Journal of Zhejiang University SCIENCE A
ISSN 1009-3095 (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

A new representation of orientable 2-manifold polygonal surfaces for geometric modelling

LIU Yong-jin[1], TANG Kai[2], JOENJA Ajay[3]

([1]*Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China*)

([2]*Department of Mechanical Engineering, Hong Kong University of Science and Technology, Hong Kong, China*)

([3]*Department of Industrial Engineering and Logistic Management, Hong Kong University of Science and Technology, Hong Kong, China*)

E-mail: liuyongjin@tsinghua.edu.cn; mektang@ust.hk; joneja@ust.hk

Received Mar. 20, 2006; revision accepted May 10, 2006

Abstract: Many graphics and computer-aided design applications require that the polygonal meshes used in geometric computing have the properties of not only 2-manifold but also are orientable. In this paper, by collecting previous work scattered in the topology and geometry literature, we rigorously present a theoretical basis for orientable polygonal surface representation from a modern point of view. Based on the presented basis, we propose a new combinatorial data structure that can guarantee the property of orientable 2-manifolds and is primal/dual efficient. Comparisons with other widely used data structures are also presented in terms of time and space efficiency.

Key words: Shape representation, Combinatorial data structure, Computational topology
doi:10.1631/jzus.2006.A1578 **Document code:** A **CLC number:** TP391

INTRODUCTION

Polygonal surface has become ubiquitous due to its efficient representation of highly detailed geometric objects with arbitrary topological type. In real-world simulation and analysis, a 3D physical object is modelled as a closed subset in $\mathbb{R}^3$ bounded by a connected, compact and orientable 2D manifold. For algorithmically describing orientable 2-manifold polygonal surfaces, the practical data structure should not only easily check manifold property and topological consistency, but also provide time and space efficiency for necessary topological operations.

Smooth 2-manifold surfaces are well studied in differential geometry literature (do Carmo, 1976). However, topological investigations on polygonal models are more scattered in the topology and geometry literature (Cooke and Finney, 1967; Fomenko and Kunii, 1997; Giblin, 1981; Gross and Tucker, 1987; Sieradski, 1992) and harder to locate. The work is aimed at collecting these scattered polygonal modelling work and presenting a rigorous theoretical basis for orientable polygonal surface representation from a modern point of view.

Many data structures have been proposed to algorithmically manipulate polygonal models. However, surprisingly, the only data structure we know so far that can guarantee 2-manifold property is the doubly linked face list (Akleman and Chen, 1999). In this paper, based on the proposed theoretical basis, we also present a new manifold-guaranteed data structure for comparison with most widely used data structures in terms of time and space efficiency.

PRELIMINARIES

An n-dimensional manifold (n-manifold for short) is a second countable, Hausdorff space in which each point has an open neighborhood homeomorphic to $\mathbb{R}^n$. Manifolds are particularly nice topological spaces. If the neighborhood of each point in such a topological space T is homeomorphic either to $\mathbb{R}^n$ or to the n-dimensional half-space H^n, $H^n=\{(x_1, x_2, \ldots, x_n)\in\mathbb{R}^n|x_1\geq 0\}$, then T is an n-manifold with boundary. The boundary of T, denoted as ∂T, is the set of points with a neighborhood homeomorphic to H^n. The complement $T-\partial T$ is called the interior of T, denoted by $Int(T)$.

Definition 1 Compact and connected 2-manifolds are called surfaces.

Let Φ be a closed, non-self-intersecting path in a surface S. Imagine one bug starting out at any point on Φ with a choice of orientation and carrying this orientation with it around Φ once. When it comes back to the starting point, if the chosen orientation is reversed, Φ is called an orientation-reversing path. A closed path that does not have this property is called an orientation-preserving path. A surface is orientable if every closed path in it is orientation-preserving; otherwise, the surface is nonorientable.

Given a topological space T, a k-cell c, $k\geq 0$, is a subspace of T whose interior is homeomorphic to $\mathbb{R}^k$ and whose boundary is $\partial c=Cls(c)-Int(c)$, where $Cls(c)=\cap\{B\subset\mathbb{R}^k|c\subseteq B$ and B is closed$\}$ is the closure of c. In particular, a 0-cell is called a vertex, a 1-cell an edge, a 2-cell a face (or a polygon) and a 3-cell a polyhedron. Denote the dimension of a k-cell c by $dim(c)=k$.

Definition 2 A cell d-complex (M, C) of a d-manifold M is a finite collection $C=\{c_i\}_{i\in I_C}$ of cells whose union is M such that (Cooke and Finney, 1967; Sieradski, 1992)

(1) $\forall i\in I_C$, ∂c_i is either empty or the union of cells from C;

(2) $\forall i, j\in I_C$, $i\neq j$, $Int(c_i)\cap Int(c_j)=\varnothing$;

(3) $\forall i, j\in I_C$, $i\neq j$, if $c_i\cap c_j\neq\varnothing$, then $c_i\cap c_j$ is the union of cells from C;

C is called a subdivision of M.

Given two cells $c_i, c_j\in C$, if $c_i\subseteq\partial c_j$, c_i is called a subcell of c_j. Two cells c_i, c_j are incident if either $c_i\subseteq\partial c_j$ or $c_j\subseteq\partial c_i$. If $c_i\subseteq\partial c_j$ and $dim(c_j)-dim(c_i)=1$, write $c_i\prec c_j$. Two cells c_α, c_β are adjacent if either

(1) $dim(c_\alpha)=dim(c_\beta)=0$ and, $\exists c_\gamma$, $dim(c_\gamma)=1$, such that $c_\alpha\prec c_\gamma$ and $c_\beta\prec c_\gamma$, or

(2) $c_\alpha\cap c_\beta\neq\varnothing$.

Note that (1) adjacent relation includes incident relation; (2) incident relation is a strict partial ordering of C while adjacent relation is not.

If all k-cells in (M, C) are restricted to be the convex hull of a collection of $k+1$ affinely independent points, the resulting special cell complex is called simplicial complex (Giblin, 1981).

Given a d-manifold M, the dual of a complex (M, C) is another complex (M, C^*) for which there exists a one-to-one mapping Ψ from C to C^* such that

(1) the image of a k-cell in C under Ψ is an $(d-k)$-cell in C^*, and

(2) Ψ preserves the adjacent relationship, i.e., cells $\Psi(c_\alpha)$ and $\Psi(c_\beta)$ are adjacent in C^* if and only if c_α and c_β are adjacent in C.

ORIENTABLE POLYGONAL SURFACE REPRESENTATION

Analogous to the general 2-manifold definition in Section 2, we give below a definition on polygonal models and rigorously affirm its manifoldness properties.

Definition 3 An extended cell complex model (S, C) is a cell 2-complex which satisfies

(1) for every 2-cell c_f in C, c_f is locally homeomorphic to planar convexity with internal angles strictly less than π;

(2) for every 0-cell c_v in C, the union of all the 2-cells in C that are incident to c_v is homeomorphic to a two-dimensional disk.

In the above definition, we restrict the internal angles of any convex 2-cell to be strictly less than π, such that the each vertex in C has degree of at least 3. The original cell complex presented in Definition 2 does not guarantee a manifold structure, e.g., consider two tetrahedra joined at a common vertex v (Fig.1a): the object is obviously non-manifold with a singular point v, while its structure satisfies Definition 2. To exclude this case, we set Condition (2) in Definition 3: this condition enforces the manifold property—each point of a surface has an open neighborhood homeomorphic to $\mathbb{R}^2$—on the cell complex model (Fig.1b).

Lemma 1 Given a subdivision C of a closed surface S depicted by an extended cell complex model (S, C), any two distinct faces in C either (1) are disjoint, or (2) have one vertex (0-cell) in common, or (3) have two vertices (0-cell) and one edge (1-cell) joining them in common; and the vertices, edges and faces in C satisfy (1) each face is bounded by at least three edges; (2) each edge has exactly two incident faces; (3) each vertex has degree of at least 3.

Moreover, the graph constituting of vertices and edges in C is a simplicial 1-complex.

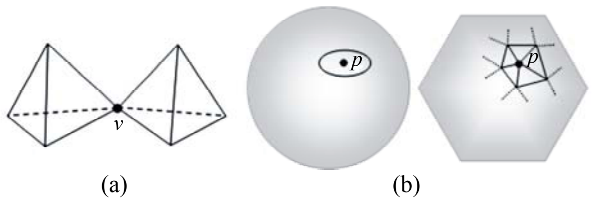

Fig.1 Manifoldness of extended cell complex model
(a) A dangling vertex; (b) Polygonal disk

Proof It suffices to investigate all (but few) configurations that may possibly violate Lemma 1. First, by Definition 2, the configurations shown in Figs.2a and 2b cannot form cell complexities. By Condition (2) in Definition 3, the configuration in Fig.2c cannot occur neither. Then the only possible configuration that could violate the combinatorial conditions in Lemma 1 is shown in Fig.2d in which two 1-cells e_1 and e_2 both belong to the boundaries of 2-cells f_1 and f_2, respectively. Let $p_1\in e_1$ and $p_2\in e_2$, and let l be the line segment joining p_1 and p_2 (Fig.2e). Since both 2-cells f_1 and f_2 are convex, $l\subset f_1$ and $l\subset f_2$; such a case can only occur when the case in Fig.2a is satisfied, a contradiction. That proves the combinatorial conditions stated in Lemma 1.

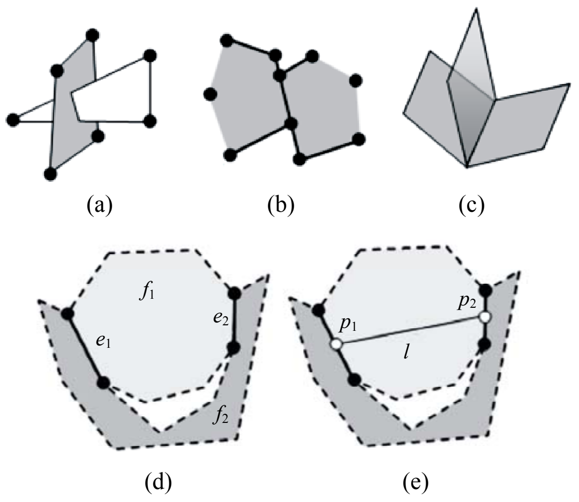

Fig.2 Proof of Lemma 1
(a)~(e) All possible non-manifold configurations

For the simplicial graph statement, the proof directly follows from its definition: a graph is called simplicial if it has no self-loops (i.e., an edge joining a vertex to itself) or multiple edges (i.e., two vertices connected by more than one edge). Self loops and multiple edges are not accepted in our statement since planar convex 2-cells only admit line segments as compatible 1-cells. That completes the proof.

Theorem 1 All the polygonal surfaces represented by extended cell complex models (S, C) are 2-manifold.

Proof A geometric object in $\mathbb{R}^3$ is 2-manifold if and only if each point of the object has an open neighborhood homeomorphic to $\mathbb{R}^2$. Given a surface $S\subset\mathbb{R}^3$ represented by an extended cell complex model (S, C), all the points $p\in S$ can be classified into three cases:

(1) $p\in Int(c_f)$: the point p lies in the interior of some face (2-cell) $c_f\in C$;

(2) $p\in Int(c_e)$: the point p lies in the interior of some edge (1-cell) $c_e\in C$;

(3) $p=c_V$: the point p coincides with some vertex (0-cell) $c_V\in C$.

By Definition 2, the configuration in Fig.2a cannot exist in a cell complex and then every point in Case (1) has a local open neighborhood homeomorphic to $\mathbb{R}^2$. By Lemma 1, each edge in a cell complex has exactly two incident faces and then every point in Case (2) has two local neighborhoods in the corresponding two incident faces each of which is homeomorphic to H^2 and whose union is homeomorphic to $\mathbb{R}^2$. By Condition (2) in Definition 3, every point in Case (3) has a local open neighborhood homeomorphic to $\mathbb{R}^2$. That completes the proof.

Corollary 1 Let a surface S of a constant genus g be represented by an extended cell complex model (S, C). Then the numbers of vertices, edges and faces in C are pairwise linearly proportional. Furthermore, the storage of any combinatorial data structure based on one adjacency relationship has the same storage bound $\Theta(n)$, where n is the number of elements in C.

Proof Let n_V, n_E, n_F denote, respectively, the numbers of vertices, edges and faces in C. The Poincare formula gives

$$n_V-n_E+n_F=2-2g=\chi,$$

where χ is the Euler characteristic. Given g being a constant, so is χ. By Lemma 1, $2n_E \geq 3n_F$ and $2n_E \geq 3n_V$. Then a simple calculation shows that

$$\begin{cases} \frac{3}{2}n_V \leq n_E \leq 3n_V - 3\chi, \\ \frac{1}{2}n_V + \chi \leq n_F \leq 2n_V - 2\chi. \end{cases}$$

That completes the proof.

Definition 4 A local orientation in a 2-cell is defined by specifying a direction of rotation about its barycenter (Fig.3a). Two 2-cells f_i and f_j sharing a common edge e are said to be coherently oriented if the two sides of e have opposite directions in f_i and f_j (Fig.3b).

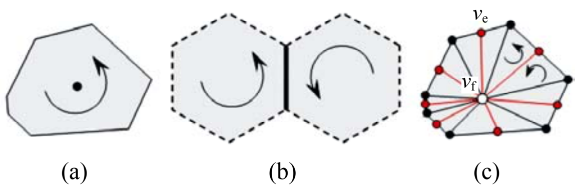

Fig.3 Local coherent orientation and 2-cell refinement
(a) Local orientation in a 2-cell; (b) Coherent orientation of an edge; (c) 2-cell refinement

Definition 5 A refinement of a 2-cell f is constructed by (Fig.3c)

(1) creating one 0-cell v_f inside f,

(2) creating one 0-cell v_e on each 1-cell $e \prec f$, and

(3) connecting v_f by new edges to every 0-cell (both old and new) on ∂f.

Since f is convex, such a refinement is always geometrically valid. If f is oriented, then each newly created 2-cell by refinement has an induced orientation from f (Fig.3c).

Theorem 2 A polygonal surface S represented by an extended cell complex model (S, C) is orientable if and only if all the 2-cells in C can be assigned an orientation in such a way that any two 2-cells sharing a common edge are coherently oriented. Otherwise, S is non-orientable.

Proof By definition, a surface S is orientable iff all closed paths in it are orientation-preserving. Let Φ be an arbitrary closed path in S. Denote by Y the set of all 2-cells in C intersected by Φ and denote by n the number of 2-cells in Y: if Φ overlaps some edge (1-cell) in C, any one of two 2-cells incidents to that edge counts. After sufficient refinements, without loss of generality, suppose each 2-cell in Y contains a single connected piece of Φ.

First, we assert that the 2-cells in Y can be ordered in such a way that f_i shares a common edge with at least one of the 2-cells $\{f_1, \ldots, f_{i-1}\}$, $2 \leq i \leq n$. To prove this assertion, index any 2-cell in Y by f_1 and set $Y=Y-f_1$; then choose any 2-cell in Y as f_2 that shares an edge with f_1 and set $Y=Y-f_2$. Keep running this process by choosing any 2-cell as f_i that shares an edge with some cell in $\{f_1, \ldots, f_{i-1}\}$ and setting $Y=Y-f_i$. If this process stops with $Y \neq \varnothing$, then we have two nonempty sets of 2-cells $Y_1=\{f_1, \ldots, f_k\}$ and $Y_2=\{f_{k+1}, \ldots, f_n\}$ such that no 2-cells in Y_1 have an edge in common with any 2-cell in Y_2. If any 2-cell in Y_2 shares a vertex v with some 2-cell in Y_1, denote the set of all 2-cells in C incident to v by $Star_f(v)=\{f: f \in C, dim(f)=2 \text{ and } v \subset \partial f\}$ (Fig.4). Let $Y=Y \cup Star_f(v)$ and restart the process. If the whole process again stops with $Y \neq \varnothing$, then we have two nonempty sets Y_1 and Y_2 such that no any 2-cell in Y_1 have an edge or a vertex in common with any 2-cell in Y_2. By Lemma 1, Y_1 and Y_2 are two disjoint nonempty sets and thus yield a partition of Φ; a contradiction to the fact that $Y=Y_1 \cup Y_2$ contains a connected, closed path Φ.

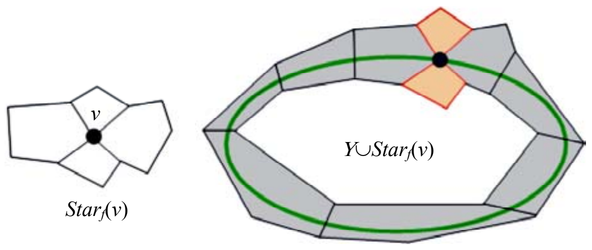

Fig.4 Proof of Theorem 2

Now, for any given closed path Φ in S, let $Y=\{f_1, \ldots, f_n\}$ be the set of finite 2-cells in C intersected by Φ and according with the order that f_i shares a common edge with at least one of the 2-cells in $\{f_1, \ldots, f_{i-1}\}$, $2 \leq i \leq n$. The ordering in Y fuses the local coherent orientation in each $f_i \in Y$ into a global one. In other words, since each 2-cell in Y contains a single connected piece of Φ, induced by orientation in each cell of Y, the pieces of Φ are coherently oriented. Thus, any closed path in S is orientation-preserving and the surface S is orientable. That proves the sufficient condition of Theorem 2.

To prove the necessary condition of Theorem 2, suppose the required orientation in Theorem 2 cannot be assigned, i.e., at least two 2-cells sharing a common edge are not coherently oriented. Let Φ be any closed loop which intersects both 2-cells and is contained completely inside these two 2-cells. It is immediately seen that Φ is an orientation-reversing path so that S is nonorientable.

Fig.5 illustrates two examples of orientability validation of a cylinder and a Mobius band by applying Theorem 2. The classification theorem for surfaces (Giblin, 1981; Sieradski, 1992) shows that all non-orientable surfaces contain at least one copy of the Mobius band and the surfaces that do not contain any copy of the Mobius band are orientable.

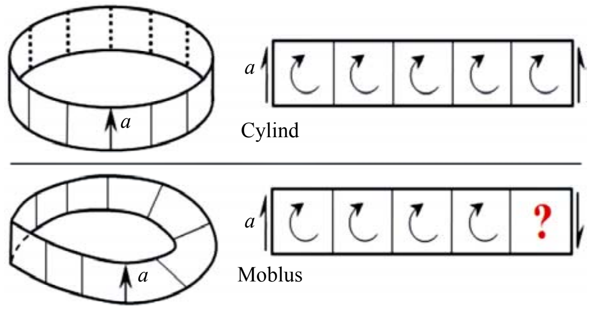

Fig.5 Orientability of a cylinder and non-orientability of a Mobius band

A NEW COMBINATORIAL DATA STRUCTURE

By Corollary 1, any combinatorial representation based on any one of vertex, edge and face lists would have the same storage bound. Note that in the cell complex model, vertices and faces are interchangeable since they are dual to each other and both are connected by edges. In this section, we propose a new combinatorial data structure that uses a quadedge list as the core.

For each edge of the polygonal surface, to offer direct access to local orientation information, akin to the half edge data structure (Mantyla, 1988), the edge is split into two directed counterparts: with each so directed as to indicate a coherent orientation (Definition 4) in one of its two incident faces. By duality, in the proposed edge list, each edge record is thus represented by a quadruple edge structure (Fig.6) whose C definition is

```
struct Edge {
    QuarterEdge   *QE[4];
    Edge   *nxte;
    Edge   *pree;
};
```

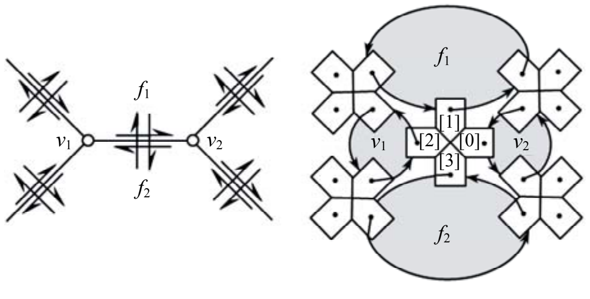

Fig.6 Quarter edge ordering: vertices and faces are represented by loops in the proposed PDER

Refer to Fig.6. Each quarter edge is also linked to its previous and next quarter edge to offer a simple and clear solution to the local traversal problem:

```
struct QuarterEdge {
    void   *data;
    QuarterEdge   *nxtqe;
    QuarterEdge   *preqe;
};
```

The data pointer is reserved to store any surface properties to the quarter edge. As a short summary, the first advantage of the proposed data structure is its conceptual simpleness and ease of implementation; i.e., it can be implemented with a single doubly connected edge list. See Fig.7 for an illustration.

Definition 6 The proposed primal/dual efficient representation (PDER for short) for describing the extended cell complex models is a single doubly connected edge list. Each edge entity consists of four quarter edges and each quarter edge is ordered as illustrated in Figs.6 and 7, i.e., fields *QE*[0], *QE*[2] refer to one (say, primal) structure and fields *QE*[1], *QE*[3] refer to the dual structure.

The PDER structure is similar in spirit to the quad-edge data structure proposed in (Guibas and Stolfi, 1985). Below is a side by side comparison to most existing well-known data structures with which we show that our proposed structure has the following distinct features: (1) direct access to both primal and

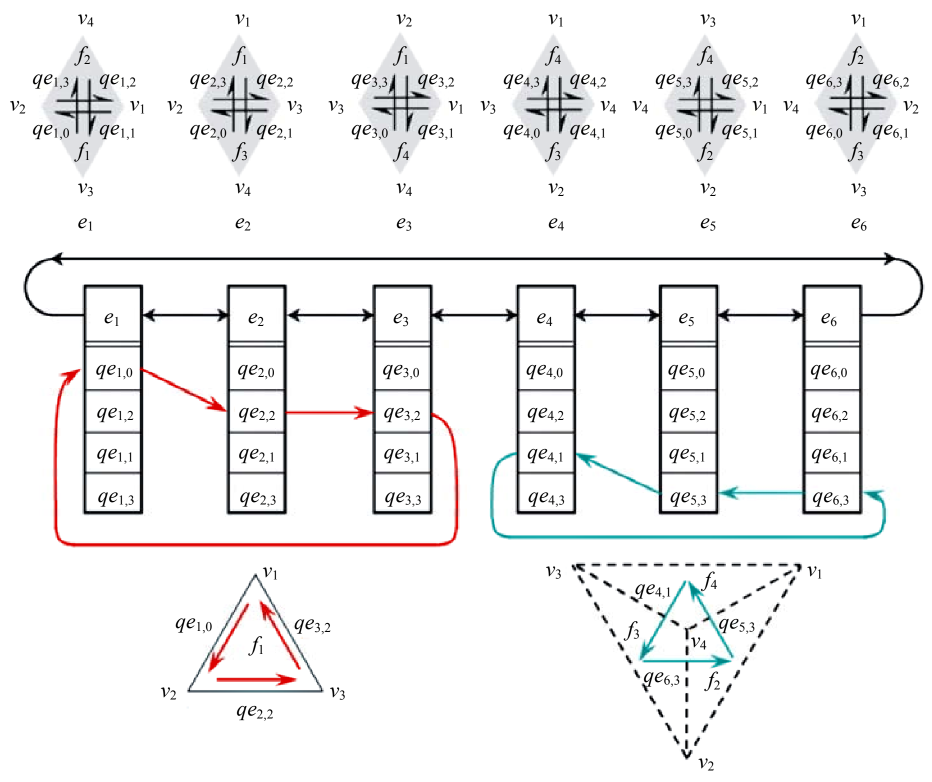

Fig.7 The proposed combinatorial structure implementing a tetrahedron model

dual topological information; (2) sufficiency to represent orientable 2-manifold surfaces; (3) time and space efficiency.

Primal/dual efficiency

To provide a uniform view to both primal and dual structures, PDER does not have additional vertex and face lists; instead, both vertices and faces of the surface are represented in PDER by closed loops in the edge lists. This point is best illustrated by Fig.7. By Corollary 1, the numbers of vertices, edges and faces are linearly proportional. Then a linear scan over all quarter edges, with one more bit for each quarter edge to indicate the check status, is sufficient and necessary to identify all vertices and faces (dual vertices). Thus we have

Lemma 2 Given a PDER for a surface S, it takes $\Theta(n)$ time to identify all vertices and faces (dual vertices) in S, where n is the number of edges in S.

Since PDER does not distinguish vertices and faces, both vertex and face can be geometrically realized with the following geometric element:

```
struct GeomElement {
        float   p[3];
        QuarterEdge   *qe;
};
```

For vertices, $p[3]$ specifies the point coordinate (x, y, z). For faces, $p[3]$ specifies the normal vector $\boldsymbol{n}_x$, $\boldsymbol{n}_y$, $\boldsymbol{n}_z$) (If the plane does not pass the origin, the vector $\boldsymbol{n}$ normalizes the plane equation $(\boldsymbol{p}-\boldsymbol{o})\cdot\boldsymbol{n}=0$ by $\boldsymbol{o}_x\boldsymbol{n}_x+\boldsymbol{o}_y\boldsymbol{n}_y+\boldsymbol{o}_z\boldsymbol{n}_z=1$; otherwise, the magnitude of the vector is scaled to be 1000). Note that point and face are dual to each other through the plane equation $\boldsymbol{n}_x\cdot x+\boldsymbol{n}_y\cdot y+\boldsymbol{n}_z\cdot z-1=0$. The pointer qe in the GeomElement structure points to an arbitrary incident quarter edge of the corresponding loop in the edge list.

To geometrically realize both primal and dual structures of S, two lists of GeomElements are provided:

(1) One is for primal structure with geometric elements one-to-one corresponding to loops formed by the fields $QE[0]$ and $QE[2]$ in the QuarterEdge elements;

(2) The other is for dual structure with geometric elements one-to-one corresponding to loops formed by the fields $QE[1]$ and $QE[3]$ in the QuarterEdge elements.

Let $QE[j]$ in the ith Edge element be denoted by $qe[i][j]$. To traverse between primal and dual structures, let each 2-cell in the surface be oriented in a counterclockwise sense and let $qe[i][0\sim3]$ be ordered as shown in Fig.8. Define *CCW_Rot*() function as

$$CCW\_Rot(qe[i][j])=qe[i][j+1 \bmod 4]$$

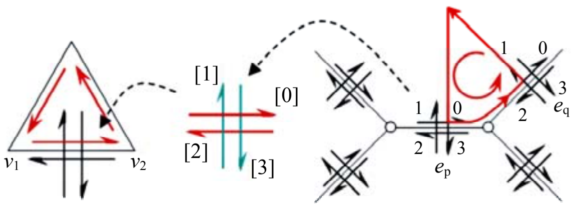

Fig.8 Counterclockwise orientation for *CCW_Rot*() function

The loop shown in Fig.8, i.e.,

$$qe[p][0]\rightarrow qe[q][0]\rightarrow qe[q][1]\rightarrow qe[p][3]\rightarrow qe[p][0]$$

that traverses from primal structure to dual structure, and then back to primal structure, can be carried out by the pseudo-code:

$$CCW\_Rot(CCW\_Rot(qe[p][0]\rightarrow nxtqe)\rightarrow nxtqe)$$

Manifoldness and orientability validation

The proposed PDER can be regarded as a special case of the quad-edge data structure (Guibas and Stolfi, 1985) since both are primal/dual efficient (and then both are edge-based) for representing surfaces. However, the quad-edge data structure is intentionally designed for non-orientable 2-manifold surfaces: in its associated edge algebra, in addition to the *Rot* function that is similar to our *CCW_Rot*() function, another important function *Flip* is defined to traverse between two possible local orientations around an edge. In contrast, our proposed PDER focuses on orientable surface representation with time and space efficiency. Consider the set Ξ of the following well-known data structures for polygonal surfaces: the incident graph (Edelsbrunner, 1987), the index mesh used in VRML format, the face adjacent graph (Ansaldi *et al*., 1985), the winged-edge data structure (WE) (Baumgart, 1972), the doubly connected edge list (DCEL) (Preparata and Shamos, 1985), the half-edge data structure (HE) (Mantyla, 1988), the doubly connected half-edge structure (DCHE) (de Berg *et al*., 1997).

Theorem 3 Given a surface S with n faces, its orientability and topological validity can be verified in $O(n)$ time with $O(n)$ storage. Furthermore, if S is orientable and topologically valid, all the data structures in Ξ representing S can be converted into PDER format in $O(n)$ time with $O(n)$ storage.

Proof The proof consists of three parts. First all the data structures in Ξ are transformed into a canonical form in $O(n)$ time with $O(n)$ storage. Then we present an algorithm with $O(n)$ time and $O(n)$ storage to convert the canonical form into a DCHE with manifoldness and orientability validation. The algorithm returns TRUE value if the conversion is successful, otherwise it reports FALSE. Finally the DCHE is converted into PDER format in $O(n)$ time with $O(n)$ storage.

First it is easily seen that given any edge-based data structure, a linear scan with one more bit for each (half-)edge to indicate the check status can convert it into the indexed mesh data structure (If the model has dangling edges, it cannot be represented by an indexed mesh structure; in such a case, the manifoldness validation will be reported to have failed). Note that

(1) given a half edge, we build up its incident face by tracing with the *nxt* pointers and, the vertex indices in the resulting face record is ordered to indicate the orientation information;

(2) by going through all the half edges in each face, all the edges in the edge list are exactly checked twice.

By Corollary 1, this procedure obviously takes $O(n)$ time with $O(n)$ storage.

Second, we present an algorithm below that converts the indexed mesh structure into a DCHE. The algorithm reports TRUE if the conversion is successful, otherwise it reports FALSE.

```
Algorithm: indexed_mesh_to_DCHE
Input: A indexed mesh ($F$, $V$)
Output: A Boolean value: if the value is TRUE, then the conversion is successful and a doubly connected half-edge structure ($E$, $V$) is also output.

1.    $E\leftarrow\varnothing$;
2.    for each face $f$ in $F$ do
2.1.      if $f$ has less than three vertices
```

```
2.1.1.      return FALSE;
2.2.     Build half edges tracing $f$ with induced orientation
         from $f$;
2.3.     Insert the newly created half edges into $E$;
2.4.     Attach the pointer pointing to the address storing $f$ to
         each of its incident vertices;
3.    for each half edge $e$ in $E$ do
3.1.     if $e\rightarrow twin$ is not filled in
3.1.1.      Mark by "0" all the faces attached to the vertex
            $e\rightarrow strv$;
3.1.2.      Mark by "0" all the faces attached to the vertex
            $e\rightarrow nxv\rightarrow strv$;
3.1.3.      Collect all the faces with mark "1" from the faces
            attached to $e\rightarrow strv$;
3.1.4.      if the number of faces with mark "1" is not equal to 2
3.1.4.1.       return FALSE;
3.1.5.      $f_{twin}\leftarrow$one of the two faces with mark "1" that is not
            $e\rightarrow face$;
3.1.6.      Search in the half edges tracing $f_{twin}$ and find $e'$
            whose vertex $e'\rightarrow strv$ is equal to $e\rightarrow nxt\rightarrow strv$;
3.1.7.      Set $e\rightarrow twin=e'$ and $e'\rightarrow twin=e$;
4.    for each vertex $v$ in $V$ do
4.1.     Find an incident half edge $e$ of $v$ in any one of its
         attached faces;
4.2.     $e'=e$;
4.3.     Start from $e'$, traverse the half edges around $v$ via
         $e'\rightarrow twin\rightarrow nxt$;
4.4.     if the above process stops by $e'$ back to $e$
4.4.1.      return TRUE;
4.5.     else
4.5.1.      return FALSE;
```

The above algorithm is self-explanatory and we only highlight the following several points. For topological validity, the combinatorial conditions in Lemma 1 are checked in Steps 2.1, 3.1.4 and 4.3. Step 4 also takes responsibility for manifoldness and orientability validation. For manifoldness validation, Step 4 checks Condition (2) in Definition 3, e.g., if the model shown in Fig.1a is read in, it cannot pass this step due to the dangling vertex in-between. For orientability validation, by Theorem 2, if any two faces (2-cells) cannot be assigned a coherent orientation, for any vertex incident simultaneously to these two faces, the orientability test in Step 4 cannot be passed.

In Algorithm indexed_mesh_to_DCHE, most storage is allocated at Step 2.4 in which each face is assigned to all its incident vertices. Since for a given face f_i the number= v_{f_i} of its incident vertices is equal to the number e_{f_i} of its incident edges, by Lemma 1, the total storage allocated is then

$$\sum_{i=1}^{n} v_{f_i} = \sum_{i=1}^{n} e_{f_i} = 2n_{\mathrm{E}},$$

where n_{E} is the number of edges in the surface S. Thus the storage bound is $\Theta(n)$, where n is the face number in S. For time complexity, the most timing-consuming parts in Algorithm indexed_mesh_to_DCHE is in Steps 3.1.1.~3.3.3. and 4.3. In these steps, for each given vertex all its incident faces need to be traversed. By duality, it equals that for each given face, traverse all its incident vertices. Thus exactly the same as the storage bound calculation above, the time complexity is $O(n)$.

Finally we show that the DCHE can be converted into PDER format in $O(n)$ time with $O(n)$ storage. It is readily seen that the DCHE can be converted into the primal structure in PDER indicated by quarter edge fields $QE[0]$, $QE[2]$ and their associated loops. To identify the dual structure in PDER by filling in quarter edge fields $QE[1]$ and $QE[3]$, we use the primal-dual-primal loop shown in Fig.8 as the compatible condition, i.e., the directions in the fields $QE[1]$ and $QE[3]$ are specified by satisfying the compatible condition:

$$q[p][0]=CCW\_Rot(CCW\_Rot(qe[p][0]\rightarrow nxtqe)\rightarrow nxtqe)$$

This process is easily shown to take $O(n)$ time with $O(n)$ storage. That completes the whole proof.

Corollary 2 Given a polygonal surface (S, C) stored in a PDER format as defined in Definition 6, the surface S is topologically valid and orientable if and only if for every quarter edge $qe[p][q]$,

(1) the loop indicated by $qe[p][q]$ is closed, and

(2) the following compatible condition is satisfied:

$$qe[p][q]=CCW\_Rot(CCW\_Rot(qe[p][q]\rightarrow nxtqe)\rightarrow nxtqe).$$

Space efficiency and scalability

PDER mainly uses pointers to represent the combinatorial structure of a surface. Suppose that an instantiation of PDER is stored in core memory and a 32-bit platform is used. Typically a pointer is represented by an address of physical memory which is again represented by an unsigned integer stored in a single machine word. Since a 32-bit platform has a word size of four bytes, both a pointer and a pointer to a pointer can be assumed to take four bytes.

Let n_E be the number of edges in a polygonal surface S. By Corollary 1, all the combinatorial data structures depicting S based on one adjacent relationship have the same linear storage bound $O(n_E)$. Now we consider the constant c in this linear bound $O(n_E)=c\cdot n_E$ for different representations. In all edge-based representations in Ξ,

(1) DCEL (Preparata and Shamos, 1985) can be obtained from WE (Baumgart, 1972) by omitting two wings *ccw-pre* and *ccw-succ*;

(2) DCHE (de Berg, 1997) can be obtained from HE (Mantyla, 1988) by omitting the edge list.

Then it suffices to compare DCEL and DCHE with our proposed PDER. Each edge entity in DCEL has six pointers to (v_a, v_b, e_a, e_b, f_a, f_b). Then DCEL totally needs $6\times4\times n_E=24n_E$ bytes. In DCHE each edge has two half edges and each half edge has five pointers to (*strv*, *twin*, *face*, *pre*, *nxt*). Then DCHE totally needs $2\times5\times4\times n_E=40n_E$ bytes. In PDER, each edge entity has four pointers to quarter-edges and two additional pointers to *preqe* and *nxtqe* stored in each quarter-edge record. Then *PDER* totally needs $(4\times2+4+2)\times4\times n_E=56n_E$ bytes (The void data pointer in QuarterEdge is not counted in this calculation).

Both DCHE and PDER can be scalable: the internal references by *pre* pointer can be hidden by locally rebuilding on demand with the knowledge that each loop formed by tracing *nxtqe* pointers for each quarter-edge is closed. By hiding the *pre* pointers, DCHE needs $2\times(3+1)\times4\times n_E=32n_E$ bytes and PDER needs $(4\times1+4+2)\times4\times n_E=40n_E$ bytes. We can further develop a full scalar representation of PDER as exploited below.

To support the function *CCW_Rot*(), we need to traverse the local orientation of an edge from *QE*[0] to *QE*[3] started from another quarter edge. We thus propose the following alternative structure for QuarterEdge:

```
struct QuarterEdge2 {
      void    *data;
      Edge    *nxtqe_in;
      unsigned char    nxtqe_index;
};
```

Then, for example, for the surface structure shown in Fig.7, the representation $e_{1,0}\rightarrow nxtqe$ by using QuarterEdge structure is equivalently expressed by $qe_{1,0}\rightarrow nxtqe\_in\rightarrow QE[qe_{1,0}\rightarrow nxtqe\_index]$. Ideally two bits are enough to represent the index set {0, 1, 2, 3}. Since byte (8 bits) is the unit of most storage measurement, we choose the type of unsigned char that takes 1 byte to implement the index set. A similar argument holds for BOOL type: ideally one bit is enough to represent a BOOL variable, while in Microsoft Visual C++ 5.0 and later, BOOL is implemented as a build-in type with a size of 1 byte.

If further the model is static and the edge number is less than 65535, instead of using a doubly connected list, we can use static array with pre-allocated memory to represent the edge list and use unsigned short int type (2 bytes) to refer to the edge index in a QuarterEdge3 structure:

```
struct QuarterEdge3 {
      void    *data;
      unsigned short int    nxtqe_in_idx;
      unsigned char    nxtqe_idx;
};
together with the edge array
(QuarterEdge3 **)Edge=new (QuarterEdge3 *)[4];
(Edge *) Edgelist=new Edge[nE];
```

In the above minimum storage case, it is easily shown that totally $4\times3\times n_E=12n_E$ bytes are needed. In the case of maximum storage, the doubly connected edge list together with the following full expansion of QuarterEdge structure:

```
struct QuarterEdge4 {
      void    *data;
      Edge    *nxtqe_in;
      unsigned char    nxtqe_index;
      Edge    *preqe_in;
      unsigned char    preqe_index;
      GeomElement    *geom;
}
```

is used, where the pointer *geom* specifies the geometric realization of the loop involving this quarter edge that indicates either a vertex or a face. In this case, totally $[(4+2)\times4+(3\times4+2)\times4]\times n_E=80n_E$ bytes are needed. Note that the adaptability of quarter edge structure from QuarterEdge2 to QuarterEdge4 can be implemented via C++ by a class hierarchy derived from a base class QuarterEdge. We thus conclude

Remark 1 The proposed PDER is scalable, i.e., its minimum required memory is $12n_E$ bytes, and additional memory can be utilized up to $80n_E$ bytes, if it is available, to speed up the retrieval of topological relationship.

Time efficiency

To evaluate the time complexity involving PDER, the performances in the following primitive operations are considered:

(1) Adjacency relationship queries—which is used for accessing the local surface topology;

(2) A minimal and complete set of modification operators—which are used for manifold-guaranteed model development.

Weiler (1985) showed that the following nine queries suffice for examining adjacency relationship:

$e((E)(E))$—find edges adjacent to e;

$e(V)$—find two vertices incident to e;

$e(F)$—find two faces adjacent to e;

$f<E>$—find the ordered circular list of edges surrounding f;

$v<E>$—find the ordered circular list of edges surrounding v;

$v<F>$—find the ordered circular list of faces surrounding v;

$f<V>$—find the ordered circular list of vertices surrounding f;

$v<V>$—find the ordered circular list of vertices surrounding v;

$f<F>$—find the ordered circular list of faces surrounding f.

The first $e((E)(E))$ query is trivial since our data structure is edge-based. By duality, we only need consider the four queries $e(V)$, $f<E>$, $f<V>$ and $f<F>$. Before answering these queries, we need to specify which one of two structures is primal. Without loss of generality, let fields $QE[0]$ and $QE[2]$ point to the primal structure. Refer to Fig.6. It is immediately shown that $e(V)$ and $f<E>$ queries are also trivial:

$e(V)$—the two vertices incident to e are just the two loops indicated by $QE[0]$ and $QE[2]$ of edge e;

$f(E)$—the ordered circular edge list of f is just the loop involving the input quarter edge qe.

The rest of the two queries $f<V>$ and $f<F>$ need to jump among loops and are also easily answered:

(1) For query $f<V>$, upon reordering, let f be specified by the loop constituting quarter edges $qe[i]$, $i=1, \ldots, k$, where k is the number of edges surrounding f. Then the loops formed by $CCW\_Rot(qe[i])$, $i=1, \ldots, k$, give the answer;

(2) For $f<F>$, again, upon reordering, let f be specified by the loop of quarter edges $qe[i]$, $i=1, \cdots, k$. Then the answer is obtained, for each vertex indicated quarter edge $CCW\_Rot(qe[i])$ by using $v<F>$ to traverse the faces and abandon those already visited.

For modelling/updating manifold surfaces, Akleman *et al.*(2003) propose a minimal and complete set of operators which includes the following four primitives:

CreateVertex(v)—insert a new vertex v into a face f and split the face by joining v to each of the incident vertices of f;

DeleteVertex(v)—delete an existing vertex v together with all its incident edges and faces, and merge all its incident faces into a new one;

InsertEdge(v_1, v_2, e)—insert a new edge e into a face f by joining the vertices $v_1 \subset \partial f$ and $v_2 \subset \partial f$;

DeleteEdge(e)—delete an existing edge e together with its two incident faces and its adjacent edges by shrinking e into a vertex.

In the context of PDER, the above set corresponds to the set of operators *CreateLoop*(l), *DeleteLoop*(l), *InsertEdge*(l_1, l_2, e), *DeleteEdge*(e), respectively. Similar to adjacent relationship queries, it is not difficult to implement the above four operators in $O(k)$ time, where k is the number of elements updated. Below we implement *InsertEdge* for an example.

To implement *InsertEdge*, first we note that by duality, inserting an edge is equal to splitting a vertex. For simplicity, let e_1, e_2, e_3, e_4 as shown in Fig.9 be specified by l_1, l_2 and let ne be the new edge to be inserted. The following code is in order:

```
void InsertEdge(e1, e2, e3, e4, ne) {
    e1→QE[0]→nxtqe=ne→QE[0];
    ne→QE[0]→nxtqe=e2→QE[0];
    e4→QE[2]→nxtqe=ne→QE[2];
    ne→QE[2]→nxtqe=e3→QE[2];
    e3→QE[3]→nxtqe=ne→QE[1];
    ne→QE[1]→nxtqe=e1→QE[3];
    e2→QE[1]→nxtqe=ne→QE[3];
    ne→QE[3]→nxtqe=e4→QE[1];
}
```

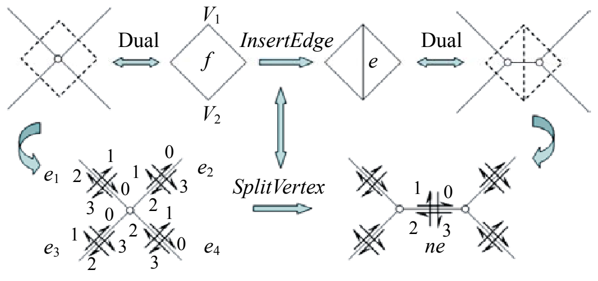

Fig.9 Implementation of operator *InsertEdge*

As a summary, we have:

Lemma 3 By using PDER, the adjacency relationship queries $e((E)(E))$, $e(V)$ and $e(F)$ can be performed in $O(1)$ time and $f<E>$, $v<E>$, $v<F>$, $f<V>$, $v<V>$ and $f<F>$ can be performed in $O(k)$ time, where k is the number of output elements (either edges or loops). Moreover, the manifold-guaranteed modification operators *CreateLoop*, *DeleteLoop*, *InsertEdge* and *DeleteEdge* can be performed in $O(k')$ time, where k' is the number of elements updated.

CONCLUSION

To examine the intrinsic properties of polygonal surfaces, such as manifoldness and orientability, in this paper we present the following results: (1) An extended cell complex model (S, C) is proposed to represent polygonal surfaces; (2) A reliable theoretical basis for orientable polygon surface representation is developed based on the proposed cell complex model (S, C); (3) Built upon the proposed theoretical basis, a new representation called PDER is proposed with a side by side comparison to most existing well-known data structures, showing that the proposed PDER is conceptually simple, easy to implement, and is efficient in time and space efficiency.

ACKNOWLEDGEMENT

The first author sincerely thanks Prof. Shi-Min Hu for his kind support and encouragement in completing this work.

References

Akleman, E., Chen, J., 1999. Guaranteeing the 2-manifold property for meshes with doubly linked face list. *International Journal of Shape Modelling*, **5**(2):159-177.

Akleman, E., Chen, J., Srinivasan, V., 2003. A minimal and complete set of operators for the development of robust manifold mesh modelers. *Graphical Models*, **65**(5):286-304. [doi:10.1016/S1524-0703(03)00047-X]

Ansaldi, S., Floriani, L.D., Falcidieno, B., 1985. Geometric modelling of solid objects by using a face adjacent graph representation. *ACM SIGGRAPH Computer Graphics*, **19**(3):131-139. [doi:10.1145/325165.325218]

Baumgart, B.G., 1972. Winged-edge Polyhedron Representation. Technical report, STAN-CS-320, Stanford University.

Cooke, G.E., Finney, R.L., 1967. Homology of Cell Complexes. Princeton University Press.

de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O., 1997. Computational Geometry: Algorithms and Applications. Springer.

do Carmo, M.P., 1976. Differential Geometry for Curves and Surfaces. Prentice-Hall.

Edelsbrunner, H., 1987. Algorithms in Combinatorial Geometry, EATCS Monographs on Theoretical Computer Science (Vol. 10). Springer-Verlag.

Fomenko, A.T., Kunii, T.L., 1997. Topological Modelling for Visualization. Springer.

Giblin, P.J., 1981. Graphs, Surfaces and Homology (2nd Ed.). Chapman and Hall.

Gross, J.L., Tucker, T.W., 1987. Topological Graph Theory. Wiley-Interscience.

Guibas, L., Stolfi, J., 1985. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. on Graphics*, **4**(2):74-123. [doi:10.1145/282918.282923]

Mantyla, M., 1988. An Introduction to Solid Modelling. Computer Science Press.

Preparata, F.P., Shamos, M.I., 1985. Computational Geometry: An Introduction. Springer-Verlag.

Sieradski, A.J., 1992. An Introduction to Topology and Homotopy. PWS-KENT Pub.

Weiler, K., 1985. Edge-based data structures for solid modelling in curved-surface environments. *IEEE Computer Graphics and Applications*, **5**(1):21-40.