cell tuple* is a $(d+1)$-tuple t of $d+1$ cells, $t = (c_0, c_1, \cdots c_d)$, such that c_i is an i-cell on the boundary of cells c_{i+1} to c_d. A function s_i for $i = 0..d$, called a *switch function*, is defined on the cell-tuples such that $t' = s_i(t)$ if the cell tuple t' is identical to t in every element except the i-th one. The s_i functions partition the set of cell-tuples into equivalent classes of size 2 each. The s_i functions have the following two properties:

- For $i = 0, \ldots, d$, s_i is an involution. That is, given a cell tuple t, $s_i(s_i(t)) = t$;
- For $i = 0, \ldots, d-2$ and $i+2 \leq j \leq d$, $s_i s_j$, where $s_i s_j(t) = s_j(s_i(t))$, is an involution. That is, $s_i s_j(s_i s_j(t)) = t$.

Figure 4(a) gives a simple example of a cell complex defined on a surface without boundary. The cell complex is

© The Eurographics Association 2007.

composed of triangle A, square B and the face C that covers the remainder of the surface. Figure 4(b) shows all the tuples in small squares, and all the s_i $(i = 0,1,2)$ functions. Two tuples are related by function s_0 if they are connected through a dotted line, by s_1 if connected by a thin solid line, or by s_2 if connected by a dashed line.

The Cell-Tuple data structure encodes all cell-tuples in a complex, and the switch functions s_i for $i = 0..d$. It is an implicit representation because the cells and their relations are only implicitly represented by the cell-tuples.

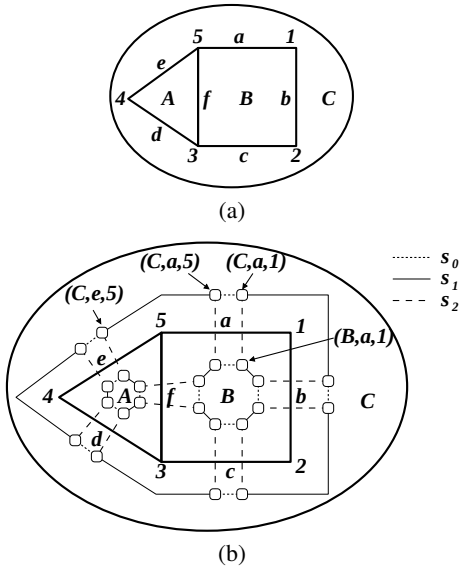


Figure 4: *(a) A simple cell complex on a surface homeomorphic to a sphere. The complex is composed of triangle A, o square B and the remaining face C on the surface; (b) all the tuples and all the switch s_i $(i = 0,1,2)$ functions encoded by the cell-tuple*

The space requirements of the Cell-Tuple data structure can be evaluated as follows. Given a d-dimensional cell complex with n_d d-cells, there are $n_d(d+1)!$ cell-tuples. The switch functions s_i are encoded as (s_i, t, t') where $s_i(t) = t'$, which consists of $n_d(d+1)(d+1)!$ pieces of information. To support topological navigation at each simplex, it is necessary to store links from each p-simplex σ to all the cell-tuples that contain of σ. This needs $n_d(d+1)!$ links and thus it results in a verbose representation.

It can be shown that all topological relations can be retrieved in optimal time from the Cell-Tuple data structure. As an example, consider, the retrieval of relation $R_{0,2}(5)$ for vertex 5 in Figure 4, which consists of all the faces that are incident at vertex 5. The retrieval starts with any of the tuples that include vertex 5, such as $(C,a,5)$. By alternately applying functions s_2 and s_1 to each new tuple visited, the cyclic sequence $(C,a,5)$, $(B,a,5)$, $(B,f,5)$, $(A,f,5)$, $(A,e,5)$, $(C,e,5)$ is obtained, which produces the set of faces $\{C,B,C\}$ that are incident at vertex 5.

5.2. Incidence Graph (IG)

The *Incidence Graph (IG)* [Ede87] is an incidence-based explicit data structure for cell complexes. The topological information captured is the set of incidence relations among cells that differ by one dimension. Formally, the IG encodes all the cells of any given cell d-complex Γ, and for each p-cell γ, its immediate boundary, and immediate co-boundary relations, namely:

- for each p-cell γ, where $0 < p \leq d$, boundary relations $R_{p,p-1}(\gamma)$,
- for each p-cell γ, where $0 \leq p < d$, co-boundary relations $R_{p,p+1}(\gamma)$

Figures 5(a)-(c) give an example that illustrates the relations encoded in the IG.

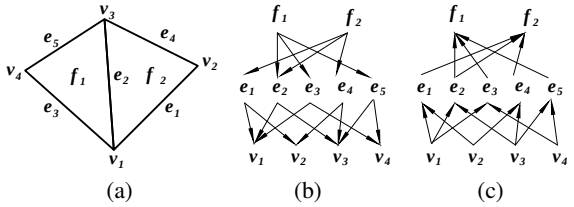


Figure 5: *(a) A simple simplicial complex formed by two triangles; (b) all the boundary relations encoded by the IG; (c) all the co-boundary relations encoded by the IG*

The design of the IG supports a simple recursive strategy to retrieve topological boundary and co-boundary relations. Boundary relation $R_{p,q}(\gamma)$ $(p > q)$ for a given p-cell γ is obtained by retrieving the encoded boundary $R_{i,i-1}$ relations of all the i-faces for $i = p, \cdots, q-1$ of γ. Co-boundary relation $R_{p,r}(\gamma)$ $(p < r)$ is obtained by retrieving the encoded co-boundary $R_{i,i+1}$ relations of all the i-cells for $i = p, \cdots, r-1$ in the star of γ. The retrieval of such relations can be done in time linear in the number of cells involved, which is thus optimal. We can evaluate the exact space requirements of the IG when it encodes a simplicial complex because each simplex has a constant number of faces. The boundary and co-boundary relations encoded by the IG for a simplicial complex amounts to

$$2 \sum_{0<p\leq d} n_p(p+1)$$

pieces of information, because each p-simplex has exactly $(p+1)$ faces of dimension $(p-1)$.

5.3. Indexed Data Structure with Adjacencies

The *Indexed data structure* is a representation for simplicial d-complexes which encodes, for each top k-simplex σ, relation $R_{k,0}(\sigma)$, i.e., the indexes to its $(k+1)$ vertices. Only

© The Eurographics Association 2007.

boundary relations of type $R_{k,j}(\sigma)$, $j < k$, for any top k-simplex σ, can be extracted in optimal time from such representation. Note that the j-simplexes on the boundary of σ are described through $j+1$ vertex indexes.

The *Indexed data structure with Adjacencies (IA)*, also called *winged representation* [Nie97, PBCF93] extends the indexed data structure into a topological data structure, which also encodes adjacency information among the simplexes. This restricts its representation domain to pseudo-manifolds. The IA data structure encodes, for each d-simplex σ in a simplicial complex Σ:

- relation $R_{d,0}(\sigma)$, i.e., the indexes of its $(d+1)$ vertices;
- relation $R_{d,d}(\sigma)$, i.e., the indexes of the $(d+1)$ d-simplexes sharing a $(d-1)$-face with σ.

Only boundary relations, as in the indexed data structure, plus relation $R_{d,d}$ can be retrieved in optimal time from the IA data structure. Vertex-based co-boundary relations can retrieved in optimal time from an extension of the IA data structure. This is achieved by encoding, for each vertex v, a partial version of relation $R_{0,d}(v)$, that we denote $R^*_{0,d}(v)$, i.e., one d-simplex for each connected component of the link of v.

Figure 6 gives an example of the retrieval of the complete $R_{0,3}(v)$ relation for vertex v from the encoded partial $R^*_{0,3}(v) = \{t_1, t_4\}$ relation. From v, t_1 is accessible through $R^*_{0,3}(v)$. t_2 is accessible through the $R_{3,3}(t_1)$ relation of t_1. Similarly, the tetrahedra t_3 and t_5 in the star of v are retrievable first by extracting t_4 from $R^*_{0,3}(v)$ and then by retrieving $R_{3,3}(t_4)$.

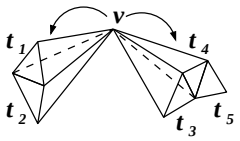


Figure 6: *Example of the retrieval of the $R_{0,3}(v)$ relation from the encoded partial $R^*_{0,3}(v) = \{t_1, t_4\}$ relation in the IA data structure*

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 $0 \le k \le d$, can be retrieved in time linear in the number of d-simplexes in the star of v. For $d \le 3$, $R_{0,k}(v)$ is optimal. Retrieval of all $R_{q,k}(\sigma)$ relations for $0 < q < k < d-1$ requires traversing the star of each of the vertices of σ and thus it takes time linear in the number of d-simplexes incident at the vertices of σ. Thus, such algorithms are still local, but sub-optimal.

The storage cost of the IA data structure with this extension is equal to $2n_d(d+1)+n_0$ items for a simplicial complex with n_d d-simplexes and n_0 vertices, only n_0 items more with respect to storing just the $R_{d,0}$ and $R_{d,d}$ relations. For a manifold simplicial 2-complex, this leads to $6n_2+n_0$, which is approximately $13n_0$ as a consequence of Euler' formula.

5.4. The Incidence Simplicial Data Structure

The *Incidence Simplicial (IS) Data Structure* [DH06] is an improved version of the Simplified Incidence Graph [DGH04] which simplifies the IG for encoding simplicial complexes. It encodes all simplexes in a simplicial complex Σ as well as the following topological relations:

- for each p-simplex σ, where $0 < p \le d$, boundary relations $R_{p,p-1}(\sigma)$,
- for each p-simplex σ, where $0 \le p < d$, partial co-boundary relations $R^*_{p,p+1}(\sigma)$, which is defined as follows: $R^*_{p,p+1}(\sigma)$ consists of one arbitrarily-selected $(p+1)$-simplex, for each connected component in the link $lk(\sigma)$ of σ. In the example of Figure 7(a), t_1, t_2 and t_3 form a connected component, and t_4 is a single component in the link of v shown in Figure 7(b). Relation $R^*_{0,1}(v) = \{e_1, e_2\}$.

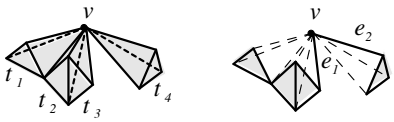


Figure 7: *(a) The star of v (b) The link of v shown in shaded area. The partial co-boundary $C^*_{p,p+1}(v)$ relation consists of one edge for each connected component in the link of v*

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 have one element with the exception of $R^*_{d-1,d}(\sigma)$, which consists of at most two d-simplexes. Note also that the IS encodes the same boundary relations as the IG.

The storage cost of the IS for encoding a simplicial complex is:

$$\sum_{0<p\le d} n_p(p+1) + \sum_{0\le p<d} k_p,$$

where n_p is the number of p-simplexes in Σ and k_p is the total number of connected components at all p-simplexes of Σ. For the case of a manifold simplicial 2-complex, there is exactly one component at each vertex (ie., $k_0 = n_0$) and the total number of components at all the edges is equal to $3n_2$ (i.e., $k_1 = 3n_2$). Therefore, the storage cost of the IS is at most equal to $6n_2 + 2n_1 + n_0$, which is approximately $19n_0$, because of Euler formula.

5.5. Comparisons

Table 1 summarizes the comparison among the various dimension-independent data structures in terms of their domain, of the complexes they can describe, and of the representation method.

We summarize the space requirements of the above data

© The Eurographics Association 2007.

Data Structure	Domain	Complexes	Method
Cell-Tuple	Manifold	Cell	Implicit
IG	Non-Manifold	Cell	Incidence-based
IA	Manifold	Simplicial	Adjacency-based
IS	Non-Manifold	Simplicial	Incidence-based

Table 1: *Data structures for d-dimensional complexes*

structures, for manifold and for arbitrary simplicial complexes. These costs are expressed throughout this report only in terms of items of topological information encoded. The storage costs of the cell-tuple, of the IG and of the IA, and IS data structures for a manifold d-dimensional simplicial complex are as follows (they are expressed in terms of the number of q-simplexes, denoted as n_q):

- Cell-Tuple: $n_d(d+1)(d+1)!+n_d(d+1)!$
- IA: $2(d+1)n_d+n_0$
- IG: $2\sum_{0<p\leq d} n_p(p+1)$
- IS: $\sum_{0<p\leq d} n_p(p+1)$ + $2n_{d-1}$+$\sum_{0\leq p<d-1} n_p$

For arbitrary d-dimensional simplicial complexes, we can only report the storage costs of the IG and of the IS data structures, since the other two are for restricted classes of complexes. In the following, k_q denotes the total number of connected components at all the links of the q-simplexes of the complex.

- IG: $2\sum_{0<p\leq d} n_p(p+1)$
- IS: $\sum_{0<p\leq d} n_p(p+1)$ + $\sum_{0\leq p<d} k_p$

We summarize in Table 2 the navigation costs by evaluating the optimality of algorithms for retrieving topological relations on the various representations.

Data Structure	Boundary relations	Co-boundary relations	Adjacency relations
Cell-Tuple	Optimal	Optimal	Optimal
IG	Optimal	Optimal	Optimal
IA	Optimal	$R_{0,k}$: optimal Others: sub-optimal	$R_{d,d}$: optimal Others: sub-optimal
IS	Optimal	Sub-optimal	Sub-optimal

Table 2: *Navigation efficiency of data structures for d-dimensional complexes*

6. Representations for Two-dimensional Cell and Simplicial Complexes

In this Section, we discuss representations for two-dimensional cell and simplicial complexes embedded in the 3D Euclidean space. We classify them according to the taxonomy introduced in Section 4, and organize their description in two subsections according to the domain of the complexes (manifold or non-manifold). We perform the comparison based on their space requirements and on their efficiency in retrieving topological relations. We evaluate and compare the non-manifold representations also based on their scalability to the manifold case. We discuss briefly those for manifold and arbitrary (non-manifold) cell complexes, and focus on representations for simplicial complexes.

6.1. Representations for Manifold 2-Complexes

We discuss here representations for cell and simplicial complexes for manifold shapes. A thorough analysis and comparison of data structures for manifold cell 2-complexes can be found in [Sam06]. Here, we briefly review the *Winged-Edge* [Bau72], the *Doubly-Connected Edge List (DCEL)* [MP78], the *Half-Edge* [Man87] the *Quad-Edge* [GS85], and the Lath-based [JLM02] data structures for manifold cell complexes, and the Star-Vertex [KT01], and the Corner Table [RSS01] data structures for manifold simplicial complexes (usually called *triangle meshes*). The Winged-Edge, DCEL, and the Half-Edge data structures are all edge-based representations, since they represent the edge as the primary entity and the relations around it. The quad-edge and the lath-based data structures are implicit representations. The Corner Table and the Star-Vertex data structures are adjacency-based representations.

6.1.1. Explicit Edge-based Data Structures for Cell 2-Complexes

The *Winged-Edge (WE) data structure* [Bau75] is historically the first one proposed for two-dimensional cell complexes. It encodes: (i) for each edge e, its two vertices, the two 2-cells (usually called *faces* in the context of boundary representations for solid objects) incident at e, and the four edges that are both adjacent to e and are on the boundary of the two faces incident at e; (ii) for each face f, a reference to one edge on the boundary of f; (iii) for each vertex v, a reference to one edge incident at v. It supports the retrieval of all topological relations in optimal time. Also the cells in the star of a vertex or on the boundary of a face can be traversed in both clockwise or counterclockwise directions. Given a cell 2-complex with n_2 faces, n_1 edges and n_0 vertices, the Winged-Edge data structure stores n_2+$8n_1$+n_0 pieces of topological information.

The *Doubly-Connected Edge List (DCEL) data structure* [MP78] is a simplified version of the WE representation. For each edge e, instead of encoding all four edges on the boundary of the two faces incident at e, it stores only two edges, one for each of the two faces incident in e. This data structure supports the traversal of all topological relations, but only in counterclockwise direction in the star of a vertex, and in clockwise direction around the boundary of a face. n_2+$6n_1$+n_0 pieces of topological information are encoded in the DCEL data structure.

© The Eurographics Association 2007.

The *Half-Edge (HE) data structure* [Man87] encodes two copies of each edge, each of which is called a half-edge. A half-edge has a direction with respect to the face to which it bounds. For each half-edge, the following information is encoded: its start vertex, the face associated with it, the previous and the next edges on the same face, the companion half-edge. Relations encoded at vertices and faces are the same as those encoded in the Winged-Edge and DCEL data structure. The Half-Edge data structure supports the retrieval of all topological relations in optimal time, and also the cells in the star of a vertex or on the boundary of a face can be traversed in both clockwise or counterclockwise directions. There are n_2+10n_1+n_0 pieces of topological information encoded in the HE data structure. An implementation of the HE data structure which has the same storage cost as the WE data structure is described in [Sam06].

The edge-based relations encoded in each of the edge-based data structures described are illustrated in Figures 8(a)-(c). All the edge-based data structures presented in this section encode, for each edge, relations $R_{1,0}$ and $R_{1,2}$, different partial $R^*_{1,1}$ relations, since only two or four edges are encoded, a partial $R^*_{0,1}$ relation for each vertex, which consists of one edge in the star of the vertex, and a partial $R^*_{2,1}$ relation for each face, which consists of one edge on the boundary of the vertex. All these data structures support the retrieval of all topological relations in optimal time.

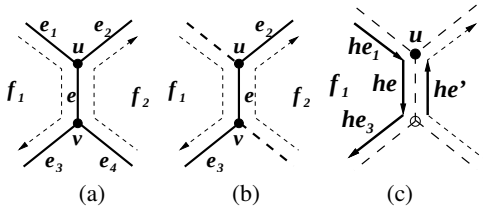


Figure 8: *Edge-based relations represented in the edge-based data structures: (a) In the Winged-Edge data structure, e has a reference to $e_1, e_2, e_3, e_4, u, v, f_1$ and f_2. (b) In the DCEL, e has a reference to e_2, e_3, u, v, f_1 and f_2. (c) In the Half-Edge data structure, each edge e is represented as two half-edges he and he. Half-edge he has a reference to he', he_2, he_3, u and f_1*

6.1.2. Quad-Edge and Lath-based Data Structures

The *Quad-Edge data structure* [GS85] and the *Lath-based data structures* [JLM02] are implicit data structures for cell 2-complexes with a manifold domain. In such complexes, edges in the star of each vertex can be ordered radially on a plane around the vertex, and the edges on the boundary of a face can be ordered clockwise or counterclockwise around the face. Thus, each edge belongs to four loops: the two at its extreme vertices, and the two at the faces sharing it. All these representations exploit these property.

In the Quad-Edge data structure, each *quad-edge* is associated with its two extreme vertices, its two adjacent faces and the next edges in its four loops. Essentially, the Quad-Edge data structure encodes the same information as the Winged-Edge data structure. In a quad-edge that corresponds to edge e, the four adjacent edges of e are organized as part of the two loops around two faces, and two loops around two vertices. In the Winged-Edge data structure, the same four edges belong to the two loops of the two faces. Same relations at vertices and faces are encoded as in the Winged-Edge data structure. As the edge-based data structures presented in Subsection 6.1.1, the Quad-Edge data structure encodes partial relations $R^*_{2,1}$ for each face, partial relation $R^*_{0,1}$ for each vertex, complete relations $R_{1,0}$, $R_{1,2}$ and a partial version of relation $R_{1,1}$ for each edge. As in the other edge-based representations, all topological relations can be retrieved in optimal time. The storage cost of the Quad-Edge data structure is n_2+8n_1+n_0.

The *Lath-based* data structures are a collection of data structures that use vertices and *laths* as the basic elements. Each lath is uniquely identified with exactly one vertex, one edge and one face of a complex. A lath is conceptually similar to a cell-tuple. A Lath-based data structure requires no separate records for edges and faces. There are three variations in the encoding of a lath, giving rise to three data structures: the *Split-Edge*, the *Half-Edge-Lath* and the *Corner* data structures. In the *Split-Edge* data structure, each lath *se* corresponds to one side of an edge and encodes a link to its start vertex u, a link to the lath se' of the other side of the same edge, and a link to the lath se_3 of the next edge in the clockwise direction on the same face (see Figure 9(a)). In the *Half-Edge-Lath* data structure, each lath *he* (illustrated in Figure 9(b)) is associated with half of an edge. It encodes a link to its vertex u, a link to the lath he' of the other half of the same edge, and a link to the lath he'_1 of the next edge in the clockwise direction around the same vertex. Joy et al. called this version of the lath-based data structure the *Half-Edge* data structure, but the edge is halved differently from that of the Half-Edge data structure described above [Man87], and we call it the Half-Edge-Lath to distinguish it from the latter. In the *Corner* data structure, a lath is associated with one corner of a vertex. Each lath u' encodes: a link to the vertex u, a link to the lath v' of the next vertex v in the clockwise direction on the same face, and a link to the lath u''' of the next face in the clockwise direction around the same vertex. All Lath-based data structures support the retrieval of all topological relations through laths in optimal time. However, because of the implicitness of the faces, edges and vertices, access from these cells to their associated laths is not time-efficient. All Lath-based data structures have the same storage costs, which is $6n_1$, because each lath stores three pieces of topological information and the total number of laths is $2n_1$ in each case elaborated as follows. For the Split-Edge and Half-Edge-Lath structures, every edge corresponds to exactly two split-edge laths, exactly

© The Eurographics Association 2007.

two half-edge laths, respectively. In the Corner structure, it can be observed that the number of corners at each vertex is equal to the degree of the vertex (the number of edges incident at that vertex), while each edge is incident at exactly two vertices.

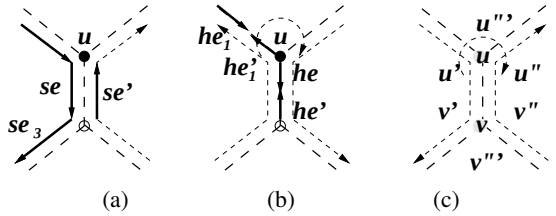


Figure 9: *(a) Split-Edge Lath; (b) Half-Edge Lath; (c) Corner Lath*

6.1.3. The Star-Vertex Data Structure

The *Star-Vertex data structure* [KT01] is an adjacency-based data structure for manifold simplicial 2-complexes. The basic entity here is the vertex. For each vertex v, the Star-Vertex data structure encodes all the vertices in the link of v in counterclockwise order. For each vertex v' in the link of v, the data structure encodes a reference to the position of vertex v'' in the link of v', such that v, v', v'' are in the same triangle.

In terms of topological relations, we can say that the Star-Vertex data structure encodes relation $R_{0,0}$ explicitly and relation $R_{2,0}$ implicitly. It only supports the retrieval of boundary relations and relations $R_{0,0}$ in optimal time. Co-boundary relations cannot be retrieved locally. The Star-Vertex data structure encodes $6n_2$ pieces of information for a manifold simplicial 2-complex with n_2 triangles, as the sum of the number of neighbors at all vertices, $\sum_v deg(v)$ is equal to twice the number of edges, and for each neighbor, two pieces of information (i.e., the neighbor vertex, and the next vertex on face) are encoded. Based on Euler' formula, this is approximately equal to $6n_2$.

6.1.4. The Corner Table (CoT) Data Structure

The *Corner Table (CoT) data structure* [RSS01] is an adjacency-based data structure for manifold simplicial 2-complexes. A *corner* is a unique index that is assigned to a triangle-vertex pair. It encodes the following information:

- For each triangle t, its three vertices a, b, c;
- For each corner c of triangle t, let e be the edge of t that is opposite to c. Then the opposite corner of the triangle that shares e is associated to c.

Formally, the Corner-Table data structure encodes the complete $R_{2,2}$ and $R_{2,0}$ relations, and the $R_{0,2}$ relation partially. All topological relations can be retrieved from the Corner-Table in optimal time. The total amount of encoded information is equal to $6n_2$, of which $3n_2$ accounts for the vertices of the triangles, and $3n_2$ accounts for the opposite corner of each corner.

6.1.5. Comparisons

We compare the data structures for manifold 2-complexes in terms of their characteristics, their space requirements and efficiency in supporting the retrieval of topological relations. Table 3 summarizes the characteristics of the various data structures in terms of the complexes they represent and of their representation method.

Data Structure	Domain	Complexes	Method
Winged-Edge	Manifold	Cell	Edge-based
DCEL	Manifold	Cell	Edge-based
Half-Edge	Manifold	Cell	Edge-based
Quad-Edge	Manifold	Cell	Implicit
Lath	Manifold	Cell	Implicit
Star-Vertex	Manifold	Simplicial	Adjacency-based
Corner Table	Manifold	Simplicial	Adjacency-based

Table 3: *Characteristics of the data structures for manifold 2-complexes*

The storage cost of each data structure is evaluated based on the topological information encoded. We also consider the two-dimensional instances of the dimension-independent data structures, except for the Cell-Tuple data structure, which is the dimension-independent generalization of the Quad-Edge data structure. The storage costs of these data structures for a cell 2-complex with n_2 faces, n_1 edges and n_0 vertices, are listed below. From Euler' formula, we have that $n_2 \leq 2n_0$ and $n_1 \leq 3n_0$. Thus, for the sake of comparison, we can express all the storage costs in terms of the number of vertices.

- Winged-Edge: $n_2 + 8n_1 + n_0 \approx 27n_0$
- DCEL: $n_2 + 6n_1 + n_0 \approx 21n_0$
- Half-Edge: $n_2 + 10n_1 + n_0 \approx 33n_0$
- Quad-Edge: $n_2 + 8n_1 + n_0 \approx 27n_0$
- Laths: $6n_1 \approx 18n_0$
- IG: $8n_1 \approx 24n_0$

The storage costs of these data structures are evaluated for six data sets of manifold cell-complexes and reported in Table 4. The Laths is the most compact data structures for manifold cell 2-complexes, followed by the Incidence Graph. The compactness of the lath-based data structures is achieved, however, at the expense of the fact that access from vertices, edges and faces to their associated laths is not efficient. The IG, which is 1.33 times the size of the Laths data structures, on the other hand, explicitly represents all these entities. Explicit edge-based data structures generally are less space-efficient. Among them, the Half-Edge data structure has the largest space requirements, which is 1.8 times that of the Laths data structures.

The storage costs of the data structures for manifold simplicial 2-complexes and of the two dimensional instances of the IA and the IS data structures are:

© The Eurographics Association 2007.

Data set	n_0	n_1	n_2	deg(V)	deg(F)
Football 1	1232	2340	1110	3.80	4.22
Football 2	930	1500	572	3.23	5.24
Crumb	312	564	254	3.62	4.44
Multidode	80	150	72	3.75	4.17
Torus	10.2k	20.5k	10.2k	4.00	4.00
Cone	641	1310	671	4.09	3.90

(a)

Data set	CT	IG	WE	DC	HE	QE	L
Football 1	26.6k	18.7k	21.1k	16.4k	25.7k	21.1k	14.0k
Football 2	13.7k	12.0k	13.5k	10.5k	16.5k	13.5k	9.00k
Crumb	6.1k	4.5k	5.1k	4.0k	6.2k	5.1k	3.4k
Multidode	1.7k	1.2k	1.4k	1.1k	1.7k	1.4k	0.9k
Torus	246k	164k	184k	143k	225k	184k	123k
Cone	16.1k	10.5k	11.8k	9.17k	14.4k	11.8k	7.86k

(b)

Table 4: *(a) Six data sets of manifold cell 2-complexes: deg(V)=Average number of faces incident at a face, deg(F)=Average number of vertices on a face; (b) Storage cost of seven data structures for data sets in (a): CT (Cell Tuple), IG (Incidence Graph), WE (Winged Edge), DC (DCEL), HE (Half-Edge), QE (Quad-Edge), L (Lath)*

- Winged-Edge: $13n_2 + n_0 \approx 27n_0$
- DCEL: $10n_2 + n_0 \approx 21n_0$
- Half-Edge: $16n_2 + n_0 \approx 33n_0$
- Quad-Edge: $13n_2 + n_0 \approx 27n_0$
- Laths: $9n_2 \approx 18n_0$
- Corner Table: $6n_2 \approx 12n_0$
- Star-Vertex: $6n_2 \approx 12n_0$
- IS: $9n_2 + n_0 \approx 19n_0$
- IA: $6n_2 + n_0 \approx 13n_0$

We have evaluated the storage costs of these data structures for six data sets of manifold simplicial 2-complexes and the results are reported in Table 5. We can see that the space requirements of the Corner-Table, of the IA and of the Star-Vertex data structures are comparable, but all of them encode only vertices and triangles. When applied to simplicial complexes, the edge-based representations have the largest space requirements, at least twice the storage cost of those encoding only vertices and triangles. The IS data structure and the lath-based ones are somehow in-between, and, as the edge-based representations, encode all the entities uniquely and explicitly.

Finally, we summarize in Table 6 the navigation costs by evaluating the optimality of algorithms for retrieving topological relations on the various representations.

Data set	n_0	n_1	n_2	deg(V)
Car	6.94k	18.0k	11.8k	5.09
Doll	551	1.38k	831	4.52
Face	2.09k	6.15k	4.05k	5.83
Temple	6.85k	17.8k	11.00k	4.82
Sofa	8.09k	23.5k	15.1k	5.61
Lion	5.17k	15.2k	10.1k	5.84

(a)

Data set	WE	DC	HE	QE	L	CoT	SV	IS	IA
Car	163k	127k	199k	163k	108k	70.7k	70.7k	114k	77.7k
Doll	12.4k	9.65k	15.2k	12.4k	8.27k	5.0k	5.0k	8.56k	5.54k
Face	55.3k	43.0k	67.6k	55.3k	36.9k	24.3k	24.3k	38.8k	26.4k
Temple	160k	125k	196k	160k	107k	66.0k	66.0k	111k	72.9k
Sofa	211k	164k	258k	211k	141k	90.8k	90.8k	147k	98.9k
Lion	137k	106k	167k	137k	91.1k	60.4k	60.4k	96.1k	65.5k

(b)

Table 5: *(a) Six data sets of manifold simplicial 2-complexes: deg(V)=Average number of faces incident at a face; (b) Storage cost of nine data structures for data sets in (a): WE (Winged Edge), DC (DCEL), HE (Half-Edge), QE (Quad-Edge), L (Lath), CoT (Corner Table), SV (Star-Vertex), IS (Incidence Simplicial) and IA*

Data Structure	Boundary relations	Co-boundary relations	Adjacency relations
Winged-Edge	Optimal	Optimal	Optimal
DCEL	Optimal	Optimal	Optimal
Half-Edge	Optimal	Optimal	Optimal
Quad-Edge	Optimal	Optimal	Optimal
Lath	Optimal	Optimal	Optimal
Star-Vertex	Optimal	Not supported	$R_{0,0}$: optimal Others: not supported
Corner Table	Optimal	Optimal	Optimal

Table 6: *Navigation performances of data structures for 2-dimensional complexes specific for manifold domains*

6.2. Representations for Arbitrary Two-Dimensional Complexes

In this Subsection, we review representations for non-manifold shapes discretized through cell and simplicial complexes. The first data structure proposed in the literature for cell 2-complexes is the *Radial Edge (RE)* data structure [Wei88], which has been extended and specialized in [GCP90, YK95]. More recent simplified representations are the *Partial Entities (PE)* [LL01] and the *Loop Edge-use (LE)* data structure [MH01] for cell complexes. The PE data structure has the same representation power as the RE data structure, but it is considerably more compact. The LE data structure is a specialization of the RE data structure to regular cell complexes.

© The Eurographics Association 2007.

We describe first the Radial-Edge and the Partial-Entities data structures first. Then, we present and analyze in details data structures for simplicial 2-complexes, namely, an edge-based data structure, the *Directed Edge (DE)* data structure [CKS98], which can be viewed as an extension of the Half-Edge data structure to the non-manifold simplicial case, an adjacency-based data structure, called the *Triangle-Segment (TS)* data structure [DMPS04], which extends the IA data structure to the non-manifold case, and an incidence-based data structure called the *Vertex-Face (VF)* data structure [VL97]. We compare such representations based on their storage requirements and on their performance in retrieving topological relations, also with respect to two-dimensional instances of the Incidence Graph and of the IS data structure described in Subsections 5.2 and 5.4, respectively.

One important issue in evaluating a data structure for non-manifold shapes is its *scalability* to the manifold case, which is evaluated as the overhead of the storage cost of data structure when applied to a manifold shape with respect to that of a data structure of the same type but specifically designed for manifold shapes. This is relevant since in a typical modeling scenario we need to have a representation capable to deal with non-manifold shapes, but most of the shapes will be in any case manifold.

6.2.1. The Radial-Edge data structure

The *Radial Edge (RE)* data structure [Wei88] has been developed in order to describe the decomposition of the boundary of non-manifold and non-regular three-dimensional objects. The decomposition is not a cell complex as defined in algebraic topology, since the 2-cells are not necessarily homeomorphic to closed disks, but they can be multiply connected 2-manifolds with boundary. The connected components formed by the edges bounding any 2-cell (face) are called *loops*. The entities in the RE data structure are thus: regions, shells, faces, loops, edges and vertices. A *region* is a solid objects, which is bounded by a collection of shells. A *shell* is thus an oriented boundary surface of a region, consisting of maximal connected sets of 2-cells (faces). In addition, faces, loops, edges and vertices are characterized by orientations, namely *face-uses*, *loop-uses*, *edge-uses* and *vertex-uses*. A face f has two face-uses associated with it, which correspond to the two possible orientations of f. The oriented boundary of a face-use is described by loop-uses. A loop-use is composed of a circular list of edge-uses. Each edge-use associates an edge e with the orientation induced on e by the face-use to which it belongs. Since each edge is bounded by two vertices, each edge-use is associated with two vertex-uses, which describe the use of those vertices as the boundary of that edge-use.

Here, we present, for clarity, a simpler version of the RE data structure for representing an object described by a connected cell 2-complex, in which the 2-cells are homeomorphic to disks, and there are no isolated vertices. Thus, every face is bounded by exactly one loop. This simple version of the RE data structure does not contain high-level topological elements, namely, regions, and shells. This simple version of the RE data structure has the following entities: faces, edges, vertices, face-uses (which also capture their oriented boundaries originally described by loop-uses), edge-uses and vertex-uses. It encodes only the following information:

- For each face f, a reference to a face-use (for example, in Figure 10(a) f_1 points to f_{u1});
- For each face-use f_u (see f_{u1} of Figure 10(a)):
  - the face f which it bounds (that is f_1 in the example);
  - a reference to the other face-use on the boundary of f (that is f_{u2} in the example);
  - a reference to an edge-use on f_u (that is e_{u1} in the example);
- For each edge e, a reference to an edge-use that is associated with e (for example, in Figure 10(b) e points to e_{u1});
- For each edge-use e_u in face-use f_u of face f (such as e_{u1} in Figure 10(b)):
  - the corresponding undirected edge e;
  - its face-use f_u (that is f_{u1} in the figure);
  - the mate edge-use in the other face-use of f (that is e_{u2} in the figure);
  - the adjacent edge-use radially ordered around e (that is e_{u6});
  - the previous edge-use in f_u (see Figure 10(c), the previous edge-use of e_{u1} is e_{u4});
  - the next edge-use in f_u (in Figure 10(c), the next edge-use of e_{u1} is e_{u2});
  - the start vertex-use of e_u (in Figure 10(c), the start vertex-use of e_{u1} is v_{u1};
- For each vertex v, a reference to one vertex-use that is associated with v (in Figure 10(d), v has a reference to v_{u1};
- For each vertex-use v_u that is associated with one edge-use e_u (such as vertex-use v_{u1} of Figure 10(d)):
  - the corresponding undirected vertex v;
  - the previous vertex-use of v (that is v_{u5});
  - the next vertex-use of v (that is v_{u2});
  - its edge-use e_u (that is e_{u1});

While the edge-uses around an edge can be ordered, the vertex-uses at a vertex cannot be ordered. Therefore, the list of vertex-uses at vertex v simply collect all the vertex-uses at v.

The RE data structure can be formalized in terms of topological relations as follows (note that the formalization does not take into account the orientations captured by face-uses, edge-uses and vertex-uses):

- For each face f: relation $R^*_{2,1}(f)$, which consists of one edge on the boundary of f,

© The Eurographics Association 2007.

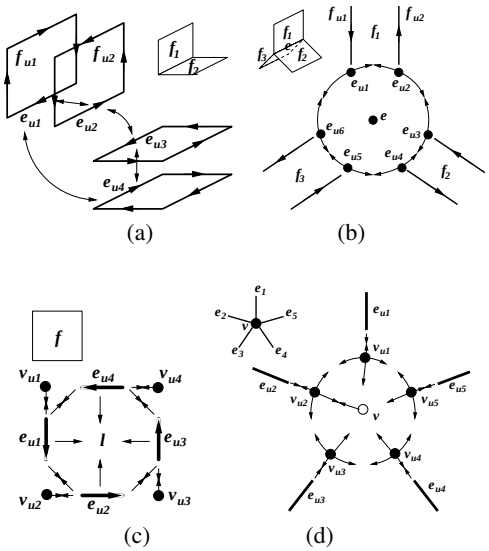


Figure 10: *(a) Edge-uses radially ordered around an edge between two faces; (b) Cross-section view of edge-uses radially ordered around an edge between three faces (c) Planar view of a loop l bounding face f; (d) Vertex-uses of the same vertex v*

- For each edge e:
  - relation $R_{1,2}(e)$, where the faces are ordered around e;
  - partial relation $R^*_{1,1}(e)$, defined as the collection of the pair of edges adjacent to e and bounding the faces incident in e, ordered around e, so that both the $2i$-th element and the $(2i+1)$-element in this relation are on the i-th face in $R_{1,2}(e)$;;
  - relation $R_{1,0}(e)$, ordered by indices of the vertices;
- 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 face defined by an edge-use. Relation $R^*_{1,1}(e)$ captures the association between an edge-use e_p and the edges following and preceding e_p in the boundary of the face f with which e_p is associated. The adjacency of edge-uses at the same edge e is implicitly expressed through the order in $R^*_{1,1}(e)$. It can be seen that all topological relations can be retrieved in optimal time from the RE data structure.

The Tri-Cyclic Cusp representation [GCP90] extends the RE data structure with new elements (called cusps) introduced to handle the inclusion relations of topological disks at non-manifold vertices. The Coupling Entities representation [YK95] is an improvement over both the RE and the Tri-Cyclic Cusp data structures, obtained by introducing entities that address the relationships at the loop cycles formed by edges around faces, the radial cycles formed by faces around edges and cycles formed by faces at vertices.

The RE representation is not highly scalable to the degree of manifold singularities of a shape. It has been shown in [LL01] that, when the domain is manifold, the RE representation occupies about four times as much storage space as the manifold representations such as the Winged-Edge described in Section 6.1.1, which resembles the RE in terms of the entities and relations encoded.

6.2.2. The Partial Entities Data Structure

The *Partial Entities (PE) data structure* [LL01] has been proposed for cell complexes. It encodes *regions*, *shells*, *faces*, *loops*, *edges*, and *vertices*, *partial-faces* which are the two sides of a face, *partial-edges* and *partial-vertices*.

Each face has a unique orientation defined based on the geometry of its surface normal. Each face is bounded by one loop with orientation, which consists of a cycle of partial-edges. Each partial-edge corresponds to the appearance of an edge on a 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 top 1-simplex we, called a *wire-edge*, has a loop that consists of two partial-edges of we. The partial-edge is comparable to the edge-use in the RE data structure, except that each edge-use is associated with one face-use, while a partial-edge is associated with a face. As there are two face-uses to each face in the RE data structure, the number of edge-uses in the RE data structure is twice the number of partial edges in the PE data structure.

Partial-faces are the defining components of shells and are comparable to the face-uses in the RE data structure. Each face has two partial-faces as a face may belong to two shells. A partial vertex is a copy of a vertex split for each manifold surface sharing it.

The PE data structure is designed to encode objects with several boundaries and several connected components. By limiting the domain to objects with just one connected component, and with faces that are homeomorphic to 2-disks, and are thus bounded by one loop, we have simplified the original PE data structure for the purpose of highlighting its capability in representing the connectivity among the entities of a cell complex. Therefore, high-level topological elements, namely, regions and shells are not represented. We also do not consider partial-faces and partial-vertices because their primary function is associated with the description of the high-level topological elements. The simplified version of the PE data structure encodes the following information (we refer to Figure 11 to illustrate it):

- For each face f: a reference to a partial-edge on its boundary. (In Figure 11, f_1 has a reference to e_{p1});
- For each partial-face f_p
- For each edge e, a reference to a partial-edge that describes e. (In Figure 11, e has a reference to e_{p1});

© The Eurographics Association 2007.

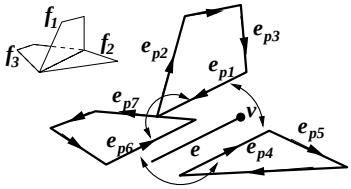


Figure 11: *Elements of the PE structure: relations at a non-manifold edge e shared by three faces: f_1, f_2, f_3*

- For each partial-edge e_p bounding face f, there is a reference to: (see e_{p1} in Figure 11 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 all partial-edges e_p that start at v (In Figure 11, 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 as follows:

- For each face 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;
  - Partial relation $R^*_{1,1}(e)$ which is defined as follows: $R^*_{1,1}(e)$ consists of the edges on the boundary of the faces 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 face defined by a partial-edge. Relation $R^*_{1,1}(e)$ captures the association between a partial-edge e_p and the edges following and preceding e_p in the boundary of the face f with which e_p is associated. The adjacency of partial-edges at the same edge e is implicitly expressed through the order in $R^*_{1,1}(e)$. Thus, the PE data structure encodes the same relations as the RE one. All topological relations can be retrieved in optimal time from the PE data structures, as described in [LL01].

It has been shown in [LL01] that the storage cost of the PE representation of non-manifold cell-complexes is half that of the RE representation, while the PE representation of a manifold cell 2-complex uses twice as much space as that of the Winged-Edge representation (see Section 6.1.1).

The primary difference between the RE and the PE data structure is that the PE data structure considers each face to have one orientation geometrically defined based on its face normal. The orientation of its boundary can thus be uniquely defined. In the RE data structure, a face entity is without orientation. The face-uses of the RE data structure describe all the possible orientations of each face. In the RE data structure, the connectivity among the faces, edges and vertices is defined through face-uses, edge uses and vertex-uses. In the PE data structure, however, the connectivity among faces, edges and vertices is captured at the faces, at partial-edges and at partial-vertices (which have almost one-to-one correspondence with their corresponding vertices), Thus, the only entity required for topological navigation is the partial-edge.

6.2.3. 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, to simplicial 2-complexes embedded in the three-dimensional Euclidean space. The DE data structure is based on the concept of directed edge. A *directed edge* e_d of an edge e in a simplicial 2-complex is an occurrence of e on the boundary a triangle incident at e. A directed edge is similar to the *edge-use* and to the *partial-edge* in the RE and PE data structures, respectively.

In the DE data structure, the entities stored are directed edges and vertices. Triangles and undirected edges are not explicitly encoded. Triangles are implicitly referenced through 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. Thus, the DE data structure encodes the following information:

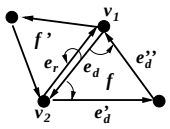


Figure 12: *An illustration of the relations encoded at a directed edge e_d*

- 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 12 for the illustration of the symbols)
  - the start vertex v_1;

© The Eurographics Association 2007.

  - 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 for each connected component of the link of v.

The topological relations encoded in the DE data structure are:

- 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 RE data structure;
- For each vertex v: partial relation $R^*_{0,1}(v)$, which consists of one edge for each connected component of the link of v.

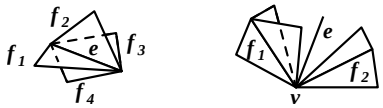
In our formalization, we have considered the undirected edge e, instead of its oriented version and, thus, the information in the directed edges in the DE data structure has been transferred to the undirected edge and described by partial relation $R^*_{1,1}(e)$, and in its ordering. Note that the DE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

The DE data structure is highly scalable to the degree of manifoldness in a simplicial 2-complex. An cost-effective implementation reported in [CKS98] has a storage cost of $68n_2$ bytes, which is 1.13 times the cost of the Winged-Edge data structure for representing 2-manifolds. The DE data structure is also highly adaptable to the availability of memory space by trading off the amount of topological information encoded with assess time. The DE data structure is implementable at three levels of details. The full level has a storage cost of $68n_2$ bytes, while the medium and the small levels have respectively $44n_2$ bytes and $32n_2$ bytes.

6.2.4. The Triangle-Segment (TS) Data structure

The *Triangle-Segment (TS) data structure* [DMPS04] describes simplicial 2-complexes embedded in the two-dimensional Euclidean space. It encodes all vertices, the top 2-simplexes, i.e., the triangles, and the top 1-simplexes, that called *wire-edges* together with the following topological relations (see Figure 13):

- For each triangle t:
  - boundary relation $R_{2,0}(t)$;
  - a partial $R^*_{2,2}(t)$ relation defined as follows: for each edge e of t, it encodes the triangle(s) that are immediately preceding and succeeding t in counter-clockwise order around edge e. In the example of Figure 13(a), $R^*_{2,2}(f_2) = \{f_1, f_3\}$.
- For each wire-edge we, boundary relation $R_{1,0}(we)$;
- For each vertex v:
  - a partial $R^*_{0,2}(v)$ relation which encodes one triangle for each connected component of the link of vertex v, as illustrated in the example of Figure 13(b), $R^*_{0,2}(v) = \{f_1, f_2\}$;
  - a partial $R^*_{0,1}(v)$ relation, which encodes the list of the wire-edges in the star of vertex v, for example, $R^*_{0,1}(v) = \{e\}$ in Figure 13(b).



(a) (b)

Figure 13: *(a) Non-manifold edge e shared by three faces; (b) Non-manifold vertex v shared by two connected components and wire-edge e*

In the TS data structure, only wire-edges are explicitly encoded, but not edges bounding triangles. In a compact implementation of the TS data structure, relation $R^*_{2,2}$ is implemented through arrays and bit flags, while relations $R^*_{0,1}(v)$ and $R^*_{0,2}(v)$ are implemented as linked lists. It has been shown in [DMPS04] that the TS data structure supports the retrieval of all topological relations in optimal time.

In [DMPS04], a highly manifold-scalable implementation of the TS data structure is reported which has a storage cost overhead of one byte per triangle for a manifold simplicial 2-complex.

6.2.5. The Vertex-Face Data Structure

The *Vertex Face (VF) data structure* [VL97] has been developed to describe regular simplicial complexes (i.e., simplicial 2-complexes without wire-edges). It encodes all vertices, edges, triangles explicitly and the following topological relations:

- For each triangle t, boundary relation $R_{2,1}(t)$
- For each edge e, boundary relation $R_{1,0}(e)$
- For each vertex v, co-boundary relation $R_{0,2}(v)$

Boundary relations as well as co-boundary relations based on vertices, namely relations $R_{0,0}(v)$, $R_{0,1}(v)$ and $R_{0,2}(v)$ can be retrieved in optimal time, while edge-based co-boundary relations are retrieved in sub-optimal time, since we need to consider $R_{0,2}$ relation for both the extreme vertices of any edge. As a consequence, also retrieving adjacency relations is sub-optimal. Algorithms for retrieving topological relations are reported in [HF04].

© The Eurographics Association 2007.

6.2.6. Comparisons

We compare the data structures for cell and simplicial 2-complexes in terms of their characteristics, their space requirements and their efficiency in supporting topological navigation. Table 7 summarizes the domain, the kinds of complexes which can be described by the various data structures, and the representation methods.

Data Structure	Domain	Complexes	Method
RE	Non-manifold	Cell	Edge-based
PE	Non-manifold	Cell	Edge-based
DE	Non-manifold	Simplicial	Edge-based
TS	Non-manifold	Simplicial	Adjacency-based
VF	Regular	Simplicial	Incidence-based

Table 7: *Characteristics of the data structures for arbitrary 2-dimensional complexes*

For simplicity, we report in Table 8 the storage costs of the data structures reviewed for simplicial 2-complexes plus the 2D instances of the Incidence Graph and of the IS data structure. The notations are specified in the caption.

Data Structure	storage cost
RE	$73n_2 + n_1 + n_0$
PE	$36n_2 + n_1 + 8n_1^t + n_0$
DE	$15n_2 + 2n_1^t + C_v$
TS	$6n_2 + C_e + C_v$
VF	$6n_2 + 2n_1$
IG	$6n_2 + 4n_1$
IS	$6n_2 + 2n_1 + C_v$

Table 8: *Storage costs of the data structures for simplicial 2-dimensional complexes with n_2 triangles, n_1 edges, of which n_1^t are wire-edges, and n_0 vertices. The total number of connected components at the link of non-manifold edges is denoted by C_e, and the total number of connected-components at all vertices is denoted by C_v*

In Table 9, we report an evaluation of the storage costs for some simplicial 2-complexes. Among the three edge-based data structures, namely, the RE, the PE and the DE, the space consumption of the RE data structure is twice that of the PE, which is about 2.4 times that of the DE. The TS is the most compact among the seven data structures compared in Table 9. The incidence-based data structures, i.e., the VF, the IG and the IS are less space-consuming than the edge-based structures, but not as compact as the adjacency-based TS data structure.

We summarize in Table 10 the navigation costs by evaluating the optimality of the algorithms for retrieving topological relations on the various representations.

Data set	n_0	n_1	n_2	n_1^t	C_e	$C_v - n_0$
cylinders	91	300	204	0	16	2
pies	696	2.98k	2.30k	0	1.92k	72
frame	987	2.16k	1.08k	216	0	390
cubes	2.20k	10.7k	9.60k	0	14.9k	0
densetower	8.32k	24.6k	15.9k	896	2.42k	1.92k

(a)

Data set	RE	PE	DE	TS	VF	IG	IS
cylinders	15.3k	7735	3153	1333	1824	2424	1917
pies	172k	86.6k	35.3k	16.5k	19.8k	25.7k	20.5k
frame	82.0k	43.8k	18.0k	7.9k	10.8k	15.1k	12.2k
cubes	714k	358k	146k	74.7k	79.0k	100k	81.2k
densetower	1,191k	611k	250k	108k	144k	194k	155k

(b)

Table 9: *(a) Five non-manifold 2D simplicial data sets: C_e=# connected components at non-manifold edges, C_v=# connected components at all vertices; (b) Storage cost of seven data structures for 2D data sets in (a)*

Data Structure	Boundary relations	Co-boundary relations	Adjacency relations
RE	Optimal	Optimal	Optimal
PE	Optimal	Optimal	Optimal
DE	Optimal	Optimal	Optimal
TS	Optimal	Optimal	Optimal
VF	Optimal	$R_{0,k}$: optimal Others: sub-optimal	Sub-optimal

Table 10: *Navigation efficiency of data structures for 2-dimensional complexes specific for non-manifold domains*

7. Representations for Three-dimensional Cell and Simplicial Complexes

In this Section, we discuss representations for three-dimensional cell and simplicial complexes embedded in the three-dimensional Euclidean space. There are relatively few representations for describing 3D shapes discretized as cell and simplicial 3-complexes. Most of such representations are limited to the manifold domain. Representations for manifold cell complexes are the Facet-Edge [DL89, Muc93] and the Handle-Face [LT97] data structures. The Compact Half-Face (CHF) data structure [LLLV05] specializes the handle-face data structure to manifold simplicial 3-complexes (these latter are usually called *tetrahedral meshes*). The Non-manifold Indexed Data Structure with Adjacencies (NMIA) proposed in [DH03] generalizes the IA data structure to arbitrary simplicial complexes. In this Section, we analyze and compare such representations also with respect to the three-dimensional instances of the dimension-independent data structures discussed in Section 5.

© The Eurographics Association 2007.

7.1. The Facet-Edge (FE) Data Structure

The *Facet-Edge (FE)* data structure [DL89] is an extension of the *Quad-Edge* data structure developed for cell 2-complexes (see Subsection 6.1.2), and thus it is an implicit representation for manifold cell 3-complexes. The basic entities encoded in the Facet-Edge data structure are the vertices and the so-called *facet-edges* defined on the 2-cells and 1-cells (faces and edges). The three-dimensional cells and their topological information are encoded through the topological vertex-based relations of the complex which is dual to the given one. The dual The 0-cells of the dual complex correspond to the 3-cells of the original complex, its 1-cells to the faces, its 2-cells to the edges and its 3-cells to the vertices.

The boundary of each face (2-cell) f contains a ring of edges (1-cells) $e_1,..,e_n$. This ring is called a *face-ring*, and may be ordered in two directions. The star of an edge e contains a ring of faces $f_1,...,f_m$. This ring is called an *edge-ring* and can also be ordered in two directions. A *facet-edge* pair uniquely associates a face f with an edge e on the boundary of f. Each facet-edge pair exists in four versions. Each version is associated with exactly one face-ring of f and exactly one edge-ring of e. Each version of a facet-edge has its dual which is defined in the dual complex. Figure 14(a)-(h) illustrates the concept of facet-edge. The four unique facet-edges formed by face f_1 and edge e_1 in Figure 14(a) are shown in Figures 14(b)-(e). Figure 14(f) shows the dual complex of the one shown in Figure 14(a), in which edge e_1^* is the dual of f_1 and the highlighted faces correspond to the edges of f_1. Figure 14(g) shows the dual of the facet-edge shown in Figure 14(b).

Given a face f and an edge e on its boundary, the four versions of facet-edges and the duals of each of them are related by the following operators.

- **Clock**, which returns the facet-edge with the face-ring in reversed direction;
- **Rev**, which returns the facet-edge with the edge-ring in reversed direction;
- **Fnext**, which returns the facet-edge of the next face in the same edge-ring as f;
- **Enext**, which returns the facet-edge of the next edge in the same face-ring as e;
- **Dual**, which returns the facet-edge with the same orientation in the dual complex.

Operators **Clock**, **Rev**, **Fnext** and **Enext** are illustrated in Figure 14(h) for the facet-edge shown in Figure 14(b).

In addition, the relations between the facet-edges and their vertices are described by the operator **Org**, which returns the start vertex of each facet-edge. **Org** induces a partition (known as *origin partition* on the set of facet-edges in the complex and in its dual complex. Note that, while the origin partition in the 3-complex captures the incidence relations between edges and vertices, such a partition in the dual complex captures the incidence relations between the 3-cells and their bounding faces.

The Facet-Edge data structure describes a complex Σ by encoding, for each facet-edge pair a associated with edge e in a face-ring and with face f in an edge-ring, the preceding and succeeding facet-edges in both the face-ring and the edge-ring of a. The successors of a in the face-ring in two opposite directions correspond to a**Fnext** and a**ClockFnext**. The successors of a in the edge-ring in two opposite directions correspond to a**DualFnext** and a**DualClockFnext** in the dual-complex of Σ. Using the example of Figure 14(a), the successors of (f_1,e_1) in the face-ring of e_1 in both directions are respectively (f_2,e_1) and (f_3,e_1). The successors of (f_1,e_1) in the edge-ring of f_1 are (f_1,e_2) and (f_1,e_3), which correspond to the facet-edges (f_2^*,e_1^*) and (f_3^*,e_1^*) shown in Figure 14(f).

The Facet-Edge data structure also encodes the vertex-based functions by implementing the partition of the facet-edges induced by the **Org** operator on the 3-complex and on the dual complex.

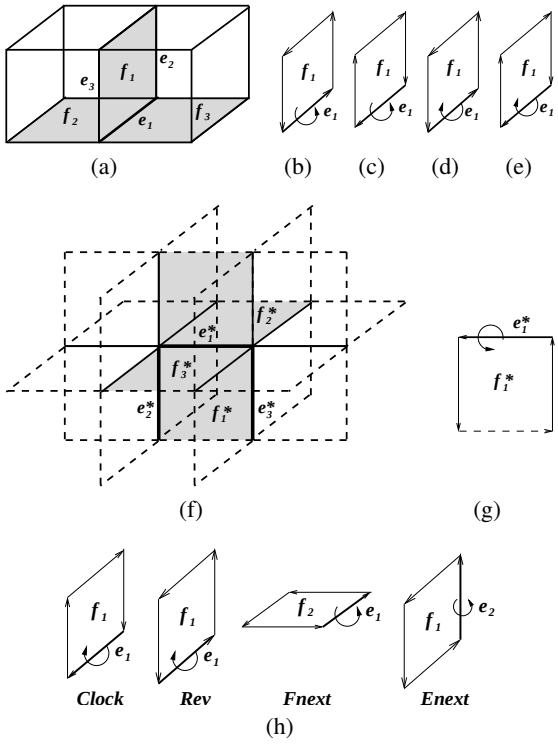


Figure 14: *An illustration of the concept of facet-edge: (a) a model of two cubes; (b)-(e) the four facet-edge pairs formed by face f_1 and edge e_1; (f) the dual complex of (a); (g) the dual of the facet-edge shown in (b); (h) the facet-edges mapped from (b) by operators **Clock**, **Rev**, **Fnext**, and **Enext**, respectively*

In terms of topological relations, the FE data structure en-

© The Eurographics Association 2007.

codes relations $R_{2,1}$ in the form of facet-edges, relation $R_{1,2}$ in the form of face-rings, partial relation $R^*_{1,1}$ in the form of edge-rings, relations $R_{1,0}$ and $R_{2,3}$ implicitly as the incident vertices of the edges, and relation $R_{0,1}$ as the origin partition. Relation $R_{3,2}$ is implicitly encoded as the origin partition of the dual. It can be shown that topological relations can be retrieved from the FE data structure in optimal time.

The storage cost of the FE data structure for a general cell 3-complex with n_3 3-cells, n_2 faces, and n_0 vertices is $4\sum_{i=1..n_2} deg(f_i)+\sum_{i=1..n_3} deg(c_i)+\sum_{i=1..n_0} deg(v_i)$, where $deg(f_i)$ is the number of edges on the i-th face, $deg(c_i)$ the number of faces on the i-th 3-cell and $deg(v_i)$ the degree of vertex v_i. The term $4\sum_{i=1..n_2} deg(f_i)$ derives from the encoding of the successors on the edge-ring and face-ring of each facet-edge. The remaining two terms derive from the vertex-based relations. In the case of simplicial 3-complexes, $deg(f_i) = 3$, $deg(c_i) = 4$ and $\sum_{i=1..n_0} deg(v_i) = 2n_1$, so the amount of information encoded by the FE data structure is $4n_3 + 12n_2 + 2n_1$.

Unlike incidence-based and adjacency-based representations, the FE data structure does not explicitly encode cells as entities but it preserves the orientation of the cells, which makes it a suitable choice for applications that depend on the orientation of cells. The FE data structure has been specialized to the simplicial case in *Triangle-Edge data structure* [Muc93] for the application of computing a tetrahedralization of a solid object and of the simplification of tetrahedal meshes [NE04].

7.2. The Handle-Face (HF) Data Structure

The *Handle-Face (HF) data structure* [LT97] is an explicit representation for manifold cell 3-complexes. It is similar to representations for cell 2-complexes embedded in the three-dimensional Euclidean space, like the RE and the PE data structures (see Section 6.2), since the 3-cells are described in the HF data structure through their boundaries, made by faces, edges and vertices. The HF data structure contains two types of entities: the *basic entities*, which are *faces*, *edges* and *vertices* in the complex, and the *surface entities*, which describe the boundary of a 3-cell. The surface entities are *surfaces*, *half-faces*, *surface edges*, *surface oriented edges* and *surface vertices*. Each half-face belongs to one 3-cell surface, and is bounded by a cycle of surface-oriented edges, whose orientation is aligned with the orientation of the half-face. Each surface-edge belongs to one 3-cell, and corresponds to exactly one pair of surface-oriented edges on the same 3-cell. At most two half-faces may correspond to one face. Several surface-edges may correspond to the same edge, and the same occurs for surface vertices. Figure 15 illustrates the entities in the HF data structure.

Thus, the HF data structure encodes the 3-cells as collection of half-faces bounded by surface-oriented edges belonging to surface-edges incident at surface vertices. Each

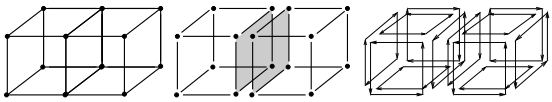


Figure 15: *(a) A 3-complex with two cubic 3-cells, composed of 11 faces, 20 edges, and 12 vertices; (b) there are 12 half-faces, 24 surface-edges, and 16 surface vertices in the whole complex. of all the half-faces, only two (the shaded pair) are interior half-faces, while the rest are boundary half-faces; (c) surface oriented edges of the two 3-cells*

half-face, surface-edge and surface-vertex is associated with the corresponding actual face, edge and vertex in the complex. Besides these incidence relations, two adjacency relations are encoded, namely, the one between each pair of half-faces belonging to the same face, and the one of the surface-oriented edges that correspond to the same edge in the same pair of half-faces.

A surface-oriented edge represents an association between a surface-edge and a half-face, and it is associated with its own half-face and its start surface-vertex. Thus, it corresponds to the half-edge in the Half-Edge data structure, to the edge-use in the RE data structure and to the partial-edge in the PE data structure.

We can formalize the HF data structure in terms of topological relations as follows:

- For each face f, partial $R^*_{2,1}(f)$ relation, which encodes one edge bounding face f;
- For each edge e, partial $R^*_{1,1}(e)$ relation, which encodes all the edges that share a face with edge e, radially ordered around e and relation $R_{1,2}$, which encodes all faces around an edge in the same radial order;
- For each vertex v, partial relation $R^*_{0,1}(v)$, which encodes one edge incident in vertex v.

All the relations are ordered, but the above formalization does not capture the association between faces and half-faces and thus the orientations on the faces. Note that the 3-cells are not represented explicitly in the HF data structure. The drawback is that no attribute can be attached to the 3-cells as a consequence. Also, the HF representation encodes the same relations as in the RE and PE data structures. On the other hand, unlike the RE and the PE data structures, the HF data structure cannot represent shapes with dangling edges or faces, as well as 3D shapes with non-manifold vertices and edges. As all representations which encode orientations by duplicating the basic entities, the HF data structure is quite verbose.

All topological relations at faces, edges and vertices can be retrieved in optimal time from the HF data structure, and the representation of surface entities allows retrieving the

© The Eurographics Association 2007.

boundaries of the 3-cells even if these latter are not explicitly represented.

7.3. The Compact Half-Face (CHF) Data Structure

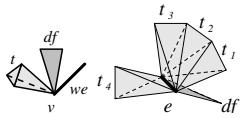
The *Compact Half-Face (CHF) data structure* [LLLV05] is a specialization of the HF data structure for representing manifold simplicial 3-complexes (usually called *tetrahedral meshes*). It is a multi-level data structure that encodes the connectivity among simplexes at different levels of detail. The CHF data structure encodes almost the same entities as the HF data structure. They are the physical entities, namely, tetrahedra, faces, edges, and vertices, and the surface entities which uniquely belong to each tetrahedron, namely, the half-faces and half-edges, which correspond to the surface-oriented edge.

The CHF is designed with four levels (0-3). We describe levels 0 to 2 in detail. Level 3 addresses the boundary information of the 3-manifold, whereby boundary cells are encoded. This level is not elaborated here as it is an application-specific feature of the data structure.

- Level 0 encodes the vertices of each tetrahedron. Given a tetrahedron t, based on the encoded ordering of the vertices of t, the half-faces and half-edges of t can be combinatorially computed and expressed in terms of the order of their vertices. At this level, the only relations explicitly encoded are $R_{3,0}$ for all tetrahedra.
- At level 1, half-faces that belong to the same pair are encoded, and based on that mate half-edges can be computed combinatorially. Formally, the relations encoded are $R_{3,3}$ for all tetrahedra, and partial relation $R^*_{1,3}$, which encodes for each edge on tetrahedron incident in it.
- Up to level 1, only tetrahedra, half-faces and vertices are explicitly addressable entities. Level 2 encodes the faces and edges explicitly by mapping one half-face from each pair of half-faces to the physical face at which attributes can be assigned, and by mapping each half-edge defined by its vertices to one of its incident half-faces. In addition, one incident tetrahedron is encoded for each vertex. The relations explicitly encoded are partial relations $R^*_{0,3}$ which encodes an incident tetrahedron for each vertex, partial relations $R^*_{1,2}$, which associates each edge to its half-face.

The CHF data structure at level 2 supports topological navigation. Namely, the retrieval of tetrahedron-based relations $R_{3,3}$ and $R_{3,0}$, and face-based relations $R_{2,*}$ can be performed in constant time. The retrieval of vertex-based and edge-based co-boundary relations can be performed in time linear with respect to the size of the relations.

The amount of information encoded by the CHF data structure up to level 2 is $8n_3 + n_2 + n_1 + n_0$, where each of level 0 and level 1 contributes to $4n_3$, level 2 contributes to $n_2 + n_1 + n_0$. In the actual implementation described in [LLLV05], the mapping between the half-faces and the actual faces uses a red-black tree structure, which requires a cost of $O(n_2 log_2 n_2)$, and the mapping between the edges and the half-faces has a cost of $O(n_1 log_2 n_1)$. The storage cost of this implementation of the CHF data structure is $O(8n_3 + n_2 log_2 n_2 + n_1 log_2 n_1 + n_0)$



(a) (b)

Figure 16: *Properties of simplicial complexes in 3D space: (a) non-manifold vertex whose star has more than one connected components which may be of mixed dimensions; (b) non-manifold edge whose star consists of fans of tetrahedra and dangling faces*

7.4. The Non-manifold Indexed Data structure with Adjacencies

The *Non-Manifold Indexed data structure with Adjacencies (NMIA)* [DH03] is a representation for arbitrary simplicial 3-complexes embedded in the three-dimensional Euclidean space. Another representation for such complexes based on a decomposition approach will be described in Section 8. The NMIA data structure is an adjacency-based data structure which encodes the vertices, and all the top simplexes of a simplicial 3-complex, namely tetrahedra, top 2-simplexes, that we call *dangling faces* and top 1-simplexes, that we call *wire-edges* (for instance, triangle *df* Figure 16(a)), and a top 1-simplex a *wire edge* (for instance, edge *we* in Figure 16(a)).

Given a simplicial 3-complex Σ, we consider, for each edge e of Σ, the sub-complexes of the star of e bounded by different connected components of the link of e. Any of such sub-complexes is called an *edge-based cluster*. An edge-based cluster may consist of just one single dangling face (such as *df* in Figure 16(b) and the three faces in Figure 16(c)), one single tetrahedron, or a fan of tetrahedra (such as t_1, t_2 and t_3 in Figure 16(b)).

The *NMIA* data structure encodes the following topological relations:

- For each tetrahedron t:
  - relation $R_{3,0}(t)$, i.e., the four vertices of t;
  - relation $R_{3,3}(t)$, i.e., the four tetrahedra adjacent to t along a triangle;
  - for each non-manifold edge e of t, the top simplexes preceding and following t around e, and not belonging to the same edge-based cluster as t. For the example of Figure 16(b), we associate with t_3) at edge $e\,t_4$, but not t_2.
- For each dangling face f:

© The Eurographics Association 2007.

  - relation $R_{2,0}(f)$;
  - for each non-manifold edge e of f, the top simplexes preceding and following f around e. In the example of Figure 16(b), t_4 and t_1 are associated with dangling face d_f,
- For each wire-edge w: relation $R_{1,0}(w)$;
- For each vertex v: one top simplex from each connected component of the link of v.

The NMIA data structure is a non-manifold extension of the IA data structure. The extension is done by encoding the multiple connected components at non-manifold vertices and non-manifold edges. Note that the NMIA data structure encodes a partial version of relations $R_{0,1}$, $R_{0,2}$ and $R_{0,3}$, at each vertex v, namely $R_{0,1}$ only for wire-edges, $R_{0,2}$ only for the connected components of the link of v consisting only of edges (and thus the corresponding portion of the star consists only of dangling faces), and $R_{0,3}$, for each connected component of the link containing at least one triangle (and thus the corresponding portion of the star contains at least one tetrahedron). At any non-manifold edge e, it encodes a partial $R_{1,2}$ relation, restricted to dangling faces in the star of e, and a partial $R_{1,3}$ relation, restricted to the tetrahedra in the star of e, which belong to different edge-based clusters.

An efficient and compact implementation of the NMIA data structure is described in [DH03]. For a simplicial complex with n_3 tetrahedra, n_2^t dangling faces and n_1^t wire-edges, such that the total number of components at non-manifold vertices is C_e and the total number of components at all vertices is C_v, the storage cost of such implementation is $8n_3 + 3n_2^t + 2n_1^t + C_e + C_v$. The data structure scales very well to manifold simplicial complexes. In the case of a manifold domain, $n_2^t = n_1^t = C_e = 0$ and $C_v = n_0$. Therefore, the NMIA data structure encodes only $8n_3 + n_0$ pieces of information, which is equivalent to the IA (see Section 5.3).

All boundary topological relations can be retrieved in optimal time from the NMIA data structure. Co-boundary relations $R_{1,3}$ and $R_{0,3}$ can be retrieved in sub-optimal time because they are retrieved by visiting the stars of edges and vertices, respectively, which may consist of dangling-faces not in the relations. All other co-boundary relations can be retrieved in optimal time. Apart from adjacency relation $R_{3,3}$, which is encoded in data structure, all other adjacency relations are retrieved through a combination of boundary and co-boundary relations and can be retrieved in optimal time. Algorithms for extracting topological relations from the NMIA data structure are described in [DH03]. In [DH04], efficient algorithms are presented for performing update operations on a simplicial 3-complex encoded in an NMIA data structure.

7.5. Comparisons

We compare the above data structures in terms of their characteristics, their storage costs and efficiency in supporting topological navigation. Table 11 summarizes the comparison among the various data structures in terms of their domain, represented complex, and representation method.

Data Structure	Domain	Complexes	Method
HF	Manifold	Cell	Edge-based
CHF	Manifold	Simplicial	Edge-based
Facet-Edge	Manifold	Cell	Implicit
NMIA	Non-Manifold	Simplicial	Adjacency-based

Table 11: *Characteristics of data structures for 3-complexes*

We evaluate here the storage costs of the various data structures for simplicial 3-complexes except the HF data structure, which is represented into the CHF, and the specialization of the dimension-independent ones for the manifold and non-manifold domains.

In the case of manifold simplicial 3-complexes, the storage cost of the data structures are evaluated to be:

- NMIA: $8n_3 + n_0$
- IA: $8n_3 + n_0$
- IG : $8n_3 + 6n_2 + 4n_1$
- IS : $4n_3 + 5n_2 + 3n_1 + n_0$
- CHF: $8n_3 + n_2 log_2 n_2 + n_1 log_2 n_1 + n_0$
- FE: $4n_3 + 12n_2 + 2n_1$

A numerical comparison of their storage costs is made experimentally based on five data sets of manifold simplicial 3-complexes shown in Table 12. The CHF in its full capacity has the highest storage cost, seconded by the implicit FE representation, which has slightly more than half its storage cost. The incidence-based IG and the IS data structures are more compact than the FE. The most compact is the adjacency-based IA data structure.

Data set	n_0	n_1	n_2	n_3
Rings	2.52k	13.2k	18.8k	8.13k
Basket	1.21k	6.43k	9.22k	4.00k
Cylinder	1.31k	7.79k	11.6k	5.16k
Gargoyle	2.73k	14.7k	22.0k	10.0k
Torus	2.29k	15.4k	24.0k	10.9k

(a)

Data set	IA	IG	IS	CHF	FE
Rings	67.6k	231k	169k	516k	285k
Basket	33.2k	113k	82.6k	237k	139k
Cylinder	42.6k	142k	104k	295k	176k
Gargoyle	82.7k	271k	197k	597k	333k
Torus	89.2k	293k	212k	641k	362k

(b)

Table 12: *(a) Five manifold 3D simplicial data sets; (b) Storage cost of six data structures for the data sets in (a)*

The only data structures that can describe non-manifold

© The Eurographics Association 2007.

simplicial 3-complexes are the NMIA, the Incidence Graph (IG) and the IS data structures. Table 13(a) shows five non-manifold data sets with mixed tetrahedra, dangling-faces and wire-edges. The storage cost of the NMIA data structure and of 3D instances of the IG and of the IS data structure are compared for these 3-complexes. The NMIA is the most compact as it encodes only tetrahedra and vertices explicitly. The IG is at least three times as much as the NMIA because it encodes all simplexes and a large number of incidence relations. IS is more compact than the Incidence Graph because it only encodes a subset of the incidence relations encoded by the IG.

Data set	n_0	n_1	n_2	n_3	n_1^t	n_2^t	C_e	$C_v - n_0$
Bucket	53	167	160	48	6	32	32	14
Wheel	402	2093	2728	1148	96	32	56	256
Balloon	1108	3913	3616	856	64	1632	0	160
Flasks	1301	6307	8465	3455	0	460	104	0
Teapot	4658	17.9k	17.0k	5666	2944	3930	144	5959

(a)

Data set	NMIA	IG	IS
Bucket	591	2012	1105
Wheel	10.2k	33.9k	17.7k
Balloon	13.1k	44.2k	23.4k
Flasks	30.4k	104k	53.2k
Teapot	73.8k	219k	120k

(b)

Table 13: *(a) Five non-manifold 3D simplicial data sets; (b) Storage cost of three non-manifold data structures for the data sets in (a)*

Topological relations can be retrieved in optimal time from the HF, and the FE data structures, while all but $R_{0,3}$ and $R_{1,3}$ relations can be retrieved in optimal time from the NMIA data structure. $R_{0,3}$ and $R_{1,3}$ relations can be retrieved in suboptimal time.

8. Decomposition Approaches

Another way to represent non-manifold shapes consists of decomposing them into manifold, or nearly-manifold, components. The various decomposition approaches in the literature try to realize in the discrete case a stratification of the shape which has been defined for analytic sets (see [Whi65]). In this Section, we focus on those approaches which have been developed as a basis of data structures for non-manifold shapes.

Selective Geometric Complexes (SGCs) [RO89] describe arbitrary-dimensional non-manifold objects through collections of mutually disjoint cells, which are defined as open subsets of d-manifolds. Thus, the cells can be either open, and not simply connected, and they form a stratification of the shape. Note that SGC is not equivalent to a combinatorial cell complex. In SGCs, cells and their mutual adjacencies are encoded in an Incidence Graph (IG) (see Section 5.2). Thus, the storage cost and the navigation efficiency of the SGC representation is exactly that of the IG.

Some techniques have been proposed in the literature for decomposing the boundary of regular non-manifold 3D shapes into manifolds, that is of solid objects without dangling faces or edges, the so-called *r-sets*, [FR92, GTLH98, RC99]. The objective is to apply modeling tools developed for manifold shapes (data structures and manipulation operators) to non-manifold ones. In [FR92], the result of the decomposition is represented as a graph in which the arcs describe non-manifold singularities. The approach has been applied for identifying form features in r-sets. In [RC99], a decomposition algorithm for a non-manifold object is presented which minimizes the number of duplications introduced by the decomposition process. In [GTLH98], the idea of cutting a two-dimensional non-manifold complex into manifold pieces is exploited to develop compression algorithms. A cut-and-stitch technique is proposed for handling non-manifold cell 2-complexes. The cutting part of the cut-and-stitch technique decomposes the star of every non-manifold vertex in such a way that 2-cells (faces) that share manifold edges in the star are in the same component. Each such component is homeomorphic to a disc or to a half-disc. After cutting, the non-manifold edges in the original complex become boundary edges in the new complex. Also, multiple components may have resulted from cutting. The stitching part of the cut-and-stitch technique merges selected edges that are created in the cutting phase

8.1. Combinatorial Stratification

Pesco et al. [PTL04] propose an approach inspired from stratification to represent non-manifold shapes. A cell 2-complex describing the boundary of a 3D non-manifold shape is decomposed into subcomplexes, which are the analogous of the strata (the components) in a stratification of an analytic set. They define a *combinatorial stratification* of a cell 2-complex Γ as a collection of k-dimensional connected combinatorial manifolds $S = \{M_1, \cdots, M_n\}$ ($k = 0, 1, 2$) with or without boundary such that the union $\cup_i M_i$ gives Γ and the intersection between any two elements M_i and M_j in S is either empty of a sub-complex of both M_i and M_j. A combinatorial stratification is not necessarily unique. As an example, two valid stratifications of a simplicial 2-complex in Figure 17(a) are shown in Figures 17(b) and (c). The resulting set of strata and their connectivity provides a description of the original shape which is used as the basis for a data structure for non-manifold shapes discretized as cell 2-complexes, called the *Handle-Cell (HC) data structure*.

The *Handle-Cell (HC) data structure* consists of two sets of cells, namely the *global cells* and the *local cells*. The global cells, i.e., global vertices, global edges and global faces are the vertices, edges and faces of the given cell complex. The local cells are the cells that describe the strata.

© The Eurographics Association 2007.

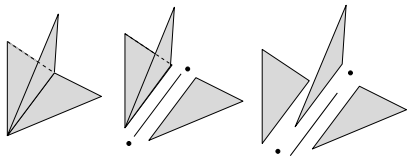


Figure 17: *(a) A 2-complex that consists of three triangles sharing an edge; (b) and (c) two valid stratifications of (a) which result in different manifold components*

The strata are points, curves and surfaces. Curves are composed of curve-vertices and curve-edges. Surfaces are composed of surface-vertices, surface-edges, boundary-curves and surface-faces. Since strata are 2-complexes with a manifold domain, surfaces are represented through the Half-Edge data structure. This is conceptually similar to the representation of the surfaces of the 3-cells in the Handle-Face (HF) data structure (see Section 7.2). Curves are described as lists of edges connected by vertices. The connectivity among the strata is captured through the sharing of global vertices and global edges.

The Handle-Cell data structure can be formalized in terms of topological relations as follows:

- For each face f: Relation $R^*_{2,1}(f)$ which consists of one edge on the boundary of f,
- For each edge e:
  - Relation $R_{1,2}(e)$, which consists of all faces incident in e;
  - Partial relation $R^*_{1,1}(e)$, ordered around edge e, so that both the $2i$-th element and the $(2i+1)$-element in this relation are on the i-th face of e;
  - Relation $R_{1,0}(e)$, which consists of the extreme vertices of edge e;
- For each vertex v: Relation $R_{0,1}(v)$, which consists of the set of edges incident in a vertex v.

The Handle-Cell data structure supports efficient topological navigation, as the incidence relations among q-cells and $(q-1)$-cells are fully encoded and the edge-based adjacency relations among edges in the 2-manifold strata are encoded. The HC data structure encodes a large number of topological relations in order to support incremental shape construction in the non-manifold domain through a specific category of topology-modifying operators.

The Handle-Cell data structure is closely related to the Handle-Face data structure for manifold 3D cell complexes, and it is similar to the data structures for non-manifold 2-complexes, such as the Radial-Edge data structure (Section 6.2.1) and Partial-Edge data structure (Section 6.2.2). The primary difference between the HC representation and the latter group lies in the explicit description of the stratification encoded in the HC data structure. Within the decomposition paradigm, the SGC can be considered as an object stratification into open connected subsets of manifolds, while the HC approach is a combinatorial stratification.

8.2. Initial Quasi-manifold Decomposition

A decomposition of a non-manifold shape into simpler parts can be obtained by splitting the shape at those elements (vertices, edges, faces, etc.) where singularities occur. In order to be effective, the decomposition process should remove as many singularities as possible, without introducing artificial, or arbitrary, “cuts" through manifold parts. Under these assumptions, a decomposition into manifold components is possible, in general, only for two-dimensional complexes. In three or higher dimensions, a decomposition into manifold components may need to introduce artificial cuts through the object. In six or higher dimensions, a decomposition into manifold components is not feasible in general, since the class of d-manifolds has been proven to be not decidable for $d \geq 6$ [Nab96].

In [DMMP03], a decomposition of a non-manifold complex in arbitrary dimensions is proposed, which is *unique*, since it does not make any arbitrary choice in deciding where the object has to be decomposed, and *natural*, since it removes singularities by splitting the complex at non-manifold simplexes only. Such a decomposition is known as the *standard* decomposition of the original complex. The components of such decomposition, 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 Σ in which all top simplexes have dimension d and such that the star of each vertex of Σ is $(d-1)$-connected, i.e., can be traversed by moving between adjacent d-simplexes through their common $(d-1)$-face. If an IQM is embeddable in R^d where $d \geq 3$, it must be a pseudo-manifold complex (i.e., a $(d-1)$-connected complex in which every $(d-1)$-simplex is on the boundary of one, or two d-simplexes).

The properties of the IQM decomposition makes it a good basis for defining representation for non-manifold simplicial shapes. The *Double-Level Decomposition (DLD)* data structure [HVD06] is a representation for a simplicial 3-complex based on its IQM decomposition. The IQM decomposition of a simplicial 3-complex splits the complex into one-, two- and three-dimensional IQM components. One- and two-dimensional IQM components are manifold. In the 3D case, a 3D IQM component is a maximal subcomplex formed by tetrahedra in which the link of every vertex is homeomorphic to a 2D manifold. The class of IQM 3-complexes is thus a superclass of 3-manifolds including shapes such as the pinched-pie shown in Figure 1(b) and (c). The decomposition can be computed by splitting the star of each non-manifold edge e into manifold components, followed by splitting the star of each non-manifold vertex in the same way (see [DMMP03]).

© The Eurographics Association 2007.

In the DLD data structure, an indexed data structure with adjacencies is used to encode each IQM component. An h-dimensional IQM ($h = 1, 2, 3$) can be effectively described by an indexed data structure with adjacencies, since the star of each vertex in any 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 simplex σ of Σ, which is shared by several IQM components, is called a *split simplex*, which can be either a non-manifold vertex or a non-manifold edge. The copy of split simplex σ in a component C_i, to which simplex σ belongs, is denoted as σ_i and it is called a *simplex copy*. The relations among the components in an IQM decomposition of a complex described by the split simplexes is represented as a hypergraph H, called the *decomposition graph*in which the nodes correspond to IQM components and each hyperarc corresponds to a split simplex σ and it connects all components C_i sharing v. In the example shown in Figure 18, vertex v in Figure 18(a) is split into vertices v_1, v_2 and v_3 in the decomposition shown in Figure 18(b). In the hypergraph shown in Figure 18(c), a hyperarc associates v with the three components C_1, C_2 and C_3 through the three vertex copies.

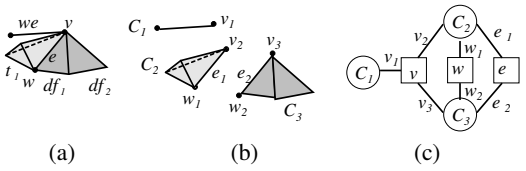


Figure 18: *IQM decomposition of a complex*

The decomposition graph is encoded in the following data structure:

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

The decomposition graph supports both a vertex-based and an edge-based traversal among components connected through the same hyperarc. Given a simplex copy σ_i from any component C_i, we can follow the reference to its hyperarc and find all other simplex copies σ_j connected with σ, as well as all other components sharing σ.

The DLD data structure encodes only the top simplexes, the vertices and the non-manifold edges. It is adjacency-based, since it encodes the adjacency relation among top simplexes as the IA data structure and its extensions.

The IQM decomposition approach taken by the DLD data structure differs from the stratification approach of the HC representation in the following two aspects. First, the HC representation is based a decomposition for cell 2-complexes, approach while the IQM decomposition is dimension-independent. Also, unlike the combinatorial stratification, an IQM decomposition is unique. The DLD representation is specific for simplicial 3-complexes, and is thus highly optimized for compactness and scalability.

We compare the DLD data structure with the NMIA data structure, which also extends the IA data structure to the non-manifold case. It has been shown experimentally in [HVD06] that the storage costs of the DLD and the NMIA data structures are comparable. Both the DLD and the NMIA data structures encode only vertices and top simplexes. Their primary difference is that the DLD data structure encodes the complex as an IQM decomposition, thus allowing the non-manifold singularities to be explicitly represented, while the NMIA data structure encodes the complex as a single piece with non-manifold singularities distributed inside the complex. Both the NMIA and the DLD data structure are comparable to the IA when the domain is manifold, and thus they are highly scalable. Also, both data structures support an efficient retrieval of topological relations.

9. Concluding Remarks

In this state-of-the-art report, we have reviewed, analyzed and compared representations for simplicial and cell complexes, with a special emphasis on data structures for simplicial complexes. We have classified the data structures in each group according to the basic kinds of the topological entities they represent. We have described each data structure in terms of the entities and topological relations it encodes, and we have evaluated it based on its expressive power, on its storage cost, on the efficiency in supporting navigation inside the complex. We have also discuss a decomposition approach to modeling non-manifold shapes, which has led to powerful and highly scalable representations.

The algorithms for extracting topological relations from a complex are the basis for performing update operations on the complex. There is a vast literature on update and construction operators. Updating a manifold 2D cell complex describing the boundary of a 3D object has been extensively studied in the solid modeling literature for more than twenty years, and several proposals exist for primitive update operators which maintain the validity of Euler' formula, the so-called *Euler operators* (see, for instance, [Man87]). Such operators have also been defined for non-manifold two-dimensional complexes (see, for instance, [LL01, LL91, YK95, Wei88]) by considering different variants of Euler' formula. In both the manifold and non-manifold cases, the effect of any other operation on the complex is then expressed as a suitable sequence of Euler operators. Higher-level operators based on the Handle-body

© The Eurographics Association 2007.

theory have been proposed [PTL04]. The Handle-body theory studies the topological changes generated by attaching handles to a manifold without boundary. Handle body operators change also the topological type of the domain of the complex.

Primitives for updating simplicial complexes have been proposed in the literature, mainly for triangle and tetrahedral meshes (see, for instance, [DM02, VG00]). Some of them do not affect the topology of the domain of the complex, but only the combinatorial structure of the subdivision (see, for instance, [VG00]). The most common update operators for simplicial complexes are those applied in the mesh simplification algorithms. The problem of simplification of simplicial complexes has been extensively studied in computer graphics for triangle meshes (see, e.g., [DM02, Gar99, LRC*02], for a survey), and, more recently, some algorithms have been developed for tetrahedral meshes, mostly based on edge collapse. These approaches are based on contracting one edge to one of its extreme vertices or to a new vertex [CM02, CCM*00, GS98, HC94, RO96, THJ99]. In [PH97], the problem of applying a vertex-pair contraction (which consists of contracting a pair of vertices to a new vertex) on a d-dimensional simplicial complex is addressed. The complex is represented as an Incidence Graph. In [DMPS04], we have developed algorithms for performing vertex-pair contraction and its inverse, vertex split, on a two-dimensional simplicial complex described as a TS data structure [DMPS04], and in [DH04] for three-dimensional simplicial complexes described as an NMIA data structure.

Specific simplification algorithms have been developed for finite element mesh generation from CAD models [CDM04, FRL00, VL97, VL01]. In this case, the idealization of a simplicial complex is performed through a set of geometrical and topological transformations [VL01], involving detail removal operators (e.g., vertex removal and re-meshing), which change the shape of a component without modifying its topology, topological detail removal operators (e.g., hole removal), which change the topology of the complex while preserving the dimension of the part, and dimension-reduction operators, which reduce the dimension of a part, by contacting, for instance, a tubular part to a wire.

Acknowledgments

This work has been partially supported by the European Network of Excellence AIMSHAPE under contract number 506766, by the National Science Foundation under grant CCF-0541032, by the MIUR-FIRB project SHALOM under contract number RBIN04HWR8 and by the MIUR-PRIN project on "Multi-resolution modeling of scalar fields and digital shapes".

© The Eurographics Association 2007.

References

[Ago05] AGOSTON M.: *Computer Graphics and Geometric Modelling*. Springer, 2005. 2

[Bau72] BAUMGART B. G.: *Winged-edge polyhedron representation*. Technical Report CS-TR-72-320, Stanford University, Department of Computer Science, October 1972. 1, 7

[Bau75] BAUMGART B. G.: A polyhedron representation for computer vision. In *Proceedings AFIPS National Computer Conference* (1975), vol. 44, pp. 589–596. 7

[Bri89] BRISSON E.: Representing geometric structures in D dimensions: topology and order. In *Proceedings 5th ACM Symposium on Computational Geometry* (1989), ACM Press, pp. 218–227. 4

[CCM*00] CIGNONI P., COSTANZA D., MONTANI C., ROCCHINI C., SCOPIGNO R.: Simplification of tetrahedral volume data with accurate error evaluation. In *Proceedings IEEE Visualization 2000* (2000), IEEE Computer Society, pp. 85–92. 23

[CDM04] CUTLER B., DORSEY J., MCMILLAN L.: Simplification and improvement of tetrahedral models for simulation. In *Proceedings Second ACM/Eurographics Symposium on Geometry Processing* (Nice, France, July 2004). 23

[CKS98] CAMPAGNA S., KOBBELT L., SEIDEL H.-P.: Directed edges - a scalable representation for triangle meshes. *Journal of Graphics Tools 3*, 4 (1998), 1–12. 11, 13, 14

[CM02] CHOPRA P., MEYER J.: TetFusion: an algorithm for rapid tetrahedral mesh simplification. In *Proceedings IEEE Visualization 2002* (October 2002), IEEE Computer Society, pp. 133–140. 23

[DGH04] DE FLORIANI L., GREENFIELDBOYCE D., HUI A.: A data structure for non-manifold simplicial d-complexes. In *Proceedings of the 2nd ACM/Eurographics Symposium on Geometry Processing* (Nice (France), 8–10 July 2004), Kobbelt L., Schroder P., Hoppe H., (Eds.), pp. 83–92. 6

[DH03] DE FLORIANI L., HUI A.: A scalable data structure for three-dimensional non-manifold objects. In *Proceedings of the 1st ACM/Eurographics Symposium on Geometry Processing* (Aachen (Germany), 23–25 June 2003), Kobbelt L., Schroder P., Hoppe H., (Eds.), pp. 72–82. 15, 18, 19

[DH04] DE FLORIANI L., HUI A.: Update operations on 3D simplicial decompositions of non-manifold objects. In *Proceedings of the 9th ACM Symposium on Solid Modeling and Applications* (Genova (Italy), 9–11 June 2004), Fellner D., (Ed.), pp. 169–180. 19, 23

[DH06] DE FLORIANI L., HUI A.: *A Dimension-Independent Simplicial Data Structure for Non-manifold*

Shapes. Tech. Rep. CS-TR-4794, University of Maryland, College Park, April 2006. 4, 6

[DL89] DOBKIN D., LASZLO M.: Primitives for the manipulation of three-dimensional subdivisions. *Algorithmica 5*, 4 (1989), 3–32. 15, 16

[DM02] DE FLORIANI L., MAGILLO P.: Multi-resolution mesh representation: models and data structures. In *Principles of Multi-resolution Geometric Modeling* (Berlin, 2002), Floater M., Iske A., Quak E., (Eds.), Lecture Notes in Mathematics, Springer Verlag, pp. 364–418. 23

[DMMP03] DE FLORIANI L., MESMOUDI M. M., MORANDO F., PUPPO E.: Decomposing non-manifold objects in arbitrary dimension. *CVGIP: Graphical Models 65*, 1/3 (January 2003), 2–22. 21

[DMPS04] DE FLORIANI L., MAGILLO P., PUPPO E., SOBRERO D.: A multi-resolution topological representation for non-manifold meshes. *Computer-Aided Design Journal 36*, 2 (February 2004), 141–159. 11, 14, 23

[Ede87] EDELSBRUNNER H.: *Algorithms in Combinatorial Geometry*. Springer Verlag, Berlin, 1987. 4, 5

[FR92] FALCIDIENO B., RATTO O.: Two-manifold cell-decomposition of R-sets. In *Proceedings Computer Graphics Forum* (September 1992), Kilgour A., Kjelldahl L., (Eds.), vol. 11, pp. 391–404. 20

[FRL00] FINE L., REMONDINI L., LÉON J.-C.: Automated generation of FEA models through idealization operators. *International Journal for Numerical Methods in Engineering 49* (2000), 83–108. 23

[Gar99] GARLAND M.: Multi-resolution modeling: survey and future opportunities. In *Eurographics '99 – State of the Art Reports* (1999), Eurographics Association, pp. 111–131. 23

[GCP90] GURSOZ E. L., CHOI Y., PRINZ F. B.: Vertex-based representation of non-manifold boundaries. In *Geometric Modeling for Product Engineering*, Wozny M. J., Turner J. U., Preiss K., (Eds.). Elsevier Science Publishers B. V., North Holland, 1990, pp. 107–130. 10, 12

[GS85] GUIBAS L., STOLFI J.: Primitives for the manipulation of general subdivisions and computation of Voronoi diagrams. *ACM Transactions on Graphics 4*, 2 (April 1985), 74–123. 7, 8

[GS98] GROSS M. H., STAADT O. G.: Progressive tetrahedralizations. In *Proceedings IEEE Visualization'98* (Research Triangle Park, NC, 1998), IEEE Computer Society, pp. 397–402. 23

[GTLH98] GUEZIEC A., TAUBIN G., LAZARUS F., HORN W.: Converting sets of polygons to manifold surfaces by cutting and stitching. In *Conference abstracts and applications: SIGGRAPH 98* (1998), Computer Graphics, ACM Press, pp. 245–245. 20

[HC94] HAMANN B., CHEN J. L.: Data point selection for piecewise trilinear approximation. *Computer-Aided Geometric Design 11* (1994), 477–489. 23

[HF04] HUI A., FLORIANI L. D.: *Notes on Compact Data Structrures for Finite Elements Applications*. Tech. rep., Department of Computer Science, University of Maryland, College Park, September 2004. 14

[HVD06] HUI A., VACZLAVIK L., DE FLORIANI L.: A decomposition-based representation for 3d simplicial complexes. In *Proceedings of the 4th Eurographics Symposium on Geometry Processing* (Cagliari (Italy), June 2006), pp. 101–110. 21, 22

[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*, Farin G., Hagen H., Hamann B., (Eds.). Springer Verlag, Heidelberg, 2002. 7, 8

[KT01] KALLMANN M., THALMANN D.: Star vertices: a compact representation for planar meshes with adjacency information. *Journal of Graphics Tools 6*, 1 (2001), 7–18. 7, 9

[Lie94] LIENHARDT P.: N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry and Applications 4*, 3 (1994), 275–324. 4

[LL91] LUO Y., LUKÁCS G.: A boundary representation of form features and non-manifold solid objects. In *Solid Modeling Foundations and CAD/CAM Applications* (Austin, TX, June 1991), ACM Press. 22

[LL01] LEE S. H., LEE K.: Partial-entity structure: a fast and compact non-manifold boundary representation based on partial topological entities. In *Proceedings Sixth ACM Symposium on Solid Modeling and Applications* (Ann Arbor, Michigan, June 2001), ACM Press, pp. 159–170. 10, 12, 13, 22

[LLLV05] LAGE M., LEWINER T., LOPES H., VELHO L.: CHF: a scalable topological data structure for tetrahedral teshes. In *18th Brazilian Symposium on Computer Graphics and Image Processing (Sibgrapi 2005)* (Oct 2005), pp. 349–356. 15, 18

[LRC*02] LUEBKE D., REDDY M., COHEN J., VARSHNEY A., WATSON B., HUEBNER R.: *Level of Detail for 3D Graphics*. Morgan-Kaufmann, San Francisco, 2002. 23

[LT97] LOPES H., TAVARES G.: Structural operators for modeling 3-manifolds. In *Proceedings Fourth ACM Symposium on Solid Modeling and Applications* (May 1997), ACM Press, pp. 10–18. 15, 17

[Man87] MANTYLA M.: *An Introduction to Solid Modeling*. Computer Science Press, 1987. 1, 7, 8, 13, 22

[MH01] MCMAINS S., HELLERSTEIN C. S. J.: Out-of-core building of a topological data structure from a poly-

© The Eurographics Association 2007.

gon soup. In *Proceedings Sixth ACM Symposium on Solid Modeling and Applications* (2001), pp. 171–182. 10

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

[Muc93] MUCKE E.: *Shapes and Implementations in Three-Dimensional Geometry*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1993. 15, 17

[Nab96] NABUTOVSKY A.: Geometry of the space of triangulations of a compact manifold. *Commun. Math. Phys. 181* (1996), 303–330. 21

[NE04] NATARAJAN V., EDELSBRUNNER H.: Simplification of three-dimensional density maps. *IEEE Transactions on Visualization and Computer Graphics 10*, 5 (2004), 587–597. 17

[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. 4, 6

[PBCF93] PAOLUZZI A., BERNARDINI F., CATTANI C., FERRUCCI V.: Dimension-independent modeling with simplicial complexes. *ACM Transactions on Graphics 12*, 1 (January 1993), 56–102. 4, 6

[PH97] POPOVIC J., HOPPE H.: Progressive simplicial complexes. In *ACM Computer Graphics Proceedings Annual Conference Series, (SIGGRAPH '97)* (1997), ACM Press, pp. 217–224. 23

[PTL04] PESCO S., TAVARES G., LOPES H.: A stratification approach for modeling two-dimensional cell complexes. *Computers and Graphics 28* (2004), 235–247. 20, 23

[RC99] ROSSIGNAC J., CARDOZE D.: Matchmaker: manifold BReps for non-manifold R-sets. In *Proceedings Fifth Symposium on Solid Modeling and Applications* (New York (USA), 9–11 June 1999), Bronsvoort W. F., Anderson D. C., (Eds.), ACM Press, pp. 31–41. 20

[RO89] ROSSIGNAC J. R., O'CONNOR M. A.: SGC: a dimension-independent model for point-sets with internal structures and incomplete boundaries. In *Geometric Modeling for Product Engineering*, Wozny M. J., Turner J. U., Preiss K., (Eds.). Elsevier Science Publishers B. V. (North–Holland), Amsterdam, 1989, pp. 145–180. 20

[RO96] RENZE K. J., OLIVER J. H.: Generalized unstructured decimation. *IEEE Computational Geometry and Applications 16*, 6 (1996), 24–32. 23

[RSS01] ROSSIGNAC J., SAFONOVA A., SZYMCZAK A.: 3D compression made simple: Edge-Breaker on a Corner Table. In *Proceedings Shape Modeling International 2001* (Genova, Italy, May 2001), IEEE Computer Society. 7, 9

[Sam06] SAMET H.: *Foundations of Multi-Dimensional Data Structures*. Morgan-Kaufmann, August 2006. 7, 8

[THJ99] TROTTS I. J., HAMANN B., JOY K. I.: Simplification of tetrahedral meshes with error bounds. *IEEE Transactions on Visualization and Computer Graphics 5*, 3 (1999), 224–237. 23

[VG00] VELHO L., GOMES J.: Variable resolution 4-k meshes: concepts and applications. *Computer Graphics Forum 19*, 4 (2000), 195–214. 23

[VL97] VÉRON P., LÉON J. C.: Static polyhedron simplification using error measurements. *Computer-Aided Design 29*, 4 (1997), 287–298. 11, 14, 23

[VL01] VÉRON P., LÉON J.-C.: Using polyhedral models to authomatically sketch idealized geometry for structural analysis. *Engineering with Computers 17* (2001), 373–385. 23

[Wei88] WEILER K.: The radial-edge data structure: a topological representation for non-manifold geometric boundary modeling. In *Geometric Modeling for CAD Applications: Selected and Expanded Papers from the IFIP WG5.2 Working Conference, Rensselaerville, NY, USA, 12-16 May 1986*, Encarnacao J. L., Wozny M. J., McLaughlin H. W., (Eds.). Elsevier Science Publishers B. V. (North–Holland), Amsterdam, 1988, pp. 3–36. ISBN: 0444704167. 10, 11, 22

[Whi65] WHITNEY H.: Local properties of analytic varieties. In *Differential and Combinatorial topology, A Symposium in Honor of Marston Morse* (1965), Cairns S. S., (Ed.), Princeton University Press, pp. 205–244. 20

[YK95] YAMAGUCHI Y., KIMURA F.: Non-manifold topology based on coupling entities. *IEEE Computer Graphics and Applications 15*, 1 (January 1995), 42–50. 10, 12, 22

© The Eurographics Association 2007.