Computer-Aided Design 44 (2012) 1115–1126

Contents lists available at SciVerse ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Q-Complex: Efficient non-manifold boundary representation with inclusion topology

Long Zeng[a,*], Yong-Jin Liu[b], Sang Hun Lee[c], Matthew Ming-Fai Yuen[a]

[a] Department of Mechanical Engineering, Hong Kong University of Science and Technology, Hong Kong, China
[b] TNList, Department of Computer Science and Technology, Tsinghua University, Beijing, China
[c] Graduate School of Automotive Engineering, Kookmin University, Republic of Korea

ARTICLE INFO

Article history:
Received 11 July 2011
Accepted 3 June 2012

Keywords:
Inclusion topology
Quasi-cell-complexes
Q-Complex
Non-manifold
Topological data structure

ABSTRACT

The interpretation of auxiliary entities as boundary entities in previous non-manifold boundary (NMB) representations may change a model's intended topology and increase the complexity of the corresponding data structure. In this paper, entities appearing in a modeling process are classified into boundary and non-boundary entities. Non-boundary entities are usually embedded into embedding-space entities. These embedding relationships are described by inclusion topology. To support inclusion topology, a new mathematical framework—quasi-cell-complex, as well as a topological data structure—Q-Complex, are proposed. Quasi-cell-complex is an extension of cell-complex with inclusion topology supported. Q-Complex is an NMB representation, in which a new topological entity—embedder is created for inclusion topology and zone/disk is adopted to capture the complete adjacencies around a vertex. Thus, Q-Complex allows full adjacencies, incidence-ordering, and inclusion relationships to be derived, and the efficiency of most basic queries is several times faster than most state-of-the-art NMB representations, without increasing storage. Additionally, the benefits of inclusion topology for shape modeling and feature modeling are explored.

1. Introduction

Explicit boundary representation (B-rep) schemes decompose the boundary of a model into a set of cell entities and store their adjacencies and incidence-ordering (i.e. ordering between incident entities of a reference entity [1]) to accelerate traversal algorithms. Non-manifold boundary (NMB) representations supply a unified representation for the wireframe model, the surface model, the solid model, and their mixtures. However, most NMB representations treat all entities as boundary entities. This leads to at least two observed defects: topological error and data redundancy.

Treating all entities as boundary entities may erroneously alter the model's initial topology, leading to wrong design intent. As shown in Fig. 1(a), the vertex on the top face of the cube will be considered as an inner loop, similarly to the vertex at the bottom face. The auxiliary edge, which is meant to be a construction line, would be taken as a through hole. Thus, the genus of this cube in a typical NMB data structure will be 1. This is inconsistent with the design intent to input a construction line as an auxiliary edge while maintaining the topological integrity of the cube.

Likewise, the compactness of the data structure may be affected when treating non-boundary entities as boundary entities. For example, the top face of the ring has only one outer boundary loop (red in Fig. 1(b)). If six on-surface offset edges (blue) are added to the face as construction lines for downstream applications, this face will be decomposed into seven patches. Six of them will each have an outer and an inner boundary loop. However, these construction lines are only used to support the design process and should not be treated as boundary entities.

To better support the use of auxiliary data in a computer-aided-design process, which is necessary in NMB representations, a simple scheme is to categorize all entities into boundary and non-boundary entities. Boundary entities are the topological entities which contribute to a model's boundary; otherwise they are non-boundary entities. Auxiliary data, which are often used to support design operations, can be considered as a kind of non-boundary entities, embedded in other entities. For example, the vertices A and B (Fig. 1(a)) are embedded into the top and bottom faces respectively, and the auxiliary edge E (Fig. 1(a)) is embedded into the region bounded by the cube-faces. Thus, entities A, B, and E are not part of the cube's boundary and its topology is preserved. Similarly, if the on-face edges (Fig. 1(b)) are considered

* Corresponding author. Tel.: +852 2358 7189; fax: +852 2358 1543.
E-mail addresses: zelog@ust.hk (L. Zeng), liuyongjin@tsinghua.edu.cn (Y.-J. Liu), shlee@kookmin.ac.kr (S.H. Lee), meymf@ust.hk (M.M.-F. Yuen).

0010-4485/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2012.06.002

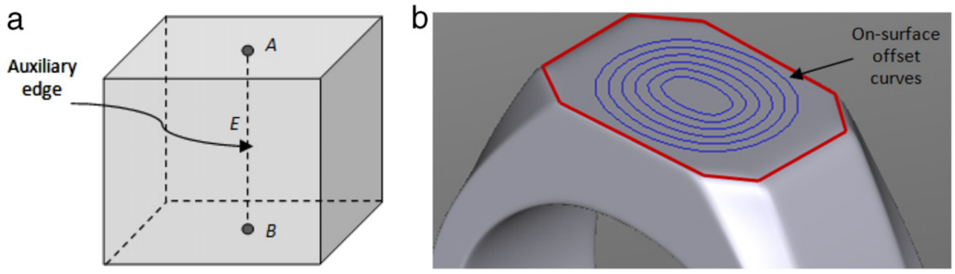


Fig. 1. Treating non-boundary entities as boundary entities (a) leads to topological error and (b) complicates the representation of the ring's top-face.

to be embedded into the top face, the compactness of the face representation would be also preserved.

To provide an embedding solution for non-boundary entities, a new mathematical framework—quasi-cell-complex as well as an NMB data structure—Q-Complex are proposed. The above mentioned embedding operation is formalized as inclusion topology, i.e. the point set of one entity is a subset of the point set of another entity. This differs from traditional cell-complexes where entities have no overlap. Inclusion topology is supported by quasi-cell-complex extended from cell-complex.

In order to represent quasi-cell-complex models, a new NMB data structure—Q-Complex is proposed. In Q-Complex, a new topological entity, embedder, is created for inclusion topology, and zone/disk entities are adopted to capture the complete adjacency around each vertex. Thus, Q-Complex allows full adjacency, incidence-ordering, and inclusion relations to be derived. As such, the efficiency of Q-Complex in handling most basic queries is several times faster than most existing NMB representations. Additionally, since only the frame of each face is stored, the memory is also efficient in most cases. In shape modeling, inclusion topology can help to prevent the model's original topology from being affected by massive non-boundary entities that are used to support the construction of the model. In feature modeling, inclusion topology can be used to capture the inter-feature relations.

The rest of the paper is organized as follows. Section 2 reviews the related work on data structures and mathematical models of non-manifold representations. A new mathematical model with inclusion topology—quasi-cell-complex is proposed in Section 3 and a hierarchical data structure—Q-Complex is presented in Section 4. Analysis and comparison are given in Section 5. Section 6 gives some potential applications of Q-Complex NMB data structure to shape modeling and feature modeling. The conclusion and future work are in Section 7.

2. Related work

In this section, background knowledge of mathematical models and their representations is introduced and related research work is reviewed.

2.1. Background

An abstract model of a physical object is a mathematical model, which allows study of a valid model using specific mathematical knowledge. A modeling space is a collection of abstract models. A representation is a set of symbols with syntactical rules. The collection of all syntactically correct representations is called a representation space. A representation scheme can be defined as a map or a function between the modeling space and the representation space [2].

In geometric modeling, an abstract model can be either manifold or non-manifold. A 2-manifold model M is a topological space where every point p has a neighborhood which is topologically equivalent to an open disk [3,4]. Any model which does not satisfy the above manifold property is called a non-manifold model.

Cell-complex is a common mathematical model for manifold or non-manifold models. It defines an abstract model as a finite collection of n-cells and the cells are disjointed with each other. An *n-cell* is a subset of n-dimensional space that is homeomorphic to an n-dimensional open ball. The corresponding 0-cell, 1-cell, 2-cell, and 3-cell are a vertex, edge, face, and region respectively. An edge can be a curved line and a face may be non-planar [5].

To facilitate the following discussion, the major abbreviations used in this paper are listed in Table 1.

Table 1
Abbreviations of major topological data structure.

Abbr.	Reference
RES	**R**adial **e**dge **s**tructure [6]
VBR	**V**ertex-based **b**oundary **r**epresentation [7]
CES	**C**oupling **e**ntity **s**tructure [8]
SGC	**S**elective **g**eometric **c**omplexes [9]
PES	**P**artial **e**dge **s**tructure [10]

2.2. Previous work on mathematical models

For manifold models, Requicha [2] introduced r-set as a mathematical model. An r-set model is a subset of Euclidean spaces, R^3. It is bounded, closed, regular, and semi-analytic. Another similar mathematical model—the manifold solid, proposed by Mantyla [11], can be understood as a special r-set with a closed 2-manifold boundary. The r-set premise was extended by Desaulniers and Stewart [12] to non-manifold r-sets. However, the condition of regularity cannot be satisfied in non-manifold geometric modeling.

There are usually two methods to handle non-manifold models, from which two different mathematical models are developed.

The first mathematical model is based on cell-complex theory, denoted as the *cell-complex model*, which adopts a finite number of n-cells to cover the model boundary [5,13]. A cell-complex is a collection of various dimensional cells. Thus, a valid cell-complex model can have a mixture of wireframe, surface, and solid models. Higher dimensional cells are bounded by lower dimensional cells. Cells are glued together without overlapping. Simplicial-complex is a special cell-complex where the faces are planar and the edges are straight segments. The simplicial-complex model is further extended to CW-complexes by Kase et al. [14] for multiple-material situations, where "C" stands for closure-finite and "W" for weak topology. Differently, Dicarlo et al. [15,16] provided a new representation using a chain-complex under field computations. A chain-complex model can be represented by a Hasse matrix where Euler operations can be performed by multi-linear matrix transformations. Thus, the topology, geometry, and physics may coexist in a single formalized framework.

The other mathematical model is based on stratification theory [17–19], denoted as the *stratified model*. Stratification theory decomposes a non-manifold model into a set of manifold components, i.e., a partition of a large point set into subsets which are various dimensional manifolds. These manifold components are denoted as strata. Mathematically, a valid stratification for a given model must fulfill three conditions: each stratum is a sub-manifold; each sub-manifold is smooth; and the stratum number is finite. Whitney stratification introduced by Gomes et al. [20] is a generalization of finite *CW*-complexes and simplicial-complex. In geometric modeling aspects, Whitney stratification is generally equivalent to the cell-complex model [21].

One common problem of the cell-complex model or stratified model is that all components are disjointed, i.e. not allowed to be overlapped.

2.3. Previous work on NMB data structures

The various representation schemes for non-manifold models can be classified into explicit and implicit schemes [18] that correspond to cell-complex models and stratified models respectively. *Explicit schemes* constitute a set of explicitly encoded strata plus their adjacency and incidence-ordering. *Implicit schemes* usually contain a set of strata and a set of functions to retrieve their topological relations. The cell-tuple structure [22], HC-Rep [17], and DiX [18] captured the incidence and incidence-ordering information of decomposed *n*-manifolds. In the combinatorial-map-based schemes [23–25], the strata are darts which are oriented edges. The cells are not explicitly represented, but can be obtained by functions and a delegated dart of the cell. In this section, only explicit NMB data structures are detailed since they are most related to the proposed *Q*-Complex data structure. Recent papers, such as compact mesh representations [26–28] and simplicial-complex based representations [29], provide additional information.

The topology in previous NMB representations usually relates to the adjacency between all topological entities. This terminology was first formally introduced in Weiler's RES [6,30] and is used in the following NMB representation, the VBR [7], CES [8], and PES [10]. To handle more general models with mixed dimensional entities and incomplete boundaries, Horvath's research group assumed phantom shells for wireframe and face group segments in a model. These imaginary shells are connected with the primary shell by new designed entities: join, spine, and palm [31]. Rossignac and O'Conner proposed the SGC data structure [9] which is composed of finite collections of mutually disjointed cells. SGC is topologically sufficient and storage efficient since it uses a simple incidence graph. However, it has no incidence-ordering information which is important to enhance the efficiency of practical algorithms.

Considering the incidence-ordering information, Yamaguchi and Kimura [8] identified three kinds of cyclic ordering among adjacency entities: the loop cycle (ordering of edges within a face), radial cycle (ordering of faces around an edge), and disk cycle (ordering of edges around vertex). RES proposed a set of use-entities, such as face-use, loop-use, edge-use, and vertex-use, which are derived from the corresponding topological entities: face, loop, edge, and vertex. These use-entities are useful in capturing the adjacency relationship between them for the complex non-manifold condition which happens at vertices, edges, and faces. However, the original design of RES is not storage efficient. For example, if a face is decomposed into two face-uses, the corresponding face boundary is also decomposed into two identical copies. Thus, PES only stores entities' frames to enhance the storage efficiency, i.e., each face-use has no real boundary, which is kept by the original face. In such a way, the PES can save about 50% of storage compared to the RES. Both RES and PES explicitly store the loop cycle and radial cycle.

The disk cycle around a non-manifold vertex is not handled in the RES and PES representations, leading to ambiguity when traversing around a non-manifold vertex. In order to overcome this drawback, Gursoz, et al. [7] proposed the VBR, in which the zone and disk topological entities are introduced to capture the adjacency and incidence-ordering information in the vertex neighborhood. VBR represents the three kinds of cycles explicitly and supplies sufficient information to derive all adjacency and incidence-ordering relationships. However, the VBR has an even higher storage cost than the RES. The CES can also achieve the same results by introducing six coupling entities to represent the neighborhoods and boundaries of basic topological entities.

A common limitation of all those NMB data structures is that all entities are considered as boundary entities. This problem is due to the cell-complex mathematical model in which the intersection of two arbitrary cells must be empty. That is, one entity is prohibited from being contained by other entities. Otherwise, the contained-entity will be considered as part of the boundary of the container-entity abiding to the conditions of cell-complex mathematical framework.

In this paper, inclusion topology, which allows one entity to be embedded into another entity, is introduced to solve the above mentioned problems which exist in most state-of-the-art NMB representations. To support inclusion topology, a new mathematical model—quasi-cell-complex and an efficient data structure—*Q*-Complex are proposed.

3. Quasi-cell-complex

In this section, the definition of the quasi-cell-complex mathematical model is given. A set of properties and the Euler–Poincarê formula for the quasi-cell-complex model are also explored.

3.1. Mathematical foundation

To make our discussion more precise, we start with some definitions.

Definition 1 (*Embedding Order*)**.** *Embedding order* is a total order on the topological entity set $T = \{\text{vertex, edge, loop, face, shell, region, body}\}$.

Total order is a binary relation which is transitive, anti-symmetric, and total. *Totality* means that any element-pair of the set T is mutually comparable. Thus, to compare two arbitrary entities in T conveniently, each is assigned with an integer value, denoted as the embedding order value (EOV), which is equal to the maximal dimensionality. The EOV of a vertex is 0 since it is 0-cell; an edge is 1; a face and a shell[1] are 2; a region and a body are 3.

Definition 2 (*Embedding Entity, Embedding-Space Entity, and Inclusion*)**.** *Embedding entity* is a topological entity which is embedded into another entity. *Embedding-space entity* is a topological entity where an embedding entity is embedded. *Inclusion* is a relation which indicates that the point set of an embedding entity is a subset of the point set of an embedding-space entity.

For example, if the intersection of two topological entities e_1 and e_2 is e_1, then e_1 is called the embedding entity, e_2 is called the embedding-space entity, and the relationship between e_1 and e_2 is described by an inclusion relation, which is denoted by the symbol "$\rightarrow$".

[1] A shell is a collection of faces, thus its maximal dimensionality is equal to the dimensionality of a face.

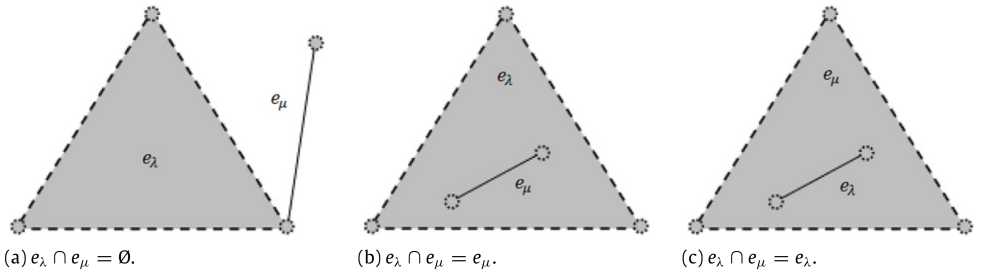

e_μ
e_λ
(a) $e_\lambda \cap e_\mu = \emptyset$.
e_λ
e_μ
(b) $e_\lambda \cap e_\mu = e_\mu$.
e_μ
e_λ
(c) $e_\lambda \cap e_\mu = e_\lambda$.

Fig. 2. Difference between cell-complex and quasi-cell-complex: without affecting the topology of the embedding-space entity, cell-complex only covers case (a) and quasi-cell-complex covers cases (a) to (c).

Remark 1. Inclusion is transitive since the embedding order is transitive. That is, if $e_1 \rightarrow e_2$ and $e_2 \rightarrow e_3$, then $e_1 \rightarrow e_3$.

Remark 2. When an entity is embedded, the EOV of the embedding entity is not greater than that of the embedding-space entity.

Given an embedding entity, there may be more than one entity acting as its embedding-space entities according to Remark 1.

Definition 3 (*Direct Inclusion*)**.** For a given embedding entity, *direct inclusion* is the inclusion between the embedding entity and the one which has the smallest EOV among all its embedding-space entities.

The cell-complex mathematical model [13] is restated as follows.

Definition 4 (*Cell-Complex*)**.** The *cell-complex* is defined as a finite set of n-cells, denoted as C, satisfying three conditions ($\wedge$ is an index set, $[e_\lambda]$ and d_{e_λ} are the closure and dimensionality of cell e_λ):

I. $C = \{e_\lambda | \lambda \in \wedge\}$.
II. If $d_{e_\lambda} = n + 1$ ($\lambda \in \wedge$), then $[e_\lambda] - e_\lambda \in C^n$, where $C^n = \{e_\mu | d_{e_\mu} \leq n, \mu \in \wedge\}$.
III. Suppose $e_\lambda,\ e_\mu \in C, \lambda \neq \mu$, then $e_\lambda \cap e_\mu = \emptyset$.

The first condition means that a cell-complex is a collection of n-cells. The second condition states that the boundary of a higher-dimensional cell is composed of some lower-dimensional cells. The third condition says that the intersection of any two cells from the cell-complex is empty, as shown in Fig. 2(a).

Actually, inclusion topology is universal, e.g. a 2-manifold can be considered as embedded in Euclidean space R^3. Inclusion topology allows an entity to be embedded into another entity. To support this, the cell-complex mathematical model is extended.

Definition 5 (*Quasi-Cell-Complex*). The *quasi-cell-complex* is defined as a finite set of n-cells, denoted as Q, satisfying the following three conditions ($\wedge$ is an index set, $[e_\lambda]$ and d_{e_λ} are the closure and dimensionality of cell e_λ):

I. $Q = \{e_\lambda | \lambda \in \wedge\}$.
II. If $d_{e_\lambda} = n + 1$ ($\lambda \in \wedge$), then $[e_\lambda] - e_\lambda \in Q^n$, where $Q^n = \{e_\mu | d_{e_\mu} \leq n, \mu \in \wedge\}$.
III. Suppose $e_\lambda,\ e_\mu \in Q,\ \lambda \neq \mu$, then $e_\lambda n e_\mu = \emptyset$ or e_λ or e_μ.

The first and second conditions are the same as those of Definition 4 with the exception of replacing the cell-complex by quasi-cell-complex. The third condition is more distinct. The relation between two arbitrary entities from quasi-cell-complex can be one of the three cases: disjointed (Fig. 2(a)), entity e_μ embedded in e_λ (Fig. 2(b)), or entity e_λ embedded in e_μ (Fig. 2(c)). However, if these two entities are from cell-complex, they must be disjointed (only Fig. 2(a)).

The relationship between a cell-complex model and a quasi-cell-complex model can be described by two theorems.

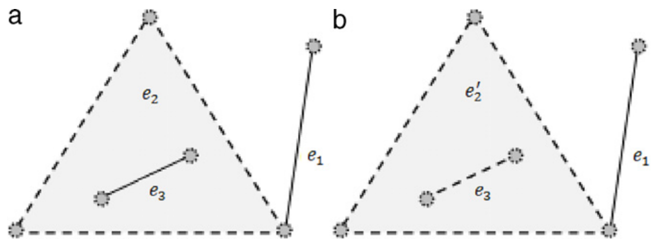


Fig. 3. A quasi-cell-complex (a) with $e_3 \rightarrow e_2$ is converted into a cell-complex (b) where e_3 is an inner boundary of e_2'.

Theorem 1. *A quasi-cell-complex model can be converted into a cell-complex model by treating the embedding entities as boundaries of the embedding-space entity.*

Proof. According to Definition 5, there are only two relation types between all entity pairs in a quasi-cell-complex, i.e. disjointed (Fig. 2(a)) or embedded (Fig. 2(b) and (c)). Thus, without loss of generality, we can choose three entities $\{e_1, e_2, e_3\}$ from a quasi-cell-complex Q, with $e_1 \cap e_2 = \emptyset$, $e_1 \cap e_3 = \emptyset$, and $e_3 \rightarrow e_2$, as shown in Fig. 3(a). That is, e_1 has no overlap with e_2 and e_3, and e_3 is embedded into e_2. Then, supposing C is an empty cell-complex initially, we only need to show that Q can be converted into a cell-complex by treating e_3 as a new boundary of e_2, to generate a new entity e_2'.

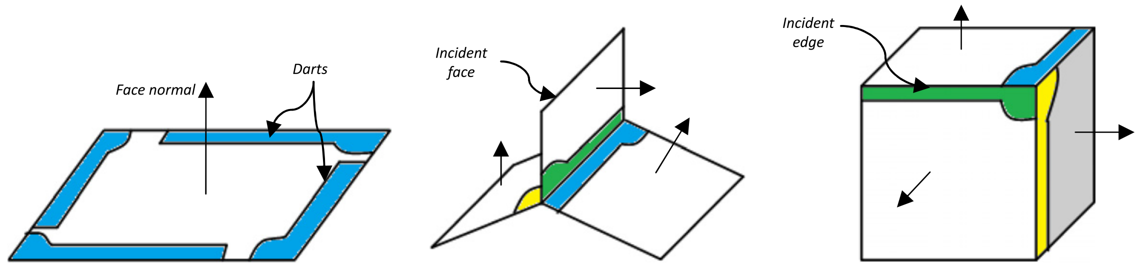
For the entity e_1, it can be added to C directly. This is because whatever entities of Q (e_2 or e_3) are added into C, e_1 always satisfies the three conditions of Definition 4 since the intersection between e_1 and others is empty ($e_1 \cap e_2 = \emptyset$ and $e_1 \cap e_3 = \emptyset$).

For entities e_2 and e_3, an operator is needed to convert their inclusion relation to separation. Since $e_3 \rightarrow e_2$, then the point set S_3 of e_3 is a subset of point set S_2 of e_2 (Definition 2 and Remark 1). The new entity e_2' is constructed by subtracting S_3 from S_2. Then, e_2' and e_3 can be added to cell-complex C and condition I of Definition 4 is satisfied naturally. Since $e_2' \cap e_3 = \emptyset$, condition III of Definition 4 is also satisfied. If S_3 is a proper subset of S_2, condition II of Definition 4 is satisfied since the point set S_2' is equal to $S_2 - S_3$. If not, e.g. a vertex embedded into another vertex, the new entity S_2' is an empty set and condition II of Definition 4 is also satisfied since the topological dimension of an empty set is -1 (see p. 147 of Ref. [32]).

Therefore, as shown in Fig. 3(b), the model $C = \{e_1, e_2', e_3\}$ is a valid cell-complex model converted from quasi-cell-complex model Q. □

From Theorem 1 we know that the modeling spaces of quasi-cell-complex and cell-complex are the same. The only difference is the treatment of the embedding entities.

Theorem 2. *A quasi-cell-complex model can be decomposed into a set of cell-complex models.*



(a) Loop cycle. (b) Radial cycle. (c) Disk cycle.

Fig. 4. Three types of cycle in non-manifold topology.

Proof. Supposing there is a quasi-cell-complex model Q, we prove this theorem by constructing a decomposition method.

Step 1. *Create a new model C. Select an arbitrary unclassified cell e_0 from the quasi-cell-complex. Add e_0 to C and to a stack L. Then tag the cell e_0 as classified.*

Step 2. *Pop a cell e_s from stack L. Retrieve all the unclassified direct adjacent cells,*[2] *ADJs, of the cell e_s. Then cells in ADJs are tagged as classified and added to model C and stack L.*

Step 3. *Repeat step* 2 *until the stack L is empty.*

Step 4. *While there still unclassified cells, repeat steps* 1 *to* 3.

According to the construction steps and Theorem 1, each newly created model C satisfies all the three conditions listed in Definition 4. Thus, each newly created C is a valid cell-complex. Thus, a quasi-cell-complex model can be decomposed into a set of cell-complexes. □

Taking the quasi-cell-complex in Fig. 1(a) for example, it will be decomposed into two cell-complexes. The cube is one cell-complex and the auxiliary edge is the other.

3.2. Euler–Poincarê formula for quasi-cell-complex

Since quasi-cell-complex is an extension of cell-complex, we recall the Euler–Poincarê formula for the cell-complex mathematical model given by Masuda [13]:

$$v - e + (f - r) - (V - V_h + V_c) = C - C_h + C_c \qquad (1)$$

where $v, e, f, r, V, V_h, V_c, C, C_h$, and C_c are the numbers of vertices, edges, faces, rings, volumes, holes through volumes, cavities in volumes, cell complexes, holes through cell complexes, and cavities in cell complexes, respectively. This formula can be derived from the Euler–Poincarê formula for finite cell-complexes [32].

Theorem 3. *The general Euler–Poincarê formula of Eq.* (1) *is also correct for a quasi-cell-complex model.*

Proof. Since a quasi-cell-complex model can be decomposed into a set of cell-complexes (Theorem 2), then the general Euler–Poincarê formula (Eq. (1)) can be applied to each cell-complex independently. The superimposing of Eq. (1) is still a valid equation. Therefore, the formula of Eq. (1) can also be applied to a quasi-cell-complex model. □

Taking the quasi-cell-complex in Fig. 1(a) for example, if it is considered as a cell-complex model, the entity numbers are $v = 10$, $e = 13$, $f = 6$, $r = 2$, $V = 1$, $V_h = 1$, $C = 1$, and $V_c = C_h = C_c = 0$. If it is considered as a quasi-cell-complex model, the entity numbers are $v = 10$, $e = 13$, $f = 6$, $V = 1$, $C = 2$, and $r = V_h = V_c = C_h = C_c = 0$. In each of the cases, the Euler–Poincarê formula Eq. (1) is satisfied.

[2] In the nine adjacency relationships between vertex, edge, and face, only VV is not direct adjacency since the adjacent vertices are not directly connected to the reference vertex.

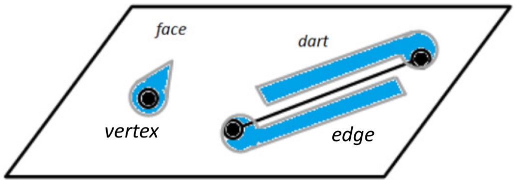


Fig. 5. Single vertex represented by a single dart while a wire edge is represented by two darts.

4. Q-Complex data structure

In this section, a set of new topological entities is introduced to support adjacencies, incidence-orderings, and inclusion topology. Their relationships are organized into a hierarchical data structure—Q-Complex.

4.1. Topological entities

In Q-Complex, the topological entities include body, region, shell, face, pface, edge, vertex, zone, disk, dart, and embedder. The definitions of zone, disk, dart, and embedder are detailed below while others, with similar usages to previous B-reps, are summarized in Table 2.

4.1.1. Dart

A *dart* is an edge-use by a face. It starts from a vertex and points along an edge. It is oriented and has an orientation flag. The flag is true if the dart's orientation is coincident with the edge's orientation. Otherwise, it is false.

Dart is used to represent explicitly three kinds of cycle: loop cycle, radial cycle, and disk cycle [8]. A loop cycle (Fig. 4(a)) is the representation of the *loop* entity and consists of consecutive darts; a radial cycle (Fig. 4(b)) represents the neighborhood of an *edge* and consists of a cyclic list of darts of its incident faces; a disk cycle (Fig. 4(c)) is the representation of the *disk* entity (detailed in Section 4.1.3) and contains a cyclic order of darts of its incident edges.

Special cases can also be represented by the dart entity. As shown in Fig. 5, a single vertex consists of a single dart. A wire edge includes two darts. A boundary edge also has two darts, only one with face associated. For a non-manifold edge, the dart number is the same as its incident face number. As shown in Fig. 4(b), the number of darts for this edge is three.

Cusp defined in VBR [7] can also represent the three types of cycle explicitly. The difference between cusp and dart is shown in Fig. 6. The cusp is used as the boundary of face-use, and the dart is used as the boundary of a face. Thus, the number of cusps is twice that of darts. This is why we can save about 50% of total storage size compared with the VBR.

Additionally, the dart and the p-edge entity defined in PES [10] are similar in concept, but are different in implementation. A p-edge is composed of a p-vertex, an edge, and an orientation flag.

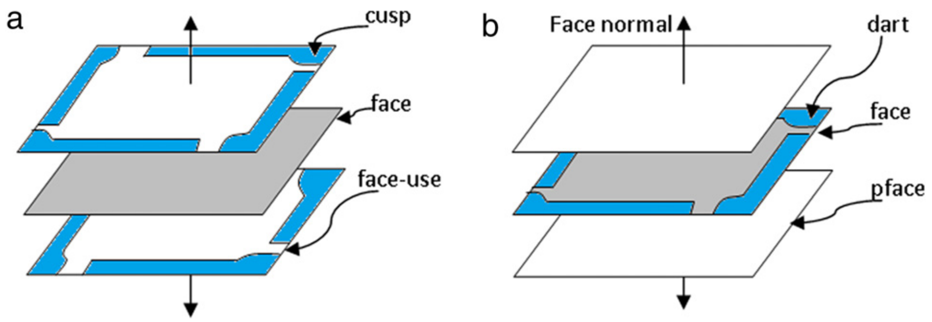


Fig. 6. (a) A cusp [33] is a boundary of face-use while (b) a dart is a boundary of a face.

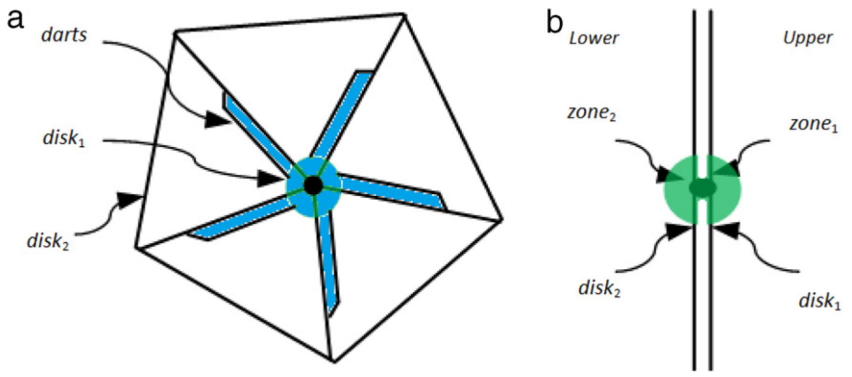


Fig. 7. Zones and disks of the vertex neighborhood (green): (a) front view; (b) side view. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

A dart only has an orientation flag of its parent edge to reduce the vertex redundancy, the two end-vertices are kept by the edge (see Appendix for more details).

4.1.2. Embedder

Embedder is a topological entity to capture the inclusion topology between an embedding entity and its direct embedding-space entity. It contains two pointers, one is the embedding entity and the other is the embedding-space entity (see Appendix).

Whether an entity should be considered as a boundary entity or a non-boundary entity depends on the designer's design intent. If the designer intends to add some construction data, an embedding operator will be called to make these auxiliary data non-boundary entities, such as the vertex on the top face in Fig. 1. With inclusion topology, the vertex is considered as an embedding entity and the face as an embedding-space entity. Thus, in Q-Complex, only an extra embedder is created to record the relationship between this vertex and the top face. The topology of the face is preserved and still equivalent to a disk. However, in previous NMB representations, such as PES, the vertex is treated as a boundary entity and represented by a single *p-edge*. A new *loop* is created and acts as an inner boundary of the top face. This makes the top face equivalent to a disk with a hole in it and its topology is changed.

4.1.3. Zone and disk

Zone and disk are adopted to capture the full topology information around a vertex, manifold or non-manifold. *Zone* is defined as a connected portion of the vertex neighborhood. The vertex neighborhood (the green area shown in Fig. 7(b))is the intersection between an open ball and the cell-complex which contains the given vertex. The ball radius is infinitesimal and its center is located at the given vertex. The vertex neighborhood is partitioned into zones by the entities incident to this vertex. Each zone is bounded by one or more disk-like boundaries, denoted as *disk*, i.e. *disk* is the boundary of zone. As shown in Fig. 7(b), the vertex neighborhood is portioned into $zone_1$ and $zone_2$, which are bounded by $disk_1$ and $disk_2$ respectively.

The concepts of zone and disk are first introduced in VBR. However, the content of disk in Q-Complex is more compact. In VBR, each disk is a cyclic list of cusps. However, in Q-Complex, it is represented by a cyclic list of darts emanating from this vertex. As shown in Fig. 7(a), $disk_1$ contains five darts and $disk_2$ has the same contents as $disk_1$. They mate with each other. Each disk maintains a *_mate* pointer (see code in Appendix). Thus, to save storage, only $disk_1$ contains a list of darts. If two neighboring disks are not mated exactly with each other, each disk would have a full list of its emanative darts.

4.2. Data structures

The hierarchical data structure to store all the above mentioned topological entities, adjacencies, and incidence-orderings is shown in Fig. 8. This data structure can represent non-manifold models with non-boundary entities which comply with the quasi-cell-complex model definition (Definition 5). Thus, it is named as Q-*Complex*.

The functionalities of each entity in Q-Complex are summarized in Table 2. More details about their implementation can be found in the Appendix. Most parent entities have a pointer to one of their child entities, while most child entities have a pointer to their parent entity. All sibling entities are singly linked, except the loop cycle and radial cycle where darts are doubly linked to enhance the accessing operator's efficiency.

Q-Complex takes advantage of both PES and VBR, i.e. only frames are stored and zone-disks are used to capture the complete adjacency around a vertex. However, Q-Complex differs from PES and VBR in two major aspects.

First, Q-Complex supports inclusion topology. With inclusion topology, boundary entities and non-boundary entities are treated differently and their relations are captured by embedders.

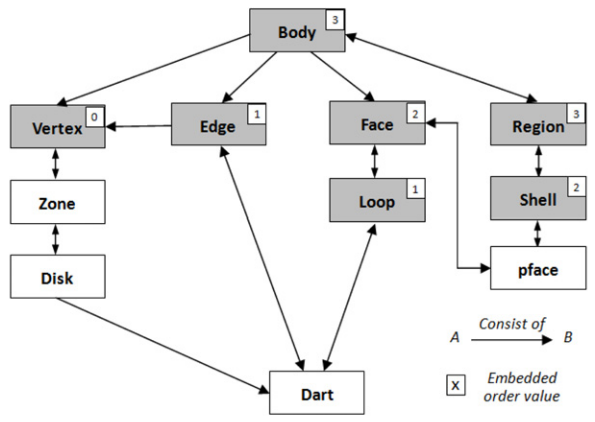


Fig. 8. Q-Complex data structure (the gray entities are those entities which can act as embedding entities and embedding-space entities, with EOV at the right-top corner).

Table 2
Summary of the basic topological entities in Q-Complex.

Entity	Functions
Body	*A repository of cell entities which satisfy* Definition 5, *containing one infinite region and an infinite shell at least.*
Region	*A connected portion of space, having multiple boundaries.*
Shell	*An oriented boundary of a region, consisting of a set of pfaces*
Face	*A bounded portion of a surface, having multiple boundaries.*
Pface	*A face-use without real boundary.*
Loop	*A connected boundary of a face, consisting of a set of darts.*
Edge	*A bounded portion of a curve, having two end vertices and orientation.*
Vertex	*A unique point in space.*
Dart	*An oriented edge-use.*
Zone	*A connected portion of vertex neighborhood, bounded by disks.*
Disk	*Boundary of a zone, consisting of darts.*
Embedder	*A topological entity supporting inclusion topology.*

Although the embedding cases are complicated, they must satisfy Definition 1 and Remark 2. Thus, the embedder is not shown explicitly in the Q-Complex configuration. The entities which can act as embedding entities and embedding-space entities are shaded and their EOV is positioned at the right-top corner (Fig. 8).

Secondly, the storage of Q-Complex is only about 40% of that of VBR.[3] To reduce data redundancy, the boundary of pface (face-use) and dart (edge-use) is not stored, which can be derived from their parent face and edge implicitly. Thus, the number of loops and darts is only half of that of VBR. Likewise, the content of disks is also optimized by exploring the mating relation between neighboring disks while each disk in VBR has a full cyclic list of cusps around the reference vertex. The speed of traversing Q-Complex is faster than that of PES since the three types of cycle are stored explicitly in Q-Complex (see Section 5.2).

5. Analysis of Q-Complex data structure

Q-Complex is an explicit representation which explicitly stores various dimensional cells and their mutual relations. The storage and query efficiency of Q-Complex are analyzed and compared with those of PES which is the most storage-efficient NMB data structure.

To facilitate the discussion, the following symbols are adopted. Let R denote the region, S the shell, F the face, L the loop, E the edge, V the vertex, Fu the face-use, Lu the loop-use, Eu the edge-use, Pf the pface, Pe the p-edge, Pv the p-vertex, Dt the dart, Z the zone, Dk the disk, Ed the embedder, A_i an A entity, $\{A_i\}$ a set of A entities, BA_i the number of B entities adjacent to an A_i entity, and $A_i \rightarrow \{B_i\}$ retrieval of all B entities adjacent to an A_i entity.

5.1. Topology analysis

In quasi-cell-complex, a model's topology mainly concerns the adjacencies, the incidence-orderings, and the inclusion among all topological entities.

Q-Complex is a data structure that meets the adequacy requirements as all the 36 adjacency relationships mentioned in RES [6] can be derived from the configuration shown in Fig. 8. Two close paths can be found in the configuration. One starts from a vertex, and leads to its zones, its disks, all the darts around it, and then back to the vertex. The other starts from the body, and leads to its faces, pfaces, shells, regions, and back to the body. The graph shown in Fig. 8 is connected with several bi-directional links. For any two entities in this data structure, there is always a path connecting them.

Q-Complex supplies full incidence-ordering information around a non-manifold vertex or edge. As shown in Fig. 9(a), the non-manifold vertex v_1 is decomposed into several zones. Fig. 9(b) shows a schematic view of the neighborhood of v_1: z_1 (blue) bounded by the inner boundary of A, z_2 bounded by the outer boundary of A and C (yellow), z_3 bounded by the inner boundary of B (green), and z_4 (red) bounded by the outer boundary of B and the inner boundary of C. The neighboring zone of z_3 is z_4; the neighboring zones of z_4 are z_2 and z_3. All such ordering information can be derived from the zone-disk entities around a vertex (see the code in Appendix). However, it fails PES. As shown in Fig. 9(c), v_1 is only decomposed into three p-vertices in PES. Each p-vertex has a pointer to its parent vertex v_1 and v_1 only points to one of them. Thus, the ordering between components A, B, and C is lost in PES.

Q-Complex supports inclusion topology while PES cannot. Inclusion topology allows one entity to be embedded into another entity. With inclusion topology, the original design intents can be preserved, as with the auxiliary edge in Fig. 1(a). Thus, the model's global topological property, such as genus, is kept. With inclusion topology, the relative position between components in quasi-cell-complex can be stored. This is useful in feature modeling which goes beyond solid modeling (see Section 6.2).

5.2. Query efficiency analysis

In order to evaluate the time efficiency of a data structure, the response time of all basic query functions is an important criterion. A query procedure returns all topological entities of a specific type for a given reference entity. BA_i is a query where A_i is the given reference entity and it returns all B entities which are adjacent to A_i.

In existing explicit non-manifold B-rep, there are six common basic topological entities: region, shell, face, loop, edge, and vertex. Thus, there are in total 36 possible adjacency relationships. As the matrix in Table 3 shows, the first column gives the reference entities and the first row gives the adjacent entity set for each reference entity.

The most important criterion for estimating the time cost of a query is the number of records and field accesses since they may require database access or at least a main memory access. This evaluation method has been widely used in previous works [10,34]. It is adopted to compare the efficiency between the PES and Q-Complex data structures (note: it is allowed to visit an entity more than once during the traversal in a basic query function unless otherwise stated).

The time complexity of basic query procedure FV_i is detailed to show the specific computation. FV_i returns all adjacent faces around the given vertex V_i. In Q-Complex, the vertex's complete

[3] The storage cost of VBR is published in PES, computed using the same method.

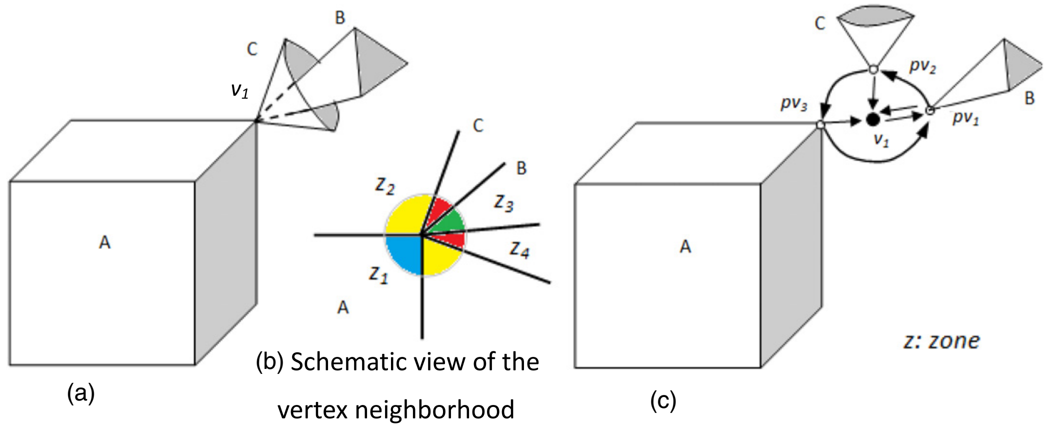


Fig. 9. Representation of a non-manifold vertex: (a–b) *Q*-Complex stores the order explicitly among components *A*, *B*, and *C* by zone-disk entities, while (c) PES fails.

Table 3
The number of field and record (in brackets) accesses for 36 basic queries of *Q*-Complex (*left of the slash*) and PES (*right of the slash*) (red: *Q*-Complex slower than PES; blue: *Q*-Complex equal to PES; black: *Q*-Complex faster than PES).

	$\{R_i\}$	$\{S_i\}$[a]	$\{F_i\}$	$\{L_i\}$	$\{E_i\}$	$\{V_i\}$
R_i	4(3)/5(3)FuR$_i$	1(1)/1(1)SR$_i$	2(1)/3(1)FuR$_i$	1(1)/1(1)LuR$_i$	2(1)/3(1)EuR$_i$	4(4)/4(2)EuR$_i$
S_i	1(1)/ 1(1)	3(2)/4(2)FuS$_i$	2(1)/3(1)FuS$_i$	1(1)/1(1)LuS$_i$	2(1)/3(1)EuS$_i$	4(2)/4(2)EuS$_i$
F_i	6(4)/ 6(5)	4(2)/ 4(3)	5(4)/8(5)DtF$_i$	1(1)/1(1)LF$_i$	2(1)/3(1)DtF$_i$	4(2)/3(2)DtF$_i$
L_i	7(5)/ 7(6)	5(3)/ 5(4)	1(1)/ 1(1)	4(3)/6(3)DtL$_i$	2(1)/3(1)DtL$_i$	4(2)/3(2)DtL$_i$
E_i	9(7)/9(7)DtE$_i$	7(5)/7(5)DtE$_i$	3(2)/3(2)DtE$_i$	2(1)/2(1)DtE$_i$	3(2)/5(3)DtE$_i$	2(1)/4(2)
V_i	9(7)/13(8)DtV$_i$	7(5)/11(7)DtV$_i$	3(2)/7(4)DtV$_i$	2(2)/6(3)DtV$_i$	3(2)/5(3)DtV$_i$	5(2)/5(3)DtV$_i$

[a] The miscalculation of the query access time for $R_i \rightarrow \{S_i\}$ in PES is corrected here.

```
Faces-of-vertex ($v_i$, f_list)
1 For each zone z at $v_i$ do
2     For each disk dk at z do
3         For each dart dt in dk do
4             l = dt->_loop
5             If l is not null then
6                 f = l->_face
7                 add f to f_list
```

Fig. 10. Query procedure for finding all adjacent faces for a given vertex.

adjacency information is described by zones and disks. The whole procedure consists of three nested loops for zones, disks, and darts (line 1 to line 3 of Fig. 10). For each *dart* in a *loop*, the *_loop* field of a *dart* is accessed at line 4 to obtain the parent *loop* and the *_face* field of a *loop* is accessed at line 6 to obtain the parent *face* at line 6.

Therefore, if ZV_i, DkV_i, and DtV_i denote the numbers of zones and disks around the reference vertex V_i, the total number of field accesses N_f is

$$\begin{aligned} N_f &= \sum_{k=0}^{ZV_i}\left(1+\int_{j=0}^{DkZ_k}(1+3DtDk_j)\right) \\ &= \int_{k=0}^{ZV_i}(1+DkZ_k+3DtZ_k) \\ &= (ZV_i+DkV_i+3DtV_i). \end{aligned} \quad (2)$$

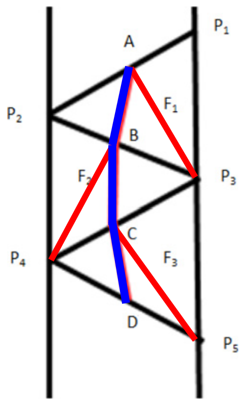
Thus, the dominant term of this query procedure is $3DtV_i$ since the dart number around a vertex is usually several times the number of zones and disks. Compared to $7DtV_i$ (the dart number is equal to the *p*-edge number) of the same query procedure in PES, the query speed of *Q*-Complex is about two times faster than that of PES because all the darts around the vertex are stored explicitly. In addition, the *p*-face in PES has multiple child entity types due to the special cases for on-surface vertex and on-surface edge. However, they are considered as embedding entities when inclusion topology is introduced.

The total number of record accesses can be counted in the same manner. In the innermost loop, three records, dart, loop and face, are accessed as many times as the number of darts around the given reference vertex. Thus, the dominant term of record accesses is $3DtV_i$. Compared to $4DtV_i$ of the same query procedure in PES, the time cost of this query procedure in *Q*-Complex is faster than that in PES.

The counts of the field and record accesses for all 36 queries of *Q*-Complex and PES are investigated and summarized in Table 3. For comparison, the count of each adjacency query procedure of PES and *Q*-Complex is expressed in the counting variables from our representation, except the adjacency relationships of shells and regions which are expressed by counting variables from the RES. This is because the corresponding variables of PES and *Q*-Complex cannot be used as correct counts if a shell includes laminar faces. This conversion is based on the relations of the topological entity number among RES, PES, and *Q*-Complex: $\#(Fu) = \#(Pf)$, $\#(Eu) = 2 \bullet \#(Pe) = 2 \bullet \#(Dt)$, and $\#(Lu) = 2 \bullet \#(L)$.

As shown in Table 3, *Q*-Complex is faster than PES for 22 basic queries, slower for only 3 queries, and has the same speed for the other 11 queries. When the vertex is a reference entity, the accessing speed in *Q*-Complex is several times that in PES. This is because PES only stores partial adjacency for a vertex neighborhood.

(a) Triangular mesh model.



(b) Detailed view.

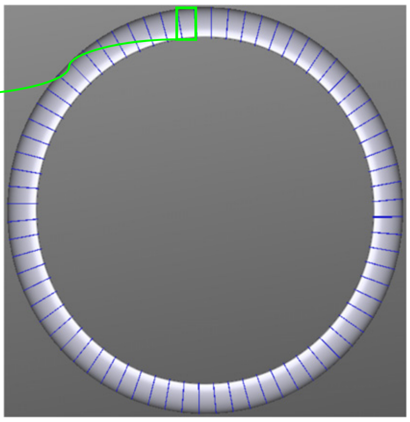

(c) With 72 curves embedded.

Fig. 11. A ring model for storage comparison between PES and Q-Complex.

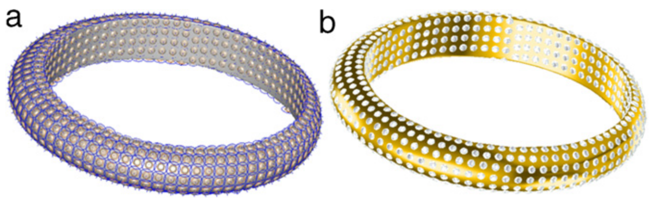


Fig. 12. The embedding curves shown in Fig. 11(c) are used for diamond setting in (a) smooth display and (b) rendering with texture.

5.3. Storage analysis

The storage cost of Q-Complex is also compared with that of PES. To facilitate the comparison, we adopt the same assumptions as PES:

- The length of a pointer is four bytes. One byte is the minimum storage unit. Thus, if there are multiple flags in one class and their summation is less than one byte, their storage is counted as one byte.
- Only topological data are counted. The attribute in each entity is ignored, and the face/edge/vertex lists in the *Body* class are also omitted since they are redundant data, used only for rendering.
- The storage estimation of each data structure is computed based on the original code.

With these assumptions, the total storage cost, S, of a data structure, D, is calculated by multiplying the number of each entity type with the size of the corresponding entity type, and summing them together. That is,

$$S(D) = \sum_{\tau_i \in D} \#(\tau_i) \cdot |\tau_i| \tag{3}$$

where τ_i is the ith entity type in data structure D; $\#(\tau_i)$, $|\tau_i|$ are the number and size of the entity type τ_i. For example, the size of the *Zone* class (see the code in the Appendix) of Q-Complex is 12 bytes since there are only three pointers.

In some cases, the number of each entity type τ can be expressed as a function of the number of an entity using statistical data, such as *face*. Then Eq. (2) can be rewritten as

$$S(D) = \sum_{\tau_i \in D} F_i(f) \cdot |\tau_i| \tag{4}$$

where f is the face number in a model, the function $F_i(f)$ means that the number of τ_i is a function of the face number.

The investigation by Wilson [34] showed the numbers of loops, edges, and vertices are approximately f, $3f$, and $2f$ respectively. Each *loop* contains about six edges and each *vertex* emanates three edges. When the face number of a triangular mesh (abbr. *Tri-mesh*) is large (usually greater than 10 k), the statistical numbers of loops, edges, and vertices are f, $1.5f$, and $0.5f$ respectively. That is each loop contains three edges and each vertex has six incident edges. Based on this observation, the storage cost of a data structure can be expressed as a function of a single variable (according to Eq. (4)). As shown in Table 4, the storage costs of PES and Q-Complex are compared.

Q-Complex is not only faster than PES, but also more compact in the case of a model with auxiliary data. Taking the ring model (Fig. 11(a)) for example, normally, the storage cost of PES is about 89% that of Q-Complex (shown in the 5th row of Table 4). However, when the 72 construction curves[4] are added to this ring model, as shown in Fig. 11(c), the storage of Q-Complex is less than that of PES (computed from data in the 6th and 7th rows of Table 4).

Since the ring model is a triangular mesh, the intersection between each embedding curve and the ring mesh model should be computed for both PES and Q-Complex. As shown in the detailed view in Fig. 11(b), line segments AB, BC, and CD are the intersections between one curve and the underlying triangles $P_1P_2P_3$, $P_2P_3P_4$, and $P_3P_4P_5$. In PES, the ring mesh model is re-triangulated according to the intersection. For example, the triangle $P_1P_2P_3$ is triangulated into three triangles P_1AP_3, ABP_3, and AP_2B. Thus, the numbers of faces, edges, and vertices are increased by two, four, and two respectively. The total entity number is summarized in the 6th row of Table 4. In Q-Complex, the new created entities are considered as embedding entities. Vertex A is embedded in edge P_1P_2 and edge AB is embedded in face $P_1P_2P_3$. Thus, the face number is preserved. The newly created entity number in Q-Complex is listed in the 7th row of Table 4.

6. Application of Q-Complex

The Q-Complex data structure is tested in a prototype system *JewelCADPro* [35]. In *JewelCADPro*, Q-Complex serves as a non-manifold topological data structure for B-rep. Since the inclusion topology is supported, some new modeling features become feasible in *JewelCADPro* and they are discussed in this section.

[4] The "curve" term used in this paper except in Table 2 consists of connected line segments which are embedded.

Table 4
Storage comparison between PES and Q-Complex (*the storage costs of Wilson and Tri-mesh are computed using* Eq. (4), *the others are computed*[a] *using* Eq. (3)).

	Number of Entities										Size (bytes)	
	$B/R/S$	$F(L)$	E	V	Pf	Pe	Pv	Dt	$Z(Dk)$	Ed	PES	Our
Wilson [33]	1/2/3	f	$3f$	$2f$	$2f$	$6f$	$2f$	$6f$	$4f$	0	$68 + 303f$	$88 + 376f$
Tri-mesh	1/2/3	f	$1.5f$	$0.5f$	$2f$	$3f$	$0.5f$	$3f$	f	0	$68 + 169.5f$	$88 + 195f$
Fig. 11(a)	1/2/3	24,024	36,365	12,341	48,048	72,730	12,341	72,730	24,682	0	4101,417	4605,444
Fig. 11(c)(PES)	1/2/3	35,544	53,645	18,101	71,088	107,290	18,101				6054,057	
Fig. 11(c)(Our)	1/2/3	24,024	42,125	18,101	48,048			84,250	36,202	11,520		5892,348

[a] Since Tri-mesh is 2-manifold and has six edges meeting a vertex, the size of normal-disk is 36 bytes, and mate-disk is 12 bytes.

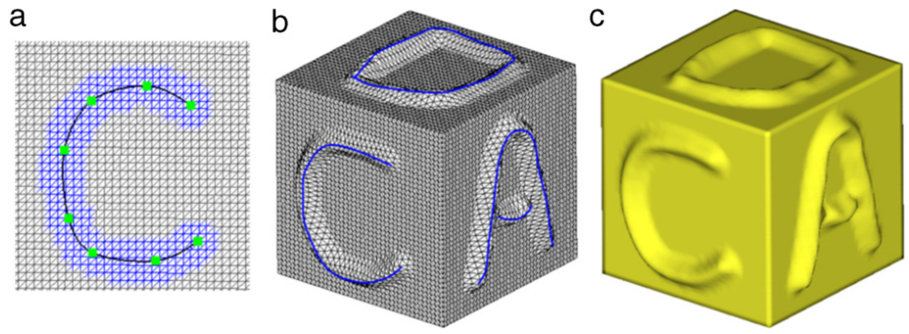
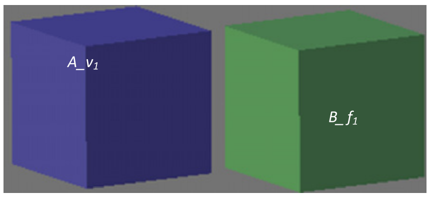
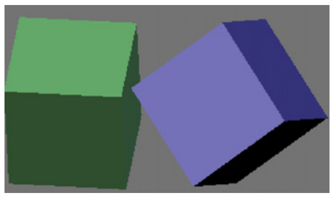


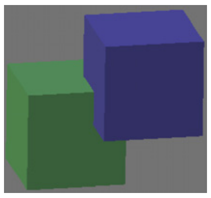
Fig. 13. Embedding curves are used for shape control: (a) "C" is embedded in the cube's front face; three characters "*C*", "*A*", and "*D*" are displayed in (b) mesh mode and in (c) smooth mode.



(a) Feature model.



(b) Vertex-coincidence.



(c) Vertex-inclusion.

Fig. 14. Feature relations are supported by inclusion topology in Q-Complex.

6.1. Shape modeling with Q-Complex

Since inclusion topology allows one entity to be embedded into another entity, auxiliary data can be stored separately without losing their relationships. This gives Q-Complex a significant advantage in shape modeling with assistant data, such as shape understanding by on-face iso-contours [36] and shape modeling by control curves [37].

Fig. 12 shows an example where embedding curves are used for diamond setting. The ring model has massive auxiliary entities which are stored in Q-Complex as embedding entities (Fig. 11(c)). The diamonds are positioned along the embedded curves. The diamond size is controlled by the width between two neighboring curves. The diamond orientation is determined by the underlying triangle's orientation. Such a task requires that the embedding entities and embedding-space entities be easily retrieved. In Q-Complex, the embedding entities, embedding-space entities, and their inclusion relation are all stored explicitly. Thus, they can be easily identified.

Fig. 13 is another example where embedding curves are applied to shape control. When a control curve is drawn, it is directly stored in Q-Complex without the re-meshing step (Fig. 13(a)). This is different from FiberMesh [38], where each newly created control curve is used to remesh the model. The remesh operation will reduce the efficiency of the drawing process. This is obvious especially when the model is complex. In Q-Complex, the control curve can be further edited by moving its handles (green points in Fig. 13(a)). Additionally, the region of interest (ROI) area (blue area displayed in Fig. 13(a)) of a control curve can be adjusted since Q-Complex also stores the relationship between a control curve and its embedding-space faces. Using this technique, three characters "*C*", "*A*", and "*D*" are constructed on a cube using these embedding curves, as shown in Fig. 13(b) and (c).

6.2. Feature modeling with Q-Complex

Feature modeling is an advanced technique going beyond solid modeling since it integrates geometrical and semantic information across different engineering domains to support product life-cycle modeling [39]. Feature modeling can benefit from Q-Complex if it is adopted to represent the geometric data of a feature model. Besides a feature's geometric information, feature relation is another important aspect in feature modeling, such as coincidence, touch, intersection, and inclusion. All these feature relations can be well represented by the inclusion topology in Q-Complex. No extra structures need to be added.

As shown in Fig. 14(a), a feature model *FM*, represented by Q-Complex, consists of two simple cubical features A and B. The relative position between feature A and feature B can be captured by inclusion topology. In Fig. 14(b), vertex A_v_1 of feature A coincides with face B_f_1 of feature B; this can be stored by creating an *embedder*. The *_entity* field of this *embedder* points to vertex A_v_1 of feature A and the *_space* field points to face B_f_1 of feature B. The *embedder* is held by face B_f_1. Thus, the touch relation between feature A and feature B is described as an inclusion relation between vertex A_v_1 and face B_f_1.

In Fig. 14(c), vertex A_v_1 of feature A is contained by the cube-region of feature B. Similarly, a new *embedder* is created, whose *_entity* field points to vertex A_v_1 of feature A and *_space* field points to the cube-region of feature *B*. The *embedder* is held by the cube-region. Thus, the partial containment relation between feature *A* and feature *B* is described as an inclusion relation between the vertex A_v_1 and the cube region of feature *B*.

7. Conclusion and future work

This paper introduced inclusion topology to preserve a model's original topology and keep the simplicity of the original data structure, without sacrificing storage. Inclusion topology allows one entity to be embedded into another entity. To support this, a mathematical framework—quasi-cell-complex was proposed which is an extension of the classical cell-complex framework. The major difference between the quasi-cell-complex model and the cell-complex model is that the former treats non-boundary entities as embedding entities while the latter considers them as boundary entities. However, a quasi-cell-complex model can be converted or decomposed into cell-complexes.

Furthermore, a new data structure—*Q*-Complex, was proposed to represent the quasi-cell-complex model. *Q*-Complex is an NMB data structure. The topological entity—embedder is proposed for inclusion topology; zone/disk is adopted to capture the complete adjacencies around a vertex, and darts are used to represent edge-use. Thus, adjacency, incidence-ordering, and inclusion can be derived from *Q*-Complex, and its efficiency for most basic queries is several times faster than the efficient data structure PES, without increasing storage. These distinct features make *Q*-Complex an attractive data structure when compared with corresponding state-of-the-art NMB data structures. In addition, by supporting inclusion topology, *Q*-Complex not only preserves the design intent topology of shape models, but also captures the inter-feature relations in feature modeling.

There are still many aspects that deserve to be studied in the future. Firstly, high-level operations like offsetting operations as well as a set of low-level topological operators like Euler operators need to be designed and implemented to facilitate the modeling process and to guarantee the integrity of the *Q*-Complex model. Compared to the existing data structures, more steps are needed to maintain the topology because the inclusion topology is additionally stored in *Q*-Complex [40]. Secondly, a data exchange interface should be designed to exchange data between *Q*-Complex and other CAD software. Thirdly, a complete geometric modeling system based on *Q*-Complex with its applications to shape modeling and feature modeling is also our future work.

Acknowledgment

The authors would like to thank Dr. Masuda [13] for his detailed explanation of the Euler–Poincarê formula mentioned in Eq. (1). We are also grateful to the anonymous reviewers for their fruitful comments, especially to one of the reviewers who provided us with an important reference. This work is partially supported by Jewellery CAD/CAM Ltd. [7], the Natural Science Foundation of China (Project Number 60970099), and the National Basic Research Program of China (Project Number 2011CB302202). The first author is also partially supported by Hong Kong Innovation and Technology Fund (ITF) UIM/210. The second author is partially supported by the 863 program of China (2012AA011801), the program for NCET and TNList Cross-discipline Foundation. The third author is supported by Korea Sanhak Foundation (2011).

Appendix. Implementation of *Q*-Complex with classes in C++

```
typedef std::vector ⟨Face∗⟩ FaceList
typedef std::vector ⟨Edge∗⟩ EdgeList
typedef std::vector ⟨Vertex∗⟩ VertexList
typedef std::vector ⟨Dart∗⟩ DartList
class Entity
{
Attribute *_attribute;
}
class Body: public Entity
{
Body *_next; //next body
Region *_region; //one of the regions in this body
FaceList facelist; //list of face pointers
EdgeList edgelist; //list of edge pointers
VertexList vertexlist; //list of vertex pointers
}
class Region: public Entity
{
Body *_body; //parent body
Region *_next; //next region in the parent body
Shell *_shell; //one of the shells in this region
Embedder *_embedder; //one of the embedders
}
class Shell: public Entity
{
Region *_region; //parent region
Shell *_next; //next shell in the parent region
Pface *_pface; //one of the pfaces in this shell
}
class Face: public Entity
{
Pface *_pface; //one of the pfaces of this face
Loop *_loop; //one of the loops of this face
Embedder *_embedder; //one of the embedders
}
class Pface: public Entity
{
Face *_face; //parent face
Shell *_shell; //parent shell
Pface *_next; //next pface in parent shell
Pface *_mate; //next pface in parent face
}
class Loop: public Entity
{
Face *_face; //parent face
Loop *_next; //next loop in parent face
Dart *_dart; //one of the darts in this loop
}
class Edge: public Entity
{
Dart *_dart; //one of the edge darts
Vertex *_start; //start vertex
Vertex *_end; //end vertex
Embedder *_embedder; //one of the embedders
}
class Vertex: public Entity
{
Zone *_zone; //oen of the zones around this vertex
Embedder *_embedder; //one of the embedders
}
class Zone: public Entity
{
Vertex *_vertex; //parent vertex
Zone *_next; //next zone in parent vertex
Disk *_disk; //one of the disks in this zone
}
class Disk: public Entity
{
Zone *_zone; //parent zone
Disk *_mate; //mate disk
Disk *_next; //next disk in the parent zone
DartList dartList; //a list of dart pointers
}
class Dart: public Entity
{
Orient orient; //1: has the same direction as parent edge
Edge *_edge; //parent edge
Loop *_loop; //parent loop
Dart *_prev_radial; //previous dart in radial order
Dart *_next_radial; //next dart in radial order
Dart *_prev_loop; //previous dart in parent loop
Dart *_next_loop; //next dart in parent loop
}
class Embedder: public Entity
{
Entity *_entity; //embedded entity
Entity *_space; //embedding-space entity
Embedder *_next; //next embedder in this embedding-space
entity
}
```

References

[1] Luo Y, Lukacs G. Incidence relationships: kernel concept in combinatorial topology. Computer Graphics and Mathematics 1992;129–46.
[2] Requicha AA. Representation for rigid solids: theory, methods, and systems. Computing Surveys 1980;12:437–64.
[3] Mantyla M. An introduction to solid modeling: rockville. Md: Computer Science Press; 1988.
[4] Takala T. A taxonomy on geometric and topological models. Computer Graphics and Mathematics 1992;147–71.
[5] Masuda H. et al. A mathematical theory and applications of non-manifold geometric modeling. In: IFIP WG 5.2/GI international symposium on advanced geometric modeling for engineering applications. Berlin (Germany); 1989. pp. 89–103.
[6] Weiler K. The radial edge structure: a topological representation for non-manifold geometric boundary modeling. In: Geometric modeling for CAD applications. 1986. p. 3–36.
[7] Gursoz EL, Choi Y, Prinz FB. Vertex-based boundary representation of non-manifold boundaries. In: Geometric modeling for product engineering. 1990. p. 107–30.
[8] Yamaguchi Y, Kimura F. Nonmanifold topology based on coupling entities. IEEE Computer Graphics and Applications 1995;15:42–50.
[9] Rossignac J, O'Connor MA. SGC: a dimension-independent model for pointsets with internal structures and incomplete boundaries. In: Geometric modeling for product engineering. 1989. p. 145–80.
[10] Lee SH, Lee K. Partial entity structure: a compact non-manifold boundary representation based on partial topological entities. Journal of Computing and Information Science in Engineering 2001;1:356–65.
[11] Mantyla M. A note on the modeling space of Euler operators. Computer Vision, Graphics, and Image Processing 1984;26:45–60.
[12] Desaulniers H, Stewart NF. An extension of manifold boundary representations to the r-sets. ACM Transactions on Graphics 1992;11:40–60.
[13] Masuda H. Topological operators and Boolean operations for complex-based nonmanifold geometric models. Computer-Aided Design 1993;25:119–29.
[14] Kase K, Teshima Y, Usami S, Kato M, Yamazaki S, Ito M, Makinouchi A. Volume CAD–*CW*-complexes based approach. Computer-Aided Design 2005; 37:1509–20.
[15] DiCarlo A, Milicchio F, Paoluzzi A, Shapiro V. Chain-based representations for solid and physical modeling. IEEE Transactions on Automation Science and Engineering 2009;6:454–67.
[16] DiCarlo A, Milicchio F, Paoluzzi A, Shapiro V. Solid and physical modeling with chain complexes. Presented at the Proceedings of the 2007 ACM symposium on solid and physical modeling. Beijing, China; 2007.
[17] Pesco S, Tavares G, Lopes H. A stratification approach for modeling two-dimensional cell complexes. Computers and Graphics (Pergamon) 2004;28: 235–47.
[18] Gomes AJP. A concise b-rep data structure for stratified subanalytic objects. In: Eurographics symposium on geometry processing. Aachen, Germany; 2003. pp. 83–93.
[19] Shapiro V. Maintenance of geometric representations through space decompositions. International Journal of Computational Geometry and Applications 1997;7:21–56.
[20] Gomes A, Middleditch A, Reade C. A mathematical model for boundary representations of n-dimensional geometric objects. Presented at the Proceedings of the fifth ACM symposium on solid modeling and applications, Ann Arbor, Michigan, United States; 1999.
[21] Gomes AJP. Implicit curves and surfaces: mathematics, data structures and algorithms. Dordrecht: Springer; 2009.
[22] Brisson E. Representing geometric structures in d-dimensions: topology and order. Presented at the Proceedings of the fifth annual symposium on computational geometry. Saarbruchen, West Germany; 1989.
[23] Cazier D, Kraemer P. X-maps: an efficient model for non-manifold modeling. Presented at the Proceedings of the 2010 shape modeling international conference. 2010.
[24] Alayrangues S, Peltier S, Damiand G, Lienhardt P. Border operator for generalized maps. 2009.
[25] Lienhardt P. Topological models for boundary representation: a comparison with n-dimensional generalized maps. Computer-Aided Design 1991;23: 59–82.
[26] Gurung T, Laney D, Lindstrom P, Rossignac J. SQuad: compact representation for triangle meshes. Presented at the Eurographics. 2011.
[27] Gurung T, Rossignac J. SOT: compact representation for triangle and tetrahedral meshes. Georgia Institute of Technology GT-IC-10-01; 2010.
[28] Gurung T, Rossignac J. SOT: compact representation for tetrahedral meshes. In: SIAM/ACM geometric and physical modeling. 2009. p. 79–88.
[29] Floriani LD, Hui A. Data structures for simplicial complexes: an analysis and a comparison. Presented at the Proceedings of the third eurographics symposium on geometry processing. Vienna, Austria; 2005.
[30] Weiler K. Edge-based data structures for solid modeling in curved-surface environments. IEEE Computer Graphics and Applications 1985;5:21–40.
[31] Horvath I. et al. About the PRODES system. Budapest: Budapest University of Technology; 1995.
[32] Hocking JG, Young GS. Topology. New York: Dover Publications; 1988.
[33] Choi Y. Vertex-based boundary representation of non-manifold geometric models. Ph.D., Carnegie Mellon University; 1989.
[34] Wilson PR. Data transfer and solid modeling. In: Geometric modeling for CAD applications. 1988. p. 217–54.
[35] Jewellery CAD/CAM Ltd. Available: http://www.jewelcadpro.com.
[36] Liu YJ, Chen Z, Tang K. Construction of iso-contours, bisectors and voronoi diagrams on triangulated surfaces. IEEE Transactions on Pattern Analysis and Machine Intelligence 2011;33:1502–17.
[37] Liu YJ, Ma CX, Zhang DL. EasyToy: plush toy design using editable sketching curves. Computer Graphics and Applications, Vol. 31. IEEE; 2011. p. 49–57.
[38] Nealen A, Igarashi T, Sorkine O, Alexa M. FiberMesh: designing freeform surfaces with 3D curves. Presented at the ACM SIGGRAPH 2007 papers. San Diego, California; 2007.
[39] Liu YJ, Lai KL, Dai G, Yuen MMF. A semantic feature model in concurrent engineering. IEEE Transactions on Automation Science and Engineering 2010; 7:659–65.
[40] Lee SH. Offsetting operations on non-manifold topological models. CAD Computer Aided Design 2009;41:830–46.