Identifying Faces in a 2D Line Drawing Representing a Manifold Object

Jianzhuang Liu, Yong Tsui Lee, *Member*, *IEEE Computer Society*, and
Wai-Kuen Cham, *Senior Member*, *IEEE*

Abstract—A straightforward way to illustrate a 3D model is to use a line drawing. Faces in a 2D line drawing provide important information for reconstructing its 3D geometry. Manifold objects belong to a class of common solids and most solid systems are based on manifold geometry. In this paper, a new method is proposed for finding faces from single 2D line drawings representing manifolds. The face identification is formulated based on a property of manifolds: each edge of a manifold is shared exactly by two faces. The two main steps in our method are 1) searching for cycles from a line drawing and 2) searching for faces from the cycles. In order to speed up the face identification procedure, a number of properties, most of which relate to planar manifold geometry in line drawings, are presented to identify most of the cycles that are or are not real faces in a drawing, thus reducing the number of unknown cycles in the second searching. Schemes to deal with manifolds with curved faces and manifolds each represented by two or more disjoint graphs are also proposed. The experimental results show that our method can handle manifolds previous methods can handle, as well as those they cannot.

Index Terms—3D models, face identification, geometry, graphs, line drawings, manifolds.

1 Introduction

A 2D line drawing is the simplest and most direct way of illustrating a 3D object. It would be very helpful if such a drawing can be used for generating a 3D model in a CAD system. Unfortunately, current CAD tools cannot directly convert a line drawing into a 3D object, denying designers, especially conceptual designers, a convenient means of input. Therefore, it is highly desirable to develop algorithms that can convert a design sketch into a 3D model. An object consists of faces. If the face configuration of an object is known before reconstructing its 3D geometry, the complexity of the reconstruction will be reduced significantly. Roughly speaking, the conversion problem can be divided into two subproblems: face identification and 3D geometry reconstruction. In this paper, we only deal with the face identification of 2D line drawings of manifold objects. For 3D geometry reconstruction, the reader is referred to references [1], [2], [3], [4], [5], [6], [7], [8].

A 2D line drawing is defined as the projection of a wireframe object where all the edges (including silhouettes) and vertices of the object are visible and the drawing can be represented by a single edge-vertex graph.[1] Manifold objects belong to a class of common solids (see the next section for the definition of a manifold). Most solid modeling systems are based on manifold geometry. Fig. 1 shows two line drawings representing two manifolds, together with their individual faces. Note that in Fig. 1a, the cycle $(1, 2, 3, 4, 1)$ is not a real face, and in Fig. 1b, there is a hole in the object. In Section 3, we point out that previous algorithms that have been reported in the literature cannot handle these manifolds if only 2D line drawings are given (the 3D coordinates of vertices and the number of faces of a drawing are unknown). It is more attractive to develop algorithms that can find faces in drawings of both manifold and nonmanifold objects. Some researchers have tried to reach this goal [9], [10], [11]. However, while successfully dealing with relatively simple nonmanifolds, their algorithms fail to find correct faces of some kinds of manifolds such as those in Fig. 1.

Face identification from line drawings with hidden lines visible has many applications, which include

1. flexible sketching interface for conceptual designers who still tend to prefer pencil and paper to mouse and keyboard in current CAD systems [3], [12],
2. automatic conversion of existing industrial wireframe models to solid models [10], [12], [13],
3. providing rich databases (face topology) to object recognition systems or reverse engineering algorithms for shape reasoning [10], [12], and
4. interactive generation of 3D models from images [7], [8].

A line drawing with hidden lines visible makes it possible to reconstruct its complete 3D model. Given a line drawing, the work of face identification can be done by a designer by manually selecting the edges of the faces. However this is troublesome. Our work in this paper can automate the process and simplify the user interface.

1. Crossing points of two or more edges are not vertices and cannot be used to form faces.

- *J. Liu is with the Department of Information Engineering, Chinese University of Hong Kong, Shatin, New Territories, Hong Kong. E-mail: jzliu@ee.cuhk.edu.hk.*
- *W.-K. Cham is with the Department of Electronic Engineering, Chinese University of Hong Kong, Shatin, New Territories, Hong Kong. E-mail: wkcham@ee.cuhk.edu.hk.*
- *Y.T. Lee is with the School of Mechanical and Production Engineering, Nanyang Technological University, Nanyang Avenue, Singapore. E-mail: mytlee@ntu.edu.sg.*

Manuscript received 30 May 2001; revised 13 Dec. 2001; accepted 25 Mar. 2002.
Recommended for acceptance by D. Forsyth.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 114236.

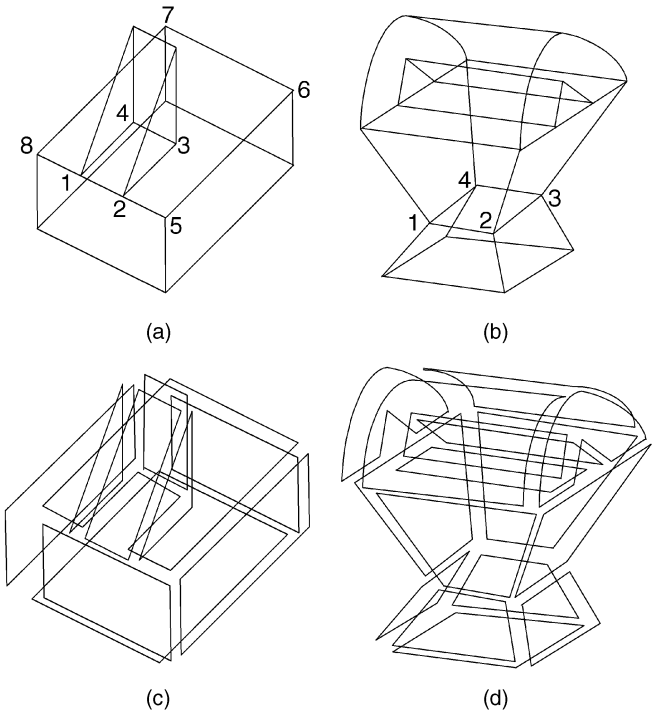

Fig. 1. (a) and (b) Two line drawings. (c) and (d) Their corresponding faces.

This paper proposes a new method for identifying faces in a line drawing representing a manifold. The method is built upon a fundamental property of 2-manifolds (or simply manifold), which states that each edge of a manifold is shared exactly by two faces [14]. First, a set of cycles, which includes all the real faces, is generated from a line drawing. Then a tree search algorithm is used to select subsets of cycles from the set such that each edge of the drawing is passed exactly twice by the cycles in each subset. In order to speed up the search, a number of properties relating to planar manifold geometry are proposed to reduce the number of cycles. An algorithm incorporating these properties is presented to efficiently discard most cycles that cannot be real faces. Schemes to deal with manifolds with curved faces and manifolds each represented by two or more disjoint graphs are also proposed. Our method can handle manifolds previous methods can handle, as well as those they cannot.

2 Terminology

Before proceeding to the next section, we summarize some graph theory and topology terms that will be used in this paper. Some definitions are simplified. More detailed descriptions, except for *chord*, *virtual line*, and *internal face* can be found in [15], [16].

- **Graph**. A graph is defined to be a set of points (vertices) that are interconnected by a set of lines (edges). A graph G may be written as $G = (V, E)$ with V and E being its vertex set and edge set, respectively.
- **Planar graph**. A planar graph G is a graph that can be drawn in the plane without any two of its edges intersecting.
- **Embedding**. An embedding of a graph G is a representation of G on a surface so that none of its edges intersect. A planar graph can be embedded in the plane.
- **Degree**. The degree of a vertex v, written $d(v)$, is the number of edges adjacent to v.
- **Cycle**. A cycle in a graph is formed by a sequence of vertices $v_0, v_1, \ldots, v_n$ where $n \geq 3$, $v_0 = v_n$, the n vertices are distinct, and there is an edge connecting v_i and v_{i+1} for $i = 0, 1, \ldots, n-1$. A cycle is denoted by $(v_0, v_1, \ldots, v_n)$. The length of a cycle is the number of edges it passes.
- **Chord**. A chord of a cycle is an edge connecting two nonadjacent vertices of the cycle.
- **Virtual line**. A virtual line of a cycle is an imaginary straight line connecting two nonadjacent vertices of the cycle. It does not appear as an edge in the drawing.
- ***k*-connected.** A graph is called k-connected if at least k of its vertices and the edges adjacent to them must be removed to make the remaining graph disconnected.
- **Tree**. A tree is a connected graph without cycles.
- **Spanning tree**. A spanning tree of a connected graph G is a tree of G and contains all the vertices of G.
- **Manifold**. A manifold, or more rigorously 2-manifold, is a solid where every point on its surface has a neighborhood topologically equivalent to an open disk in the 2D Euclidean space.
- **Genus**. The genus of a surface can be considered as the number of holes that pass through it completely. The genus of a graph G is the smallest genus of a surface on which G can be embedded.
- **Internal face**. An internal face is a face inside an object and not on its boundary. It is not a real face. The cycles $(1, 2, 3, 4, 1)$ in Figs. 1a and 1b are two internal faces.

3 Related Work

Related work on interpretation of line drawings may be divided into three areas: line labeling, 3D reconstruction from multiple views of wireframe models, and face identification and 3D reconstruction from single line drawings with hidden lines visible. Papers related to line labeling focus on finding a set of consistent labels from a line drawing without hidden lines to test if it is legal, and/or 3D reconstruction based on such a labeled line drawing [17], [18], [19], [20], [21], [22], [23]. Papers in the second group try to reconstruct a 3D CAD model from its multiple (three, in general) orthographic projections [24], [25], [26], [27]. More information can be found from three orthographic views for the reconstruction task than from a single projected view, which is the premise of the third group, to which our work here belongs. Note that we do not deal with 3D reconstruction in this paper; we discuss in the following only the work on face identification from single line drawings with hidden lines visible.

A traditional wireframe model is a collection of all the 3D vertices and edges. Some methods mentioned below use more or less the information of 3D vertex coordinates for face identification from a wireframe model. Only 2D line

drawings (without 3D information) are given in our algorithm.

Markowsky and Wesley [28] proposed a topologically-driven algorithm that can handle wireframes with straight lines. However, their algorithm requires the 3D coordinates of vertices to calculate the normals of planes and is limited to objects with only planar faces.

Hanrahan [29] and Dutton and Brigham [30] used purely topological methods to find the faces of a drawing. A drawing (graph) is embedded in the plane with a planar embedding algorithm [15]. The resulting regions represent the faces of the corresponding object. These algorithms are suitable only for objects of genus 0 whose drawings are 3-connected, because of the requirement of a unique planar embedding for a drawing. However, many drawings are not 3-connected such as the one shown in Fig. 1a.

Another approach also using concepts from graph theory was presented by Ganter and Uicker [31]. From the spanning tree of the graph of a drawing, a set of fundamental cycles is generated. Then, to identify the faces of the drawing, a cycle reduction procedure is designed using these two observations: 1) a cycle, if it is a true face in a given object, has a minimum number of edges in common with any other cycle and 2) the sum of all the edges is a minimum when the cycles make up the true faces of the object. The deficiency in this approach is that it cannot handle objects with holes and internal faces.

Courter and Brewer [32] and Hojnicki and White [13] improved Ganter and Uicker's algorithm by employing better cycle reduction schemes. Their algorithms are able to detect internal faces automatically, but fail when dealing with an object of genus > 0 if the number of faces of the object is unknown.

Bagali and Waggenspack's approach [12] is based on an efficient shortest path algorithm for cycle generation. Their algorithm is fast, conceptually simpler and easy to implement, but limited to 3-connected drawings of genus 0.

Shpitalni and Lipson [9] presented two algorithms for the face identification problem. Their first algorithm, using the planar embedding algorithm to locate faces of a drawing, is similar to those in [29], [30]. Although they put in more effort to find multiple interpretations of a drawing that is not 3-connected, the algorithm is still suitable only for manifolds of genus 0. Their second algorithm is an optimization-based procedure. The criterion they employed to formulate the face identification is based on the observation that a human tends to choose a face configuration in which as many edges as possible take part in as many faces as possible. This algorithm is suitable for a large set of drawings representing manifold and nonmanifold objects. However, it fails when handling the objects with internal faces. For example, the internal faces $(1, 2, 3, 4, 1)$ in Figs. 1a and 1b, are output as two real faces, while the real face $(1, 4, 3, 2, 5, 6, 7, 8, 1)$ in Fig. 1a cannot be found, resulting in a nonmanifold object like that in Fig. 2 when the object in Fig. 1a is seen from the right.

Liu and Lee [11] revisited the problem tackled by Shpitalni and Lipson. They formulated the face identification as a maximum weight clique problem and developed a much faster algorithm to find faces in a line drawing. Their algorithm outputs the same results of face identification, and has the same problem, as Shpitalni and Lipson's.

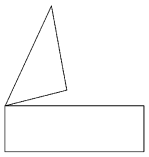

Fig. 2. The side profile of a possible nonmanifold that is made up of the faces found from the drawing in Fig. 1a by Shpitalni and Lipson's second algorithm.

A distinct decomposition method for extracting face topologies from wireframe models was proposed by Agarwal and Waggenspack [10]. The method uses a divide-and-conquer strategy to remove *stars* (tetrahedra, *N*-sided pyramids, or multiply connected stars) from a drawing. The real faces of the drawing are obtained by combining triangles that are created from the stars. This algorithm works for most manifolds and some simple nonmanifolds, and does not take the internal face $(1, 2, 3, 4, 1)$ in Fig. 1b as a real face. However, applying their algorithm to the drawing in Fig. 1a, we found that it is unable to recognize the internal face $(1, 2, 3, 4, 1)$ and also outputs a nonmanifold similar to that in Fig. 2. Note that this internal face is formed by the two touching faces $(1, 2, 3, 4, 1)$ and $(1, 2, 5, 6, 7, 8, 1)$, resulting in a real face $(1, 4, 3, 2, 5, 6, 7, 8, 1)$ on the boundary of the object. Let us consider another drawing shown in Fig. 1b (shown again in Fig. 3a), on which Agarwal and Waggenspack's algorithm fails again. According to the rules of selecting a peak vertex from which to create a tetrahedron in Agarwal and Waggenspack's method, vertex a may be chosen and two pseudoedges bd and cd are added. After the tetrahedron (Fig. 3b) is removed, the remaining object no longer contains edges ab and ac, indicating that none of the subsequently generated stars will contain them either. Thus, edge ab appears only in triangles abc and abd, and edge ac only in triangles abc and acd. Because each of these two edges is contained in only two triangles, the cycle abc is not considered as an internal face but a real face by Agarwal and Waggenspack's algorithm. However, if the object is a manifold, abc is just one end of the hole (not a face).

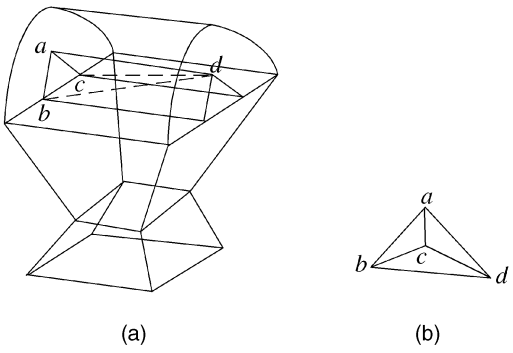

Fig. 3. A manifold to which Agarwal and Waggenspack's algorithm fails to be applied: (a) the drawing where the dashed lines are pseudoedges and (b) a tetrahedron removed from the drawing.

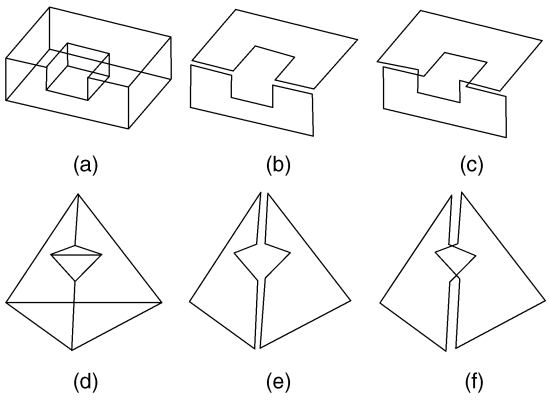

Fig. 4. Two examples of topologically correct but geometrically invalid faces: (a) and (d) drawings, (b) and (e) correct faces, and (c) and (f) geometrically incorrect faces.

The problem of generating faces that are topologically correct but geometrically invalid was noticed by the authors in [9], [10], [13]. Two examples are shown in Fig. 4. Agarwal and Waggenspack [10] and Hojnicki and White [13] used geometric checks to eliminate nonplanar faces. These checks are possible only when the 3D coordinates of the vertices of the drawings are available. Shpitalni and Lipson [9] employed an image regularity, *skewed orthogonality*, to select the most plausible faces. The degree of skewed orthogonality of the two faces in Fig. 4b is distinctly larger than that of the two faces in Fig. 4c. Thus, Shpitalni and Lipson's scheme can find the correct face configuration for the drawing in Fig. 4a. But, it may fail in dealing with the one in Fig. 4d because the pair of faces in Fig. 4e does not exhibit more skewed orthogonality than that in Fig. 4f.

For the face identification for manifolds of genus > 0, Bagali and Waggenspack [12] pointed out that no known topological algorithms exist that can solve the problem in polynomial time with respect to the problem size (number of vertices or edges of a drawing). If G is a connected graph (representing a manifold) with e edges, v vertices, f faces, and genus g, then Euler-Poincare formula [16] states that

$$f = e - v + 2 - 2g.$$

Given a general graph G, finding the genus of G was proven to be NP-complete [33], which means that determining the number of faces of G is also NP-complete. Thus, the two previous algorithms, developed by Shpitalni and Lipson [9] and Agarwal and Waggenspack [10] for objects with genus ≥ 0, have exponential complexities.

In summary, the previous approaches to the face identification from line drawings with hidden lines visible are still not satisfactory although much work has been done. It is more difficult to develop an algorithm that can perform well on both manifold and nonmanifold objects. Even for manifolds only, none of the previous algorithms can handle both the objects in Fig. 1. When the case in Figs. 4e and 4f appears, these algorithms cannot choose the correct pair of faces if no information about the 3D coordinates of the vertices is provided. In addition, it seems that it is impossible to develop an efficient (polynomial) algorithm to handle drawings with genus ≥ 0.

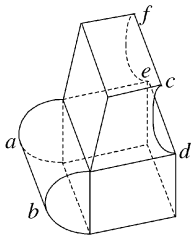

Fig. 5. A manifold where hidden edges are shown in dashed for easier observation.

4 Formulation of Face Identification

This section formulates the face identification as a search problem with two steps: finding a set of cycles from a line drawing and searching for subsets from this set such that every edge of the drawing appears exactly twice in each subset.

Recall that a 2D line drawing is a projection of a wireframe object where all the edges and vertices of the object are drawn. We also require that the object is observed from a general viewpoint such that its curved boundary, if any, is projected into curves, not straight lines. In Sections 4 and 5, we assume that a drawing (manifold) can be represented by one connected vertex-edge graph. A scheme to handle a manifold represented by more than one connected graph is given in Section 6. When a 3D manifold is projected onto the plane, the boundary of any one of its planar faces forms a nonself-intersecting cycle in the drawing, while the boundary of any one of its curved faces forms one or more cycles separated by silhouette lines or curves. Fig. 5 shows the drawing of a manifold with planar and curved faces. Separated by the silhouette line ab, one curved face is projected into two nonself-intersecting cycles, while another curved face (c, d, e, f, c) into one self-intersecting cycle.

In a line drawing of a manifold, there are many cycles, only a small subset of which represents the real faces of the manifold. Many algorithms have been developed to find all the cycles of a graph [34]. Any one of them can be used to generate all the cycles of a drawing.

Suppose that all the cycles of a drawing are given. Now, we consider how to find the real faces from them. The fundamental property of manifolds, which states that each edge of a manifold is shared exactly by two faces, is the basis of our method. With this property, the face identification problem is formulated as follows:

Definition 1. *Given a line drawing of a manifold and the set SC of cycles generated from it, 1) find subsets $X_1, X_2, \ldots, X_m$, where m is the number of subsets and $X_k \subset SC$, $1 \leq k \leq m$, such that each edge of the drawing appears exactly twice in all the cycles in X_k and 2) select solutions $Y_1, Y_2, \ldots, Y_n$, where n is the number of solutions and $Y_l \in \{X_1, X_2, \ldots, X_m\}$, $1 \leq l \leq n$, such that $|Y_l| = \max\{|X_1|, |X_2|, \ldots, |X_m|\}$.*

In this definition, $|S|$ denotes the number of elements in S which is a finite set. It is likely that $m > n$ for a line drawing. Fig. 6 shows such an example, where both subsets

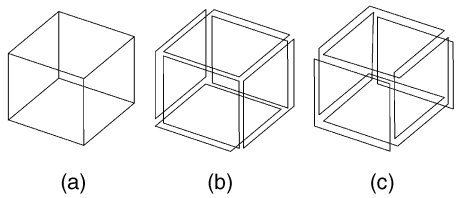

Fig. 6. (a) A cube, (b) the desired solution with six faces of the cube, and (c) a subset of four cycles where each edge of the cube appears twice.

of cycles in Figs. 6b and 6c satisfy the first condition in Definition 1. However, we choose the former subset as the solution because it contains more cycles than does the latter. The second condition in Definition 1 stems from the observation that human beings tend to choose as many faces as possible when interpreting a line drawing. If $n > 1$, more than one solution is found. In this case, further effort is needed to select the most plausible one. This issue will be discussed later.

The backtrack algorithm [35] can be used to find the solutions $Y_1, Y_2, \ldots, Y_n$ while searching a tree that is constructed by the cycles. More details about how to apply it to this problem is discussed in Section 6. The key issue in using the backtrack algorithm is that it is NP-complete and thus consumes a large amount of computational time when there are many cycles. It seems that no efficient algorithms are available to solve the problem in Definition 1. The problem becomes more difficult due to the fact that the number of cycles of a graph is generally exponential in the number of vertices [35]. Let us consider the drawing shown in Fig. 5. There are 861 cycles in it and the backtrack algorithm took 1,237 seconds (more than 20 minutes) on a 677 MHz Pentium III PC to find the unique solution (13 faces). If the object is modified a little as shown in Fig. 7, there are 1,487 cycles in it and the algorithm required 11,969 seconds (more than three hours) on the same PC to find the 14 faces.

In general, the number of cycles in a drawing is much larger than the number of real faces in the drawing. For some manifolds (such as those consisting of only planar faces), it is possible to find conditions that will exclude some cycles from being real faces (see the next section). Therefore, reducing the set of cycles while still keeping all the real faces in it is a practical approach to solving the problem.

5 Finding Cycles Faces

In this section, we exploit certain properties concerning the possibility of a cycle being a real face, based on the connectivities between the edges and the vertices within a drawing. We first consider manifolds with only planar faces and then manifolds with curved faces. An algorithm for finding the cycles of a drawing is given. A scheme is also proposed to deal with a manifold represented by more than one disjoint graph.

Here, we reiterate that vertices, which are end points of edges, are represented explicitly in a graph and the crossing point of two edges is not a vertex and cannot be used to form faces in a line drawing. Our algorithm takes a graph as the input and such crossing points do no exist in the data structure. However, the graph needs to be created from a drawing, which might be drawn directly on a computer or scanned in from a drawing on paper. In the former, the online input information can distinguish whether a point is a vertex based on, for example, the starting and ending positions of a stroke. In a scanned-in drawing, a vertex can be identified based on the continuity of the edges meeting it. In cases where such an identification cannot be made absolutely, user intervention may be required.

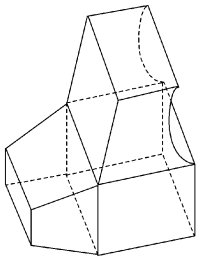

Fig. 7. An object obtained by a modification of the object in Fig. 5.

5.1 Planar Manifold Geometric Properties

It is not difficult for humans to interpret a drawing. In fact, a drawing representing a planar manifold itself carries useful geometric information that can be utilized to eliminate cycles that cannot be real faces. All the lines in such a drawing are straight. If two lines are not collinear in a drawing and a face passes through them, then they determine the plane in which the face lies. In addition, it is assumed that every line that exists in a drawing represents a real edge; that is, it separates two faces lying in two different planes. Every edge is finite and terminates at two end vertices, each formed by the intersection of three or more planes (or edges). Consequently, every vertex has degree ≥ 3.

The following Property 1, which has been mentioned before, is stated again for the frequent reference to it in the proofs of the subsequent properties.

Property 1. *Each edge of a manifold is shared exactly by two different faces.*

Corollary 1. *At a vertex of degree 3, there must be three faces, each containing a different pair of the three edges at the vertex.*

Corollary 1 is a consequence of Property 1. It requires that each of the three incident edges be fully used up, that is, shared by two of the three faces and, therefore, is not available to be part of any other face.

Property 2. *A self-intersecting cycle is not a real face in a drawing representing a planar manifold.*

It is obvious that the projection of the boundary of a planar face cannot form a self-intersecting cycle. Before presenting the next property, we consider two cycles $C_1 = (k, l, b, n, m, j, k)$ and $C_2 = (k, a, o, c, d, e, f, g, h, i, j, m, l, k)$ in Fig. 8. Obviously, they are not real faces. It is easy to see that the chord lm prevents C_1 from being a real face. C_2 has another chord kj and the next Property 3 states that this chord also prevents C_2 from being a real face.

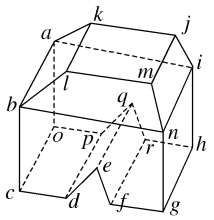

Fig. 8. A manifold used to illustrate some cycles that cannot be real faces.

Property 3. *A cycle cannot be a real face of a planar manifold if it has a chord with at least one of the chord's two vertices met exactly by three lines.*

Proof. Consider part of the drawing of a manifold shown in Fig. 9. From the condition in this property, assume that cycle $C_1 = (a, b, c, \ldots, i, j, k, \ldots, a)$ has a chord bj, the degree of vertex b is 3, and the three lines meeting at vertex b are ba, bc and bj. By Property 1, there must be two faces passing through edge bj. By Corollary 1, one of them must also contain ba and the other bc; let them be $C_2 = (j, b, a, \ldots, j)$ and $C_3 = (j, b, c, \ldots, j)$, respectively. If C_1 is a real face, then a, b, c, and j, being vertices in C_1, must be in one plane. Further, at least one of the two vertices a and c is not collinear with bj. Without loss of generality, let a be such a vertex. Then C_2 and C_1, both containing a, b and j, must lie in the same plane, which contradicts the assumption that two adjacent faces sharing a common line are not coplanar. Hence, the cycle C_1 cannot be a real face of the manifold. □

Let us consider Fig. 8 again. Cycles

$$C_3 = (l, b, c, d, e, f, g, n, m, l)$$

and $C_4 = (l, b, a, i, n, m, l)$ cannot be real faces. The next property points out that the chord bn prevents them from being real faces.

Property 4. *A cycle cannot be a real face of a planar manifold if both of the following conditions are satisfied. 1) The cycle has a chord with at least one of its two vertices being of degree 4. 2) When this chord has only one vertex of degree 4, it is not collinear with any of the other three lines meeting at that vertex; when both of the vertices of the chord are of degree 4, for at least one vertex, the chord is not collinear with any of the other three lines meeting at that vertex.*

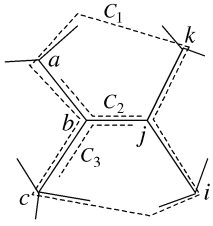

Fig. 9. Part of the drawing of a manifold (solid edges) where the cycle $C_1 = (a, b, c, \ldots, i, j, k, \ldots, a)$ has a chord bj. The dashed lines denote three cycles C_1, $C_2 = (j, b, a, \ldots, j)$, and $C_3 = (j, b, c, \ldots, j)$ that pass through vertex b.

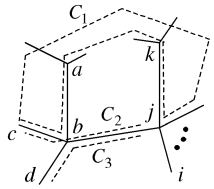

Fig. 10. Part of the drawing of a manifold (solid edges) with three cycles C_{1-3} (dashed lines) where bj is a chord of C_1 and vertex b is met by four lines.

Proof. We suppose that such a cycle is a real face and show that this leads to a contradiction. In Fig. 10, $C_1 = (a, b, c, \ldots, j, k, \ldots, a)$ with a chord bj is such a cycle where $d(b) = 4$ and bj is not collinear with any line in $\{ba, bc, bd\}$. By Property 1, there are two different real faces passing through the chord. Let the two faces be C_2 and C_3. Besides bj, these two faces must each contain a different edge in $\{ba, bc, bd\}$. Without loss of generality, let C_3 contain bd, and, hence, C_2 must contain ba or bc; let it be bc as shown in Fig. 10. Then, since they both contain vertices b, c and j that are not collinear, C_1 and C_2 must lie in the same plane. This contradicts the assumption that the two adjacent faces sharing a common line are not coplanar and thus completes the proof. □

In Fig. 1a, the cycle $(1, 4, 3, 2, 5, 6, 7, 8, 1)$ has a chord connecting vertices 1 and 2, and the degrees $d(1) = d(2) = 4$. This cycle can be a real face because the chord is collinear with both the line connecting vertices 1 and 8 and the line connecting vertices 2 and 5.

Property 5. *A cycle cannot be a real face of a planar manifold if the cycle has a chord that is completely or partially enclosed inside the cycle.*

Proof. Suppose, to the contrary, that such a kind of cycles as illustrated in Fig. 11 are real faces. Clearly, the chord ad in Fig. 11a must lie in the plane of cycle C_1. Since ad is not an edge of this cycle, it must be an edge of two other cycles (faces). Thus, it is contained in three faces, which contradicts Property 1. When the chord ad in Fig. 11b is partially enclosed inside the cycle C_2, the enclosed part of the line ad still lies in the same plane with C_2. Similarly, this part is also shared by three faces, which again contradicts Property 1. Hence, C_1 and C_2 cannot be real faces. □

Property 6. *Let the three vertices of a cycle consisting of three lines be a, b, and c. This cycle must be a real face if any of the*

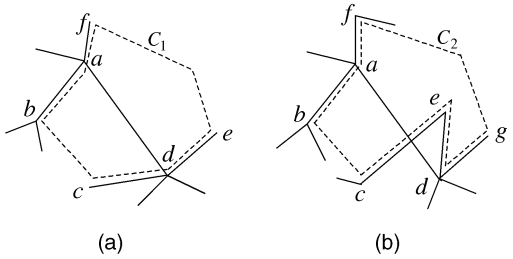

Fig. 11. (a) A cycle $C_1 = (f, a, b, \ldots, c, d, e, \ldots, f)$ with a chord ad enclosed completely by it. (b) Another cycle $C_2 = (f, a, b, \ldots, c, e, d, g, \ldots, f)$ with a chord ad enclosed partially by it. The dashed lines denote the two cycles.

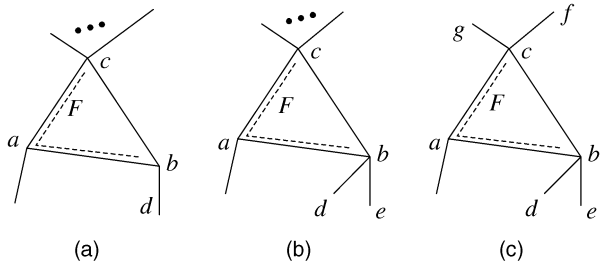

Fig. 12. Three cycles of length 3 with different degrees of vertices: (a) $d(a) = d(b) = 3$, (b) $d(a) = 3, d(b) = 4$, and (c) $d(a) = 3, d(b) = d(c) = 4$. The dashed lines denote the face F passing through lines ab and ac in each case.

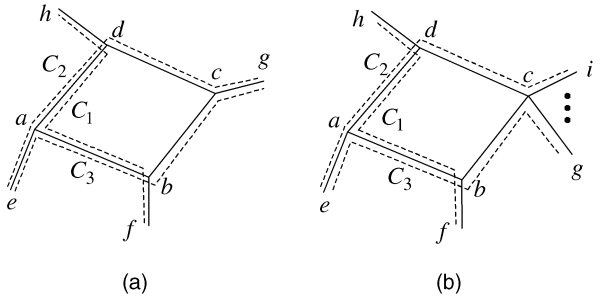

Fig. 13. Two cycles of length 4 consisting of four lines (a) $d(a) = d(b) = d(c) = d(d) = 3$ and (b) $d(a) = d(b) = d(d) = 3$, $d(c) > 3$. The dashed lines denote the three faces C_1, C_2 and C_3 passing through vertex a.

following three conditions is satisfied: 1) the degrees of at least two vertices of the cycle are 3 (Fig. 12a), 2) $d(a) = 3$, $d(b) = 4$, and line bc is not collinear with any line in $\{bd, be\}$ (Fig. 12b), and 3) $d(a) = 3$, $d(b) = d(c) = 4$, and line bc is not collinear with any line in $\{bd, be\}$ or any line in $\{cf, cg\}$ (Fig. 12c).

Proof. Consider Figs. 12a, 12b, and 12c which show three cycles of length 3 all with $d(a) = 3$. Because of Corollary 1, there must be a real face F (the dashed lines in Figs. 12a, 12b, or 12c) passing through lines ab and ac in each drawing. Now, we consider the three cases corresponding to the respective conditions.

1. Let another vertex of degree 3 be b. By Property 3, line bc cannot be a chord of the real face F. Thus, bc must be an edge of F, forming a triangular face.
2. For triangle abc, no two lines can be collinear. Since bc is not collinear with any line in $\{ba, bd, be\}$, bc cannot be a chord of the face F by Property 4. Thus, bc must be an edge of F.
3. For triangle abc, no two lines can be collinear. Since bc is not collinear with any line in $\{ba, bd, be\}$ or any line in $\{ca, cf, cg\}$, bc cannot be a chord of the face F by Property 4. Thus, bc must be an edge of F. □

The intersection of two 3D planes is a straight line, the 2D projection of which is straight too. This leads to the next property. This property is suitable for both manifolds and nonmanifolds, and also used in [9], [11].

Property 7. *The common lines of two adjacent planar faces of an object must be collinear in the drawing of the object.*

Property 8. *If two adjacent planar faces of an object have a common line and a common vertex that is not one of the two endpoints of the line, then the line and the vertex must be collinear in the drawing of the object.*

Proof. In the 3D space, one line and one point, if they are not collinear, determine a plane. Thus, the common 3D line and vertex must be collinear in order to be shared by the two different adjacent planes. The projection of the line and vertex also presents collinearity in the drawing. □

Property 9. *Let the four vertices of a cycle consisting of four lines and without any chord be a, b, c, and d. This cycle is a real face if either of the following conditions is satisfied: 1) $d(a) = d(b) = d(c) = d(d) = 3$, and at least one pair of lines in $\{(ae, cg), (bf, dh)\}$ are not collinear where ae, cg, bf, and dh are four lines that connect to vertices a, c, b, and d, respectively, (see Fig. 13a) and 2) $d(a) = d(b) = d(d) = 3$, $d(c) > 3$, and line ae and vertex c are not collinear where ae is a line that connects to vertex a (see Fig. 13b).*

Proof. In either case, there are exactly three real faces passing through vertex a by Corollary 1. Let the faces be C_1, C_2, and C_3 where C_1 passes through lines ab and ad, C_2 through ae and ad, and C_3 through ae and ab (see Figs. 13a and 13b). From Property 3, we know that if C_1 passes through vertex c, neither line bc nor cd can be a chord of C_1. Therefore they must be edges of C_1. Thus, it follows that $C_1 = (a, b, c, d, a)$ in either condition.

Now, suppose C_1 does not pass through vertex c (i.e. (a, b, c, d, a) is not a real face). Then, C_1 must pass through lines bf and dh. This causes C_2 and C_3 to pass through dc and bc, respectively, due to Corollary 1. In this case, C_2 cannot pass through bc; otherwise, it will further pass through bf, preventing C_2 from being a real face because of the chord ab (see Property 3). Similarly, C_3 cannot pass through dc. Now, let us consider two cases corresponding to the two given conditions.

1. In Fig. 13a, when C_1 does not pass through c, both C_2 and C_3 can only pass through line cg, resulting in the fact that these two faces have two common lines ae and cg. Thus, ae and cg must be collinear by Property 7. In other words, if ae and cg are not collinear, C_1 must pass through c, making (a, b, c, d, a) a real face. Because of the symmetry of Fig. 13a, we can also show that lines bf and dh must be collinear if (a, b, c, d, a) is not a real face. Therefore, if at least one pair of lines in $\{(ae, cg), (bf, dh)\}$ are not collinear, (a, b, c, d, a) must be a real face.
2. In Fig. 13b, when $d(c) > 3$, C_2 and C_3 may pass through any one or two edges except bc and dc meeting at vertex c. Anyway, line ae and vertex c are shared by C_2 and C_3. By Property 8, ae and c must be collinear. Thus, (a, b, c, d, a) must be a real face if ae and c are not collinear. □

We now consider several examples shown in Fig. 14. From Property 9, we know that the six cycles of length 4 in Fig. 14a are all real faces, and from Property 6, triangles (a, b, d, a) and (c, b, d, c) in Fig. 14b, and (i, e, f, i), and (i, e, h, i) in Fig. 14c are real faces too. More real faces of length 4 can also be found from Figs. 14b and 14c. However,

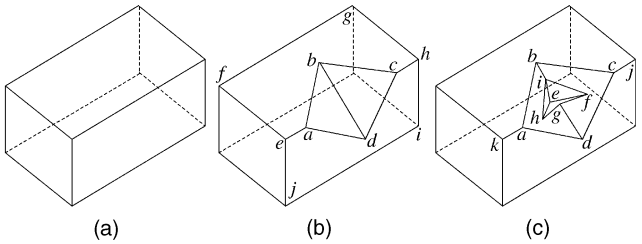

Fig. 14. (a) A rectangular block, (b) the block with a wedge removed, and (c) the object in (b) with another wedge removed.

the cycle (a, b, c, d, a) in Fig. 14b is not a real face since it has a chord bd. The cycle (a, b, c, d, a) in Fig. 14c cannot be considered as a real face because ak is collinear with cj and bi is collinear with gd (Property 9). In addition, two adjacent cycles (a, b, c, h, g, f, e, a) and (a, d, c, h, i, j, e, a) in Fig. 14b) can be two real faces in that the common lines between them, ae and ch, are collinear (Property 7).

Property 10. *Let the four edges connected to a vertex v of degree 4 be va, vb, vc, and vd, respectively. If cycles $C_1 = (a, v, b, \ldots, a)$ and $C_2 = (c, v, b, \ldots, c)$ are known to be two real faces, then all the cycles passing through both edges va and vc cannot be real faces.*

Proof. If such a cycle C_3 that passes through va and vc is a real face, then each edge in $\{va, vb, vc\}$ has been shared by two real faces in $\{C_1, C_2, C_3\}$. Since a real face passing through edge vd must also pass through an edge in $\{va, vb, vc\}$ (say, va), then va is shared by three real faces, which contradicts Property 1. Thus, C_3 cannot be a real face. □

Property 11. *Two cycles cannot be two real faces of a planar manifold if they have the same virtual line and enclose it completely.*

Proof. Suppose, to the contrary, that the two cycles can be two real faces. It is clear that a line must lie on a planar face if it is enclosed completely by the face. Thus, the virtual line lies on both the faces, meaning that it is the intersection of the two faces. However, since it does not appear to be a visible edge of the drawing, the two cycles cannot be real faces. □

Property 11 allows us to find some cycles that are topologically valid but geometrically incorrect. Fig. 15 shows such an example where two cycles $(a, b, c, d, k, i, h, g, a)$ and $(a, f, e, d, k, l, j, g, a)$ have the same virtual line gk, which is completely enclosed by the two cycles. Obviously, the two cycles cannot be real faces of the manifold.

A set of properties relating to the drawings representing planar manifolds has been defined. In the next section, we

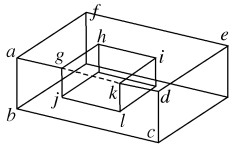

Fig. 15. A drawing where two cycles $(a, b, c, d, k, i, h, g, a)$ and $(a, f, e, d, k, l, j, g, a)$ have the same virtual line gk (not an edge) and are not real faces.

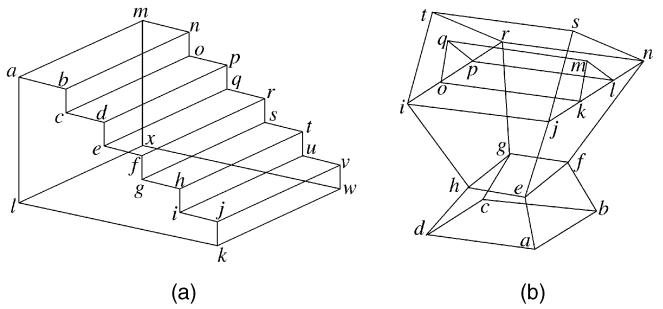

Fig. 16. Two manifolds.

will see how these properties are used to identify many of the cycles that are or are not real faces, thus remove them from and greatly improve the efficiency of, the subsequent searching for the remaining real faces.

5.2 An Algorithm Based on the Properties

There are two schemes that can use the properties developed in the last section to reduce the number of cycles in a drawing. One scheme is to generate first all the cycles using one of the available algorithms [34] and then according to the properties to eliminate as many cycles that cannot be real faces as possible. Another scheme is to combine the properties into a cycle-search algorithm such that most of the cycles unable to be real faces are not generated. Obviously, the second scheme is more efficient. Let us take two examples to see how the properties are used.

Fig. 16a is the drawing of a stair model, in which there are 4,228 cycles and only 14 cycles are real faces. If we first search for all the cycles of length 4, then we obtain 12 cycles. According to Property 9, we know that all these are real faces. Thus, the lines am, bn, co, dp, eq, fr, gs, ht, iu, jv, kw, and lx, can be deleted from the drawing because each is shared by two real faces. In the remaining drawing, there are only two cycles

$$(a, b, c, d, e, f, g, h, i, j, k, l, a)$$

and $(m, n, o, p, q, r, s, t, u, v, w, x, m)$, which must be real faces because of Property 1. For this drawing, we can easily find a very small set of cycles by Properties 1 and 9. We do not even need to search for real faces from within them since they already are.

However, we are not always so lucky with most other drawings. For the object shown in Fig. 16b, we can only determine that one cycle $C_1 = (a, b, c, d, a)$ is a real face by Property 9 at first. Now, suppose that a depth-first search algorithm [35] is employed to find other cycles. Since C_1 is a real face and no pair of its four edges are collinear, from Property 7, we know that another real face passing through edge ab is not allowed to pass through edges ad or bc. Thus, it must pass through edges ae and bf. It is clear that edge ef is not collinear with any of the edges eh, ae, and ej. From these facts and Property 4, we can deduce that among all the cycles that pass through edges ab, ae, and bf, only one cycle $C_2 = (a, b, f, e, a)$ may be a real face. By Property 1, edge ab must be shared by another cycle besides C_1. Therefore, C_2 must be a real face. Similarly, cycles (a, e, h, d, a), (b, c, g, f, b), and (c, d, h, g, c) are real faces too.

As a result, edges ab, bc, cd, ad, ae, bf, cg, and dh can be deleted from the drawing because each of them has been passed exactly by two real faces. In the remaining drawing, can we still determine which cycles are real faces? Consider the cycles passing through edge eh. From Property 10, we know that these cycles can only pass through edges hi and ej based on the real faces found so far. Clearly, there are still many such nonself-intersecting cycles in the remaining drawing (self-intersecting cycles are excluded due to Property 2). But, all these cycles except $C_3 = (e, j, i, h, e)$ enclose chord ij and cannot be real faces according to Property 5. Thus, C_3 must be a real face such that edge eh is shared by two real faces. Similarly, cycle (f, n, r, g, f) is also a real face. Therefore, edges eh and fg can be further deleted. Now in the remaining drawing, we are not able to directly determine which cycles are real faces with the help of the properties. An edge is deleted only after all the cycles passing through it are generated. However, Properties 2 and 3 can still be used to eliminate most of the cycles that cannot be real faces. There are totally 11,052 cycles and 16 real faces in Fig. 16b. Using the algorithm presented in the following, only 37 cycles are generated, seven of which are already known to be real faces. Obviously, the properties are very efficient for excluding cycles that cannot be real faces.

It has to be mentioned that the checking of the conditions stated in the properties while searching is based on the original drawing instead of its reduced one (in which some edges have been deleted). Consider the drawing in Fig. 16b. By the analysis above, we can delete edges ab, bc, cd, ad, ae, bf, cg, and dh from the original drawing at first. In the remaining drawing, we cannot say that cycle (e, f, g, h, e) is a real face according to Property 9, because neither of the conditions in Property 9 is satisfied in the original drawing. That an edge is deleted only means that no cycles in the subsequent search can pass through it.

Note that, using Properties 2 and 3 we may often avoid much fruitless search, before generating cycles that cannot be real faces. Let us consider again the drawing in Fig. 16b. Suppose we are now searching for cycles passing through edge js: $(j, s, n, r, v_1, v_2, \ldots, v_u, j)$, where $v_1, v_2, \ldots, v_u \in S - \{j, s, n, r\}$ and S is the set of vertices of the drawing (or remaining drawing after deleting some edges). Obviously, there are many different cycles obtainable by the permutations of some of the vertices $v_1, v_2, \ldots, v_u$. However, with the fact that edges js and nr intersect, we know that all these cycles cannot be real faces, and thus we can stop the search algorithm to generate these cycles.

The complete algorithm is summarized in 11 steps below. Explanations for some of the steps are given after the algorithm, which is called the DFSP algorithm in what follows because it is actually a depth-first search (DFS) algorithm with the properties incorporated to guide the search.

1. Initialization:
   a. Set a 2D array M_I indicating if any two edges intersect.
   b. Set a 2D array M_{CL} indicating if any two edges are collinear.
   c. Set a 2D array M_{CE} indicating that all edges can coexist in any cycles.
2. Search for cycles of length 3 and check if they are real faces according to Property 6.
3. Search for cycles of length 4 and check if they are real faces according to Property 9.
4. If there are real faces found, update M_{CE} to indicate that some pairs of edges cannot coexist in subsequently-generated cycles according to Property 7.
5. Delete edges each being passed by two real faces. Denote the remaining drawing as G.
6. Stop if no edge exists in G.
7. Pick up from G an edge that is passed once by a real face. If there is no such edge, pick up any one. Denote the chosen edge as l and its two end vertices as v_1 and v_2.
8. Search for cycles that pass through l in G:
   a. According to M_{CE}, check if there is an *unambiguous path* $(v_2, v_1, v^1, v^2, \ldots, v^n)$, where all these vertices are different.
   b. If such a path exists (case 1), put the sequence s_1: $v^n, v^{n-1}, \ldots, v^1, v_1, v_2$ into a 1D array M_{path}; otherwise (Case 2), put the sequence s_2: v_1, v_2 into M_{path}.
   c. Based on M_{path}, M_I, M_{CL}, and M_{CE}, starting from v_2, find a set S_c of cycles passing through s_1 (in Case 1) or s_2 (in Case 2), using the DFS algorithm in [35] incorporated with several searching rules derived from some of the properties.
9. Update M_{CE} according to Properties 7 and 10 if there is only one cycle in S_c and edge l is passed twice by two cycles so far.
10. Delete edge l from G. Denote the reduced drawing as G_1. Check if there are vertices of degree 1 in G_1. If so, delete edges connecting to these vertices from G_1. Repeat this procedure until all the degrees of vertices in the last-reduced drawing G_m are at least 2 or until no edges exist.
11. Set $G = G_m$ and go to Step 6.

We now explain some of the steps in the DFSP algorithm. The 2D array M_{CE} is used to indicate the pairs of edges that cannot coexist in subsequently-generated cycles. It is obtained according to Properties 7 and 10 after real faces have been found. For example, in the drawing shown in Fig. 16b, after the real face (a, b, c, d, a) has been found, edge ab cannot coexist with edges bc, cd, or ad in other real faces (Property 7).

In Step 7, an edge that is passed once by a real face has the priority of being chosen. This is because it is more likely to obtain an unambiguous path $(v_2, v_1, v^1, v^2, \ldots, v^n)$ from such an edge. On this path from v_2 to v^n, no deviation from it is allowed at vertices $v_1, v^1, v^2, \ldots, v^{n-1}$, which is why an *unambiguous path* is termed. We hope that n is as large as possible. For the drawing in Fig. 16b, suppose edge ab is chosen after the real face (a, b, c, d, a) has just been found. An unambiguous path passing through ab is (b, a, e) because edges ab and ad cannot coexist. This path cannot be longer since edge ae can coexist with three (not only one of) edges ef, ej, and eh according to M_{CE} obtained so far. With this unambiguous path (b, a, e), searching for paths from vertex b to vertex e is more efficient than searching for

paths from vertex b to vertex a, although both paths all correspond to cycles passing through edge ab. Starting from vertex b, suppose the algorithm now reaches vertex f. In the former case, the algorithm will not go to vertex n or g because otherwise, the chord ef will render the cycles found not real faces by Property 4. However, in the latter case, the algorithm does not know there is a chord ef when it goes to vertex n or g from f because vertex e has not yet been in M_{path}, which stores the path obtained so far.

The DFS algorithm ([35], pp. 348-353) mentioned in Step 8(c) can be used to generate all the cycles passing through sequence s_1 or s_2. However, since we can use some of the properties to eliminate many cycles unable to be real faces while searching, it is more efficient to incorporate the following search rules into the DFS algorithm.

Rule 1. Before adding a new vertex (corresponding to a new edge) into M_{path}, the DFS algorithm checks if the new edge intersects any edge in the path found so far by examining M_I. If so, try another vertex; otherwise, test the condition in Rule 2.

Rule 2. If the new edge cannot coexist with any edge in the path by examining M_{CE}, try another vertex; otherwise test the conditions in Rule 3.

Rule 3. If adding the new vertex leads to a chord in the path (such as the chord ef in the path (e, a, b, f, n) in Fig. 16b) and the chord can prevent cycles passing through this path from being real faces according to Properties 3 and 4, then try another vertex, otherwise put the new vertex into M_{path}.

Rule 4. When a cycle has been generated, check if it encloses a chord using the *inside* algorithm in [36] (p. 354). If so, discard the cycle according to Property 5.

In Step 9, if only one cycle is found in Step 8(c) and edge *l* is passed exactly twice by the cycles found so far, then this cycle must be a real face (Property 1). In this case, more pairs of edges that cannot coexist in the subsequently-generated cycles may be determined according to Properties 7 and 10. Thus, M_{CE} needs to be updated. In Step 10, deleting one edge may lead to vertices of degree 1 in the reduced drawing. Edges connected to these vertices have no contributions to further search for cycles and should be deleted.

5.3 Dealing with General Manifolds

A general manifold is defined here as a manifold of genus ≥ 0 and possibly with curved faces. If there are curved edges in a line drawing representing a manifold, the manifold is a general one. It is intuitively clear that, the DFSP algorithm cannot be applied to such drawings because most of the properties are only suitable for planar manifolds. For general manifolds, the problem of face identification becomes more complicated. Note that all the previous methods can only deal with line drawing models involving simple geometry.

It is reasonable to consider that a drawing representing a manifold is a planar one if all its edges are straight lines. The approach to dealing with general manifolds in this paper is first to transform a general manifold to a planar manifold, to which the DFSP algorithm is then applied. The transformation is performed simply by replacing all the curved edges with straight lines. Fig. 17 shows three

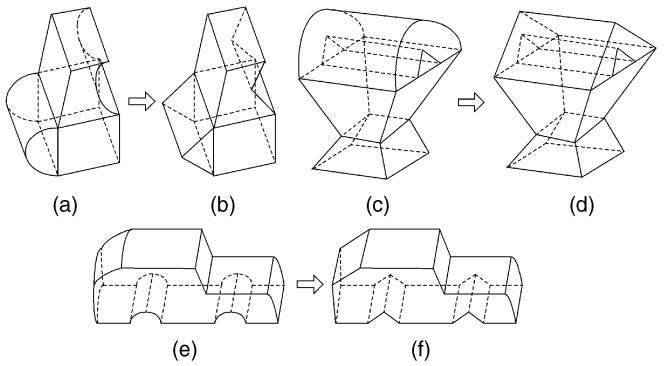

Fig. 17. Replacing curved edges with straight lines where hidden edges are shown in dashed for easier observation.

examples. We can see that each face of a planar manifold has a corresponding face in its corresponding general manifold. In other words, for these drawings, the face identification from a general manifold is equivalent to that from its transformed planar manifold. In what follows, we say that such a general manifold and its transformed one are equivalent. Can an equivalent planar manifold always be created?

Consider the drawing of a cylinder in Fig. 18a. If the four curved edges abc, adc, efg, and ehg are replaced by four straight lines, the drawing will not represent a manifold. However, if one line bf is added (Fig. 18b), then the drawing in Fig. 18c, obtained by straightening all the curved edges in Fig. 18b, represents a planar manifold. More importantly, the two drawings are equivalent. After finding the five faces (Fig. 18d) of the planar manifold in Fig. 18c, we obtain the faces (Fig. 18e) of the general manifold in Fig. 18b. Note that the added (artificial) line, not a real or silhouette edge, is still shared by two faces separated by it.

Because of the transformation, one curved face may be divided into two or more. A face combining process can merge these separated faces into one and discard added lines. Let a common edge be an edge shared by two faces. Suppose such an edge meets two other edges each belonging to one of the two adjacent faces. If these two edges meet smoothly, then it is reasonable to combine the two adjacent faces into one, discarding the common edge. For the faces in Fig. 18e, since edges ab and bc (or edges ef

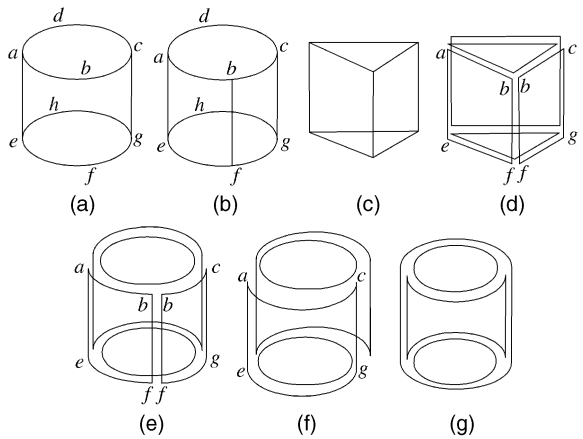

Fig. 18. Finding faces in a cylinder.

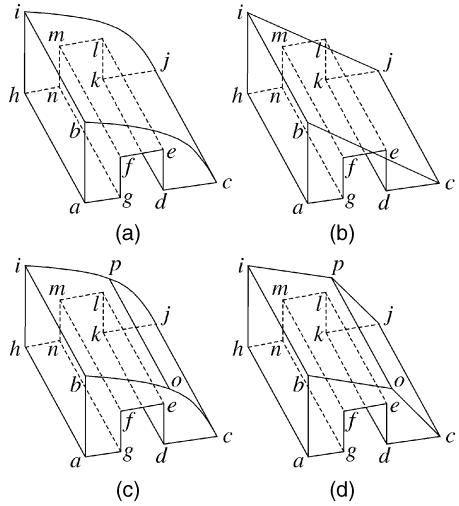

Fig. 19. Finding faces in another general manifold.

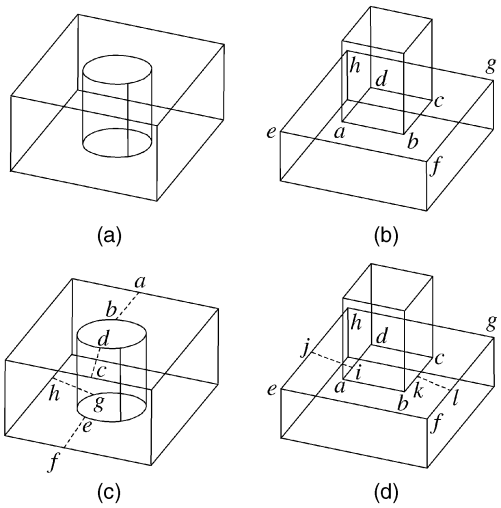

Fig. 20. Two manifolds with faces enclosed by two cycles.

and fg) meet smoothly in Fig. 18b, the common edge bf is eliminated and the two faces (a, b, f, e, a) and (c, b, f, g, c) are combined into one face (a, c, g, e, a). Now, the four faces shown in Fig. 18f can be obtained, which are the faces of the cylinder in Fig. 18a.

Adding the line bf in the drawing (Fig. 18b) not only allows the application of the DFSP algorithm to this object, but also eliminates the ambiguity of interpretation of the drawing in Fig. 18a) caused by multiple face configurations. We have two valid solutions (Figs. 18f and 18g) to the drawing in Fig. 18a, but have only one (Fig. 18f) when the line bf is added. If we do need the faces in Fig. 18g instead of those in Fig. 18f, then the added line should be bh connecting the two curved edges abc and ehg.

Let us consider another drawing shown in Fig. 19a. If the two curved edges bc and ij are replaced by two straight lines (Fig. 19b), cycles (a, b, c, d, e, f, g, a) and (h, i, j, k, l, m, n, h) will be self-intersecting and discarded by the DFSP algorithm. However, if a line op between the two curved edges is added (Fig. 19c), the drawing in Fig. 19c is equivalent to that in Fig. 19d, which is obtained by straightening the four curved edges of the former. That is to say, if the 10 faces of the planar manifold in Fig. 19d can be found, the 10 faces of the general manifold in Fig. 19c can be found too. Furthermore, since the curved edges bo and oc (or ip and pj) meet smoothly in Fig. 19c, two faces (b, o, p, i, b) and (o, c, j, p, o) should be combined together to form one face (b, c, j, i, b). Thus, nine faces of the object in Fig. 19a are obtained.

The DFSP algorithm can always be used for a line drawing representing a curved manifold if sufficient artificial edges are added to the curved faces. The validity is based on this observation: 1) every curved face can be approximated with one or more planar patches and, thus, a curved manifold can always be approximated by a planar manifold and 2) if the transformed object is a planar manifold, the properties and the algorithm developed for planar manifolds can be applied to it. Therefore, the algorithm combined with the scheme (adding artificial lines, straightening curves, then combining faces and discarding the artificial lines) can be used to find the faces of a curved manifold from an appropriate planar approximating manifold. Note that only the first step of adding artificial lines in the scheme is done manually by the user when making the line drawing.

At this stage, we do not define how many artificial lines must be added to a curved line drawing; the user has the choice. For example, for the line drawing shown in Fig. 18b, if one more artificial line connecting the curved edges adc and ehg is added, the four faces as shown in Fig. 18f will still be obtained.

Through numerous observations and experiments, we have found that a transformed planar manifold can be used to find the faces of its corresponding curved manifold if two conditions are satisfied: 1) no two lines overlap after straightening curved edges (e.g., see Fig. 18) and 2) cycles that belong to real faces are not self-intersecting (e.g., see Fig. 19). A stricter condition that implies the above two is that a straight line obtained by straightening a curve should be very close to the curve. With this condition, a transformed planar manifold will be a good approximation to its corresponding curved manifold. Although this condition may require adding more artificial edges to a line drawing, it will guarantee that from the transformed planar manifold, we can find the faces of its original curved one. Giving more edges to a drawing imposes a little more work on a designer. But, doing this often provides better visual perception of a drawing and reduces ambiguity of interpretation.

5.4 Dealing with Manifolds Represented by More Than One Graph

In Section 1, we require that a line drawing be represented by a *single* edge-vertex graph, which is also the assumption in previous related papers. However, there are manifolds each of which is represented by two or more disjoint graphs. Fig. 20a shows a manifold where a hole passes through a cube and Fig. 20b gives another example where one face of a smaller cube is on a face of another cube. In these cases, a face may be enclosed by more than one cycle. To identify the cosurface cycles, more information is required. In our work, we use dashed artificial lines to connect these cycles. For the two drawings in Fig. 20, ab, cd, ef, gh, ij, and kl are such lines. These lines are dashed to indicate that they are not edges and cannot be used to construct cycles.

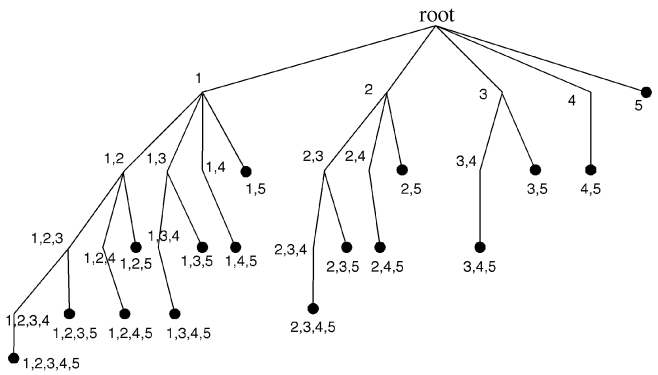

Fig. 21. A state-space tree.

At first, each graph is treated separately by the whole algorithm (see the next section). After all the cycles considered as faces in each graph have been found, the dashed lines are used to identify the real faces each enclosed by more than one cycle. Now, let us take the drawing in Fig. 20d, which contains two disjoint graphs, as an example to see how the faces are found. First, six cycles are found as faces for each graph by the DFSP algorithm. The two cycles $C_1 = (h, e, f, g, h)$ and $C_2 = (d, a, b, c, d)$ are among the 12 cycles. They are considered to be located on the same surface with the help of the two dashed lines ij and kl, forming the real face enclosed by C_1 and C_2. Thus, 11 faces are obtained for the object in Fig. 20b.

For the object in Fig. 20a, at first, all the faces of the block and the cylinder are found from the two disjoint graphs. Then four of them are used to form two real faces each enclosed by two cycles, resulting in an object with a hole.

6 Finding Real Faces from Cycles

With a set of cycles generated by the DFSP algorithm, we can construct a state-space tree and use a backtrack algorithm to find the solutions while searching in the tree. Let us see a simple example. Suppose there are only five cycles. A state-space tree constructed is illustrated in Fig. 21. A node of the tree defines a problem state and all the nodes define the state space of the problem. For our problem, each node except the root denotes a combination of some cycles. All possible combinations of different numbers of cycles determine the size of the tree. The total number of nodes is equal to $C_p^0 + C_p^1 + \ldots + C_p^p$ $(= 2^p)$ if there are p cycles.

Although the tree is huge when p is a large number, it is not necessary to expand all the nodes of the tree. Let SC be the set of cycles generated by the DFSP algorithm from a drawing and $Z \subset SC$ be a subset denoted by some node. If the cycles in Z cause any edge of the drawing to be used more than twice, this node will be deleted; otherwise, it will be expanded by adding a new cycle i into Z. The new subset $Z \cup \{i\}$, denoted by a new node of the tree, will be tested again. When a solution is found or the bottom (the solid circles in Fig. 21) of the tree is reached, backtrack one level and search again. Besides, Property 7 can be used to eliminate more nodes of the tree. Let us take the drawing in Fig. 19d as an example. Suppose that cycle (f, g, n, m, f) has already been in Z but cycle $(f, e, d, c, j, p, i, h, n, g, f)$ has not. Then, the latter cannot be added to Z because both the cycles pass through edges ng and gf that are not collinear.

As mentioned before, there are multiple solutions found by the backtrack algorithm for some drawings such as those in Figs. 4a and 4d. Property 11 provides a scheme to eliminate some solutions with topologically-valid-but-geometrically-incorrect cycles. Besides, employing skewed-orthogonality detection to select the most plausible faces [9] is another scheme.

Now, we summarize the proposed method for face identification from a line drawing representing a manifold in the following steps. Suppose the line drawing is represented by N disjoint graphs $G_1, G_2, \ldots, G_N$.

1. $i \leftarrow 1$.
2. Generate a set SC_i of cycles from G_i using the DFSP algorithm.
3. Find subsets of cycles from SC_i such that each edge of G_i appears exactly twice in each subset using the backtrack algorithm.
4. Employ Property 11 and the skewed-orthogonality detection to select the most plausible faces if there exist multiple subsets.
5. Combine faces if there exist artificial lines that separate curved faces.
6. $i \leftarrow i + 1$. Go to Step 7 if $i = N$; otherwise, go to Step 2.
7. Find the real faces each enclosed by more than one cycle if there exist artificial dashed lines.

7 Experimental Results

In this section, we present a number of examples to demonstrate that our method can successfully identify the faces of drawings representing manifold objects. Fig. 22 shows eleven drawings in the experiments, each together with the faces found by our method. Note that the shaded faces are ones enclosed by two cycles. Obviously, the results accord with human interpretation of the drawings. Table 1 summarizes the main parameters. The numbers in column 2 list the total cycles in each drawing and the numbers in column 3 are the cycles found by the DFSP algorithm. Comparing these numbers between the two columns, we can see that the DFSP algorithm eliminates most of the cycles that cannot be real faces, making the backtrack algorithm run fast to find the solutions.

For the two drawings Figs. 22c and 22f, the backtrack algorithm finds multiple solutions. (Fig. 4 has shown the multiple solutions where only different cycles in the solutions are given for each object.) However, Property 11 can be used to select the most plausible one.

For each of the drawings (a), (d), (e), (g), (h), (i), (j), and (k), the backtrack algorithm does not even need to perform searching in the state-space tree. This is because all the cycles generated by the DFSP algorithm from each of these drawings already share each edge exactly twice. On a 677 MHz Pentium III PC, the whole algorithm takes less than 0.1 second to identify the faces for each of the drawings in Fig. 22.

8 Conclusions

A new method has been proposed for finding the faces from single 2D line drawings representing manifold objects, a class of common solids. The faces identified from a drawing

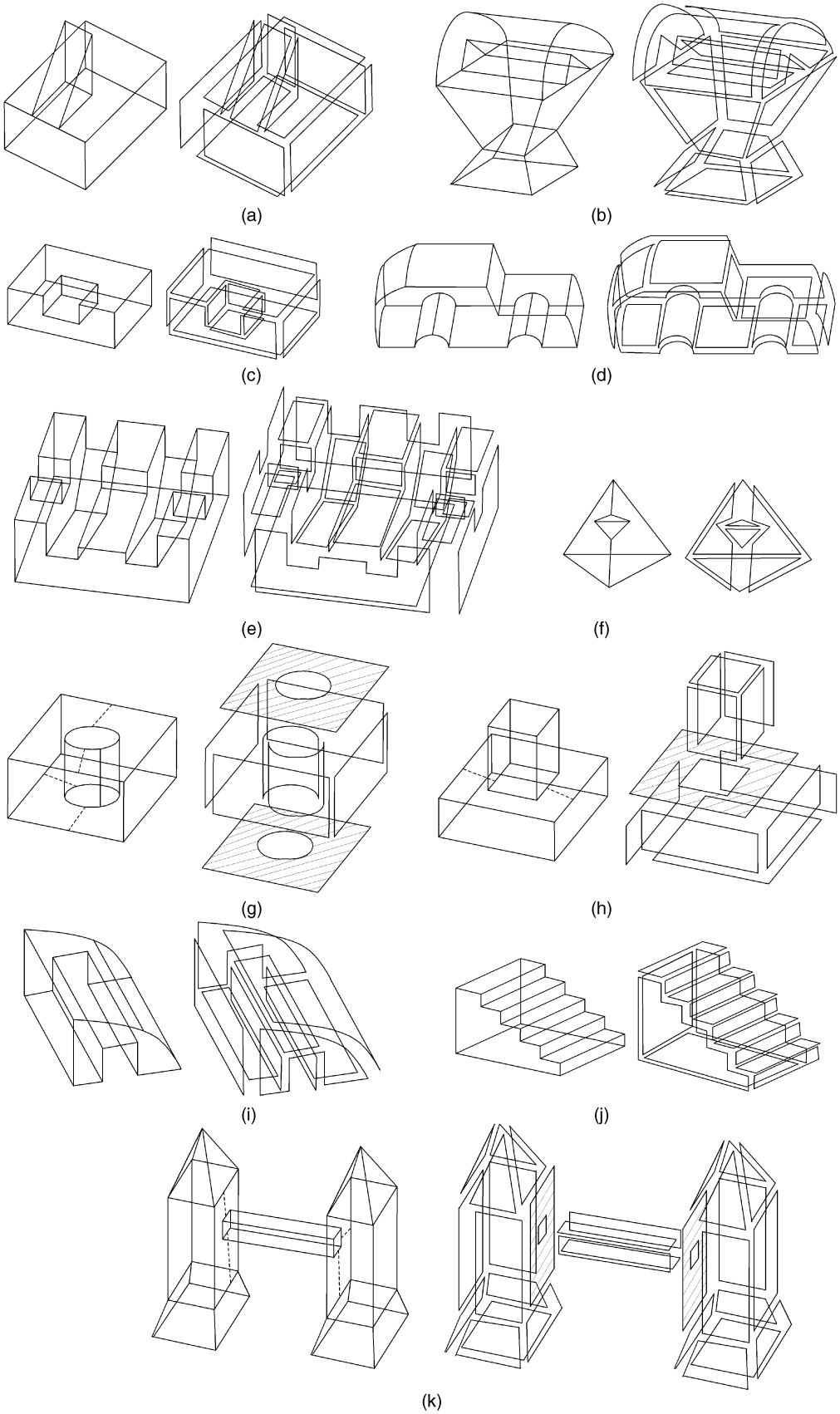

Fig. 22. Eleven line drawings and their faces found. Each of the shaded faces is enclosed by two cycles.

TABLE 1
Results for the Objects in Fig. 22

Drawing	Total cycles	Cycles found by DFSP alg.	Solutions found by backtrack alg.	Faces found finally
(a)	154	10	1	10
(b)	11052	37	1	16
(c)	224	12	2	10
(d)	8348	15	1	13
(e)	3735714	26	1	26
(f)	22	8	2	6
(g)	42	11	1	8
(h)	56	12	1	11
(i)	312	10	1	9
(j)	4228	14	1	14
(k)	1718	32	1	30

provide important information for the reconstruction of its 3D geometry. The face identification is formulated based on a property of manifolds: each edge of a manifold is shared exactly twice by two faces. The two main steps in our method are to search for cycles from a given drawing and to search for real faces from a state-space tree constructed by the cycles. The problem in this formulation lies in the backtrack algorithm taking a very long time to search for solutions if there are too many cycles (> 500, for example), the number of which is generally exponential in the number of vertices of a drawing.

Our strategy in handling the computational problem is to seek special properties which can be used to identify certain cycles early, either as faces or nonfaces, which leads to the removal of most of the cycles to be searched by the backtrack algorithm for real faces. Most of the presented properties relate to planar manifold geometry and the DFSP algorithm based on these properties is suitable for drawings representing planar manifolds. To make this algorithm work with a general manifold (with curved faces), we transform the general manifold to a planar one by simply replacing all its curved edges with straight lines. This scheme is successful if the planar manifold is a good approximation to the curved manifold. A good approximation can always be obtained by adding lines on curved faces. From the examples in this paper, we can see that adding lines on the curved faces is easy and not a burden to a designer. More importantly, doing so allows us to handle more complex manifolds. To deal with a manifold represented by two or more disjoint graphs, our scheme is to treat each graph separately and then to identify the real faces each enclosed by more than one cycle with the help of added dashed lines. From the analysis in Section 3 and the examples in Section 7, it is not difficult to see that our method can deal with various drawings representing manifold objects, including drawings the previous methods cannot handle.

While we have painstakingly investigated the properties established in this paper, we do not exclude the existence of other properties that can allow the state of a cycle to be identified early.

On a 677 MHz Pentium III PC, our method takes less than 0.1 second to find the faces for each of the drawings in Fig. 22. Since the backtrack algorithm is NP-complete, it will suffer from large computation if the number of cycles it has to deal with is still too large after applying the properties to reduce the number. We leave this problem to further research.

ACKNOWLEDGMENTS

The work was supported in part by the AoE-IT, Hong Kong.

REFERENCES

[1] T. Marill, "Emulating the Human Interpretation of Line-Drawings as Three-Dimensional Objects," *Int'l J. Computer Vision,* vol. 6, no. 2, pp. 147-161, 1991.

[2] Y.G. Leclerc and M.A. Fischler, "An Optimization-Based Approach to the Interpretation of Single Line Drawings as 3D Wire Frames," *Int'l J. Computer Vision,* vol. 9, no. 2, pp. 113-136, 1992.

[3] H. Lipson and M. Shpitalni, "Optimization-Based Reconstruction of a 3D Object from a Single Freehand Line Drawing," *Computer-Aided Design,* vol. 28, no. 8, pp. 651-663, 1996.

[4] K. Sugihara, "A Necessary and Sufficient Condition for a Picture to Represent a Polyhedral Scene," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 6, no. 5, pp. 578-586, 1984.

[5] K. Sugihara, "An Algebraic Approach to Shape-from-Image problem," *Artificial Intelligence,* vol. 23, pp. 59-95, 1984.

[6] I. Shimshoni and J. Ponce, "Recovering the Shape of Polyhedra Using Line-Drawing Analysis and Complex Reflectance Models," *Computer Vision and Image Understanding,* vol. 65, no. 2, pp. 296-310, 1997.

[7] P.E. Debevec, C.J. Yaylor, and J. Malik, "Modeling and Rendering Architecture from Photographs: A Hybrid Geometry—and Image-Based Approach," *Proc. SIGGRAPH 96 Conf.,* pp. 11-20, 1996.

[8] A. Turner, D. Chapman, and A. Penn, "Sketching Space," *Computers & Graphics,* vol. 24, pp. 869-879, 2000.

[9] M. Shpitalni and H. Lipson, "Identification of Faces in a 2D Line Drawing Projection of a Wireframe Object," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 10, pp. 1000-1012, Oct. 1996.

[10] S.C. Agarwal and J.W.N. Waggenspack, "Decomposition Method for Extracting Face Topologies from Wireframe Models," *Computer-Aided Design,* vol 24, no. 3, pp. 123-140, 1992.

[11] J. Liu and Y.T. Lee, "A Graph-Based Method for Face Identification from a Single 2D Line Drawing," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 10, pp. 1106-1119, 2001.

[12] S. Bagali and J.W.N. Waggenspack, "A Shortest Path Approach to Wireframe to Solid Model Conversion," *Proc. Third Symp. Solid Modeling and Applications,* pp. 339-349, 1995.

[13] J.S. Hojnicki and P.R. White, "Converting CAD Wireframe Data to Surfaced Representations," *Computer in Mechanical Eng.,* pp. 19-25, Mar./Apr., 1988.

[14] D.E. LaCourse, *Handbook of Solid Modeling.* New York: McGraw-Hill, 1995.

[15] G. Chartrand and O.R. Oellermann, *Applied and Algorithmic Graph Theory.* New York: McGraw-Hill, 1993.

[16] M. Mantyla, *An Introduction to Solid Modeling.* Maryland: Computer Science Press, 1988.

[17] D.A. Huffman, "Impossible Objects as Nonsense Sentences," *Machine Intelligence,* vol. 6, pp. 295-323, B. Meltzer and D. Mitchie, eds. London: Edinburgh Univ. Press, 1971.

[18] M.B. Clowes, "On Seeing Things," *Artificial Intelligence,* vol. 2, pp. 79-116, 1971.

[19] D. Waltz, "Understanding Line Drawings of Scenes with Shadows," *Psychology of Computer Vision,* pp. 19-91, P.H. Winston, ed. New York: McGraw-Hill, 1975.

[20] R. Haralick and L. Shapira, "The Consistent Labeling Problem: Part 1," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 1, no. 2, pp. 173-184, 1979.

[21] J. Malik, "Interpreting Line Drawings of Curved Objects," *Int'l J. Computer Vision,* vol. 1, pp. 73-103, 1987.

[22] M.C. Cooper, "Interpretation of Line Drawings of Complex Objects," *Image and Vision Computing,* vol. 11, no. 2, pp. 82-90, 1993.

[23] M.C. Cooper, "The Interpretations of Line Drawings with Contrast Failure and Shadows," *Int'l J. Computer Vision,* vol. 43, no. 2, pp. 75-97, 2001.

[24] R. Lequette, "Automatic Construction of Curvilinear Solid from Wireframe Views," *Computer-Aided Design,* vol. 20, no. 4, pp. 171-180, 1988.

[25] D. Lysak, "Interpretation of Engineering Drawings of Polyhedral and Nonpolyhedral Objects from Orthographic Projections," PhD thesis, Dept. of Electrical & Computer Eng., Pennsylvania State Univ. 1991.

[26] S. Ablameyko, V. Bereishik, A. Gorelik, and S. Medvedev, "3D Object Reconstruction from Engineering Drawing Projections," *Computing & Control Eng. J.,* vol. 10, no. 6, pp. 277-284, 1999.

[27] M.H. Kuo, "Reconstruction of Quadric Surface Solid from Three-View Engineering Drawings," *Computer-Aided Design,* vol. 30, no. 7, pp. 517-527, 1998.
[28] G. Markowsky and M.A. Wesley, "Fleshing out Wire-Frames," *IBM J. Research and Development,* vol. 24, no. 5, pp. 582-597, 1980.
[29] P.M. Hanrahan, "Creating Volume Models from Edge-Vertex Graphs," *Computer Graphics,* vol. 16, no. 3 pp. 77-84, 1982.
[30] R.D. Dutton and R.C. Brigham, "Efficiently Identifying the Faces of a Solid," *Computer and Graphics in Mechanical Eng.,* vol. 7, no. 2, pp. 143-147, 1983.
[31] M.A. Ganter and J.J. Uicker, "From Wire-Frame to Solid Geometric: Automated Conversion of Data Representations," *Computer in Mechanical Eng.,* vol. 2, no. 2, pp. 40-45, 1983.
[32] S.M. Courter and J.A. Brewer, "Automated Conversation of Curvilinear Wire-Frame Models to Surface Boundary Models: A Topological Approach," *Computer Graphics,* vol. 20, no. 4, pp. 171-178, 1986.
[33] C. Thomasse, "The Graph Genus Problem is NP-Complete," *J. Algorithms,* vol. 10, no. 4, pp. 568-576, 1989.
[34] P. Mateti and N. Deo, "On Algorithms for Enumerating all Circuits of a Graph," *SIAM J. Computing,* vol. 5, no. 1, pp. 90-99, 1976.
[35] E.M. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms: Theory and Practices.* New Jersey: Prentice-Hall, 1977.
[36] R. Sedgewick, *Algorithms in C.* Reading, Mass.: Addison-Wesley, 1990.

Jianzhuang Liu received the BE degree from Nanjing Institute of Posts & Telecommunications, China, in 1983, the ME degree from Beijing University of Posts & Telecommunications, China, in 1987, and the PhD degree from the Chinese University of Hong Kong, China, in 1997. From 1987 to 1994, he was a member of the academic staff in the Department of Electronic Engineering, Xidian University, China. From August 1998 to August 2000, he was a research fellow at the School of Mechanical and Production Engineering, Nanyang Technological University, Singapore. He was a postdoctoral fellow in the Department of Electronic Engineering, the Chinese University of Hong Kong from August 2000 to August 2002. He is now a visiting professor in the Department of Informaiton Engineering at the Chinese University of Hong Kong. His research interests include image processing, computer vision, pattern recognition, and artificial intelligence.

Yong Tsui Lee received the BSc degree in 1977, the MS degree, from the University of Rochester in 1980, and the PhD degree from the University of Leeds in 1983. He worked as a CAD system developer in England after his PhD until 1991, when he joined Nanyang Technological University in Singapore, where he is now an associate professor. His current research interests are in geometric modelling, reverse engineering, sketch input for CAD systems and, generally, 3D data capture for design. He is also working in the development of rapid prototyping technology. He is a member of the IEEE Computer Society.

Wai-Kuen Cham graduated from the Chinese University of Hong Kong in 1979 in electronics. He received the MSc and PhD degrees from Loughborough University of Technology, United Kingdom, in 1980 and 1983, respectively. From 1984 to 1985, he was a senior engineer with Datacraft Hong Kong Limited and a lecturer in the Department of Electronic Engineering, Hong Kong Polytechnic. Since May 1985, he has been with the Department of Electronic Engineering, the Chinese University of Hong Kong, where he is now a professor. His research interests include image coding, image processing and pattern recognition. He is a senior member of the IEEE.