Decomposition of Complex Line Drawings with Hidden Lines for 3D Planar-Faced Manifold Object Reconstruction

Jianzhuang Liu, *Senior Member*, *IEEE*, Yu Chen, *Student Member*, *IEEE*, and Xiaoou Tang, *Fellow*, *IEEE*

Abstract—Three-dimensional object reconstruction from a single 2D line drawing is an important problem in computer vision. Many methods have been presented to solve this problem, but they usually fail when the geometric structure of a 3D object becomes complex. In this paper, a novel approach based on a divide-and-conquer strategy is proposed to handle the 3D reconstruction of a planar-faced complex manifold object from its 2D line drawing with hidden lines visible. The approach consists of four steps: 1) identifying the internal faces of the line drawing, 2) decomposing the line drawing into multiple simpler ones based on the internal faces, 3) reconstructing the 3D shapes from these simpler line drawings, and 4) merging the 3D shapes into one complete object represented by the original line drawing. A number of examples are provided to show that our approach can handle 3D reconstruction of more complex objects than previous methods.

Index Terms—3D reconstruction, divide and conquer, internal face, line drawing, manifold.

1 Introduction and Related Work

A line drawing is the 2D projection of the wireframe of an object. Humans have no difficulty in perceiving the 3D geometry from a 2D line drawing. Emulating this ability is an important research topic in both computer vision and graphics. Line drawings discussed in this paper are with hidden lines visible. These line drawings can be generated by sketching on the screen with a mouse or a tablet PC pen and on paper with a pen. Though it takes more effort for the user to draw such line drawings compared with line drawings without hidden lines, they allow the reconstruction of complete and more complex objects. Much work concerning line drawings with hidden lines has been published in the computer vision literature [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12] and in CAD and graphics [13], [14], [15], [16], [17], [18], [19], [20], [21]. The applications of 3D reconstruction from this kind of line drawings include:

1. providing a flexible sketching interface in current CAD systems [14], [16], [18],
2. providing a 2D sketch query interface for 3D object retrieval from large databases or from the internet [19], [22], [23],
3. interactive generation of 3D models from images [9], [17], [20],
4. automatic conversion of existing industrial wireframe models to solid model [13], [14], and
5. building rich databases for object recognition systems and reverse engineering algorithms for shape reasoning [13], [14].

The earliest work toward 3D reconstruction from single line drawings is line labeling, which focuses on finding a set of consistent labels from a line drawing to test the correctness and/or realizability of the line drawing [3], [4], [5], [16], [24], [25], [26], [27], [28], [29], but it does not explicitly recover a 3D object from a line drawing. Most 3D reconstruction methods from a line drawing assume that the face topology of the line drawing is known in advance. This information can greatly reduce the complexity of the reconstruction. Face identification is not a trivial problem, and many methods have been proposed to find faces from a line drawing [1], [6], [7], [8], [12], [13], [14], [30].

Three-dimensional reconstruction from line drawings is usually formulated as an optimization problem. Marill proposed a criterion of minimizing the standard deviation of the angles in the reconstructed object so that a 2D line drawing can be inflated into a 3D shape [2]. Later, more criteria, such as line parallelism, face planarity, object symmetry, and so on, were proposed to do 3D reconstruction [1], [9], [11], [15], [18], [20], [21], [31], [32], [33]. The methods in [26], [34], and [35] use line labeling and shading information to recover the visible surfaces of 3D polyhedra in images from the edges of these polyhedra. Recently, some attempts [32], [33], [36] were made to recover a complete solid from a line drawing with visible lines only, but these methods are only applicable to simple objects. Almost all previous work on 3D reconstruction from line drawings focuses on planar-faced objects.

- *J. Liu and X. Tang are with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, and with Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China. E-mail: {jzliu, xtang}@ie.cuhk.edu.hk.*
- *Y. Chen is with the Department of Engineering, University of Cambridge, BE 457, Trumpington Street, CB2 1PZ, UK. E-mail: yc301@cam.ac.uk.*

Manuscript received 28 Dec. 2008; revised 1 Sept. 2009; accepted 24 Dec. 2009; published online 25 Feb. 2010.
Recommended for acceptance by F. Kahl.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2008-12-0887.
Digital Object Identifier no. 10.1109/TPAMI.2010.49.

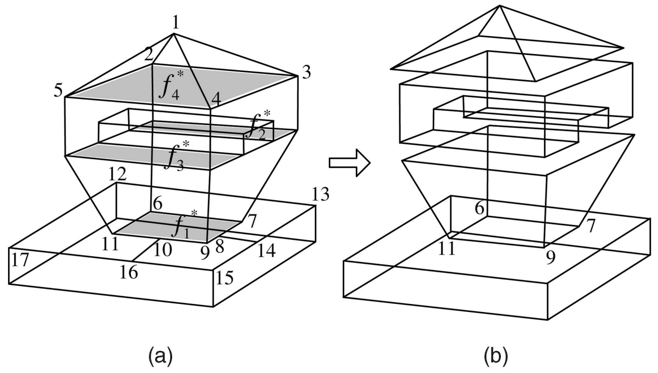


Fig. 1. Decomposition of a line drawing and illustration of some terms. (a) Cycle $(1,3,4,1)$ is a real face. The four shadowed cycles f_{1-4}^* are internal faces. Edges $\{10,16\}$ and $\{8,14\}$ are two artificial lines indicating the coplanarity of cycles $(6,7,9,11,6)$ and $(12,13,15,17,12)$. Edge $\{1,2\}$ is a chord of cycle $(1,5,2,3,1)$. Two real faces $(1,3,4,1)$ and $(1,2,5,1)$ are connected. Edge $\{5,4\}$ and the real face $(1,3,4,1)$ are connected. (b) Four simpler line drawings are decomposed from the four internal faces f_{1-4}^* in (a).

In previous optimization-based methods, the variables of the objective functions are the missing depths of the vertices of a line drawing. These methods work well for simple objects with a small number N of the variables. As N grows, however, it is very difficult for them to find expected objects. This is because, with the nonlinear objective functions in a space of large dimension N, the search for optimal solutions can easily get trapped into local minima. To handle this problem, Liu et al. proposed finding much lower dimensional search spaces where desired objects can be found more easily [9]. This method can tackle 3D reconstruction of more complex objects than previous methods. From [9], we know that the size (dimension) of the search space depends on the degree of reconstruction freedom (DRF) of a line drawing, and the DRF usually increases when the number of the internal faces (see the next section for its definition) of the object increases. In a high-dimensional space, the search for desired objects becomes difficult [9]. Our experiments show that the algorithm in [9] still often fails to obtain an expected 3D object from a line drawing when the geometry of the object becomes more complex with many faces and internal faces.

In this paper, we propose a divide-and-conquer strategy to handle the 3D reconstruction of complex planar-faced manifold objects from line drawings, which is the comprehensive version of our preliminary work published in [10]. In addition to the contents in [10], we discuss how to find internal faces, prove the existence and uniqueness of the partition of a line drawing along an internal face, and provide more experiments. Manifolds belong to a class of most common solids, the definition of which is given in the next section. This approach is based on the fact that a complex object[1] is in general the combination of less complex objects, each of which is easier to recover. Fig. 1 shows an example where a line drawing is decomposed into four simpler ones. Obviously, the 3D reconstruction from each of them is an easier task than the reconstruction from the original line drawing. Our approach includes four steps:

1. identifying the internal faces of an input line drawing,
2. decomposing the line drawing into less complex ones based on the internal faces,
3. reconstructing the 3D shape from each of these simpler line drawings, and
4. merging these 3D shapes into a complete object.

The rest of this paper is organized as follows: Section 2 states the assumptions for the reconstruction problem and defines terms that are frequently used in the paper. In Section 3, we propose our method for the decomposition of a complex line drawing into simpler ones. Section 4 presents the reconstruction algorithm for merging the 3D objects that are recovered from the simpler line drawings. Our experimental results are shown in Section 5, and finally, Section 6 concludes the paper.

2 Assumptions and Terminology

In this paper, we focus on a class of common solids, called manifolds, with planar faces. A line drawing, represented by a single edge-vertex graph with the known x and y coordinates of the vertices, is the parallel or near-parallel projection of the edges of a manifold in a generic view where all of the edges and vertices of the manifold are visible. The generic view assumption means that the topology of the line drawing is preserved under small variations of the viewpoint, which is the assumption made in most previous related work. We also assume that all of the faces of the manifold have been identified from its input line drawing. Face identification from line drawings with hidden lines visible has been studied extensively [1], [6], [7], [8], [12], [13], [14], [30], and the algorithms developed in [7] and [30] can be used to find the faces from a line drawing. For better understanding of the contents in the following sections, we here summarize the terms that appear in the rest of the paper. Many of them are illustrated with the line drawings in Fig. 1.

- **Manifold**. A manifold, or more rigorously two-manifold, is a solid where every point on its surface has a neighborhood topologically equivalent to an open disk in the 2D euclidean space [37]. This paper considers such manifolds that are made up of flat surfaces. A basic property of a manifold is that each edge is shared by exactly two faces [38].
- **Face (real face)**. A face is one of the flat surfaces that make up a manifold. In what follows, we call it a *real face* to distinguish it from an internal face.
- **Internal face**. An internal face is an imaginary face lying entirely inside a manifold M with only its edges visible on the surface. It is not a real face, but can be considered as two coincident real faces of identical shape belonging to two manifolds or two parts of the same manifold which have been glued together to build M.
- **Edge**. An edge of a line drawing is the intersection of two noncoplanar real faces. An edge e is also denoted by $\{v_{e1}, v_{e2}\}$, where v_{e1} and v_{e2} are two vertices of e.
- **Artificial line**. An artificial line is a line used to indicate the coplanar relationship of two cycles.
- **Cycle**. A cycle 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 exists an edge connecting v_i and

1. The complexity of an object is an ambiguous term. In this paper, we say that a manifold object is complex if it has both more than 30 faces and nontrihedral vertices, which causes internal faces or holes.

v_{i+1} for $i = 0, 1, \ldots, n-1$. A cycle is denoted by $(v_0, v_1, \ldots, v_n)$. Since the boundary of a face is a cycle, a face is denoted the same way as a cycle.
- **Chord**. A chord of a cycle is an edge that connects two nonadjacent vertices of the cycle.
- **Vertex set of a cycle**. The vertex set $Ver(C)$ of a cycle C is the set of all the vertices of C.
- **Edge set of a cycle**. The edge set $Edge(C)$ of a cycle C is the set of all the edges of C.
- **Connected faces**. Two faces f_a and f_b are called connected if $Ver(f_a) \cap Ver(f_b) \neq \emptyset$.
- **Connected edge and face**. An edge $e = \{v_{e1}, v_{e2}\}$ is called connected to a face f if $\{v_{e1}, v_{e2}\} \cap Ver(f) \neq \emptyset$ and $e \notin Edge(f)$.
- **Partition of a set**. Given a nonempty set S, a partition $P_S = \{S_1, S_2\}$ is a set of two nonempty subsets S_1 and S_2 of S such that $S_1 \cup S_2 = S$ and $S_1 \cap S_2 = \emptyset$.

An internal face is where two separate manifolds (or two parts of one manifold) are glued together. It may be nonplanar. However, we treat all of the internal faces as planar in this paper, which is true for most manifolds with internal faces. The advantage of this treatment is that when an object is separated along an internal face, this internal face becomes a real planar face and the decomposed line drawings still represent planar-faced manifolds.

3 Decomposition of a Line Drawing

There are many ways to separate the edge-vertex graph of a line drawing into multiple smaller graphs. However, these graphs are meaningless if they do not represent real objects. Obviously, it is desirable that each of the separated line drawings still represents a manifold. We have this as a requirement to design our method for line drawing decomposition. By observing numerous complex objects, especially man-made objects, we can see that most of them are formed by gluing two or more smaller objects together, resulting in internal faces. Therefore, our strategy is to find the internal faces from a line drawing first and then decompose it along the internal faces.

3.1 Classification of Internal Faces

Let an internal face f^* be generated by gluing two real faces f_1 and f_2. Let C_1 and C_2 be the two cycles corresponding to f_1 and f_2, respectively, in the original line drawing. We can classify f^* into one of the two types: 1) C_1 and C_2 have no contact, and 2) C_1 and C_2 have contact (partly or completely). Fig. 1a shows four examples of internal faces, in which f_1^* belongs to type 1 and f_2^*, f_3^*, and f_4^* belong to type 2. For f_4^*, C_1 and C_2 merge into one in the line drawing.

3.2 Decomposition along Internal Faces of Type 1

When f^* belongs to type 1, since C_1 and C_2 do not touch, additional information must be used to indicate the coplanarity of C_1 and C_2 so that correct face identification and reconstruction from the line drawing are possible. Using artificial lines to indicate this coplanarity is the simplest and most straightforward way, which has been used in solid modeling [7], [13]. Two artificial lines connecting two edges of C_1 to two edges of C_2 are added by the user who designs the line drawing. Note that, in general, one artificial line is not enough to indicate the coplanarity because each edge is shared by two real faces in a manifold.

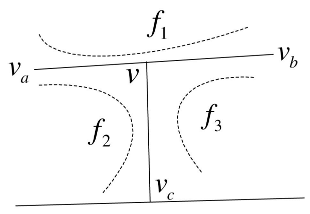


Fig. 2. Part of a line drawing with an artificial line $\{v, v_c\}$.

For an internal face of type 1, if we can detect the two artificial lines, then we can remove them and thus decompose the line drawing along this internal face. In fact, the detection of artificial lines is not difficult, as described below.

Let $\{v, v_a\}$, $\{v, v_b\}$, and $\{v, v_c\}$ be the three edges connected to a vertex v of degree 3, as shown in Fig. 2. If $\{v, v_a\}$ and $\{v, v_b\}$ are collinear, then $\{v, v_c\}$ is an artificial line. This statement is easy to verify.

Assume, on the contrary, that $\{v, v_c\}$ is not an artificial line but an edge. Since the line drawing denotes a manifold, every edge is passed through by two real faces and, hence, three real faces f_1, f_2, and f_3 pass through v (see Fig. 2). According to the assumption that the line drawing is the projection of a manifold in a generic view, the three vertices v_a, v, and v_b are also collinear in 3D space. Thus, the straight line $\overline{v_a v v_b}$ and vertex v_c define a plane in 3D space, implying that f_2 and f_3 are coplanar, which contradicts the definition that an edge is the intersection of two noncoplanar real faces. Therefore, $\{v, v_c\}$ is an artificial line.

After artificial lines are detected, they are removed from the line drawing. Note that when an artificial line is removed, its two vertices in the original line drawing are also removed. For example, when the artificial line $\{10, 16\}$ in Fig. 1a is removed, the two collinear edges $\{11, 10\}$ and $\{10, 9\}$ become one edge $\{11, 9\}$. An internal face of type 1 turns out to be a real face in the decomposed line drawing. The face $(6, 7, 9, 11, 6)$ in Fig. 1b is such an example.

3.3 Detecting Internal Faces of Type 2

Even with the real faces known from a line drawing, detecting internal faces of type 2 is not a trivial problem. In this paper, the detection is performed through a cycle-searching scheme. Since exhaustive searching is computationally expensive, we here develop properties related to internal faces of type 2 so that most cycles that cannot be such an internal face can be eliminated during the search.

In the rest of Section 3, we only consider internal faces of type 2. For concision, we simply use "internal face(s)" to denote "internal face(s) of type 2". When we say two cycles overlap, we mean that their enclosed regions overlap on the 2D line drawing plane. The first two properties below come from the definition and observation of common internal faces. They are useful to develop other subsequent properties. The assumption that all internal faces are planar is also implied.

Property 1. *An internal face except its boundary is invisible from any viewpoint.*

Property 2. *Two coplanar internal faces do not overlap.*

Property 3. *A self-intersecting cycle is not an internal face.*

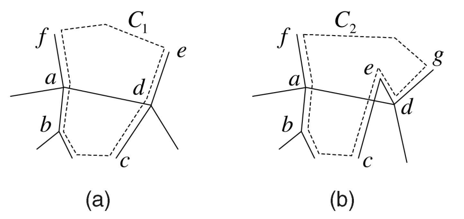


Fig. 3. (a) A cycle $C_1 = (f, a, b, \ldots, c, d, e, \ldots f)$ with a chord $\{a, d\}$ enclosed completely by C_1. (b) Another cycle $C_2 = (f, a, b \ldots c, e, d, g, \ldots f)$ with a chord $\{a, d\}$ enclosed partially by C_2.

Proof. Since the projection of the boundary of a planar face cannot form a self-intersecting cycle [7], an internal face, which is formed by gluing the real faces of two planar manifolds, cannot be self-intersecting either. □

Property 4. *If two cycles share two or more noncollinear edges and overlap, then they cannot both be internal faces.*

Proof. Since the two cycles share two or more noncollinear edges, they must be coplanar if they are internal faces. From Property 2, they cannot both be internal faces. □

Property 5. *A cycle cannot be an internal face if the cycle has a chord that is completely or partially enclosed inside the cycle.*

Proof. If this cycle is an internal face (see Fig. 3), the chord is obviously on the same plane with the cycle. An edge of a line drawing lies on the surface of the manifold and is visible from a certain viewpoint in 3D space. Hence, it causes an interior part of the internal face to be visible, which contradicts Property 1. □

Property 6. *A cycle cannot be an internal face if this cycle and a real face share two or more noncollinear edges and they have an overlapping region.*

Proof. If this cycle is an internal face and shares two or more noncollinear edges with a real face, then the cycle and the real face lie on the same plane. If they further have an overlapping region in the line drawing, then this region is visible, which contradicts Property 1. □

Property 7. *A cycle cannot be an internal face if this cycle and a real face share two or more noncollinear edges and they have edges intersecting in the line drawing.*

Proof. If this cycle is an internal face and shares two or more noncollinear edges with a real face, then the cycle and the real face lie on the same plane. If they further have intersecting edges, then part of the real face must be enclosed by the internal face, which contradicts Property 1. □

Fig. 4 shows several cases where a cycle and a real face share two noncollinear edges and they have an overlapping region in the line drawing. In fact, Property 7 is a special case of Property 6. It is stated explicitly as a separate property because it can be used to reduce fruitless search for internal faces before a path becomes a cycle. For other cases such as Figs. 4b, 4c, and 4d, where the cycle and the real face have no intersecting edges, the cycle can be determined not to be an internal face after the cycle is formed.

Property 8. *A cycle cannot be an internal face if 1) this cycle shares two or more noncollinear edges with a real face and shares another two or more noncollinear edges with another real face, and 2) the two real faces share an edge or they have an overlapping region.*

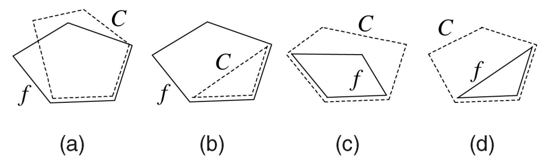


Fig. 4. Several cases where a cycle C and a real face f share two noncollinear edges and they have an overlapping region in the line drawing. (a) C and f have intersecting edges. (b)-(d) C and f have an overlapping region without intersecting edges.

Proof. Let the cycle, the two real faces, be C, f_1, and f_2, respectively. Assume, on the contrary, that C is an internal face. From the first condition, we know that C, f_1, and f_2 all lie on the same plane. If f_1 and f_2 share an edge, as shown in Fig. 5a, the definition that an edge is the intersection of two noncoplanar real faces is violated. In another case, f_1 and f_2 overlap, as shown in Fig. 5b. However, two coplanar real faces of an manifold must not overlap in the line drawing. Therefore, C cannot be an internal face. □

With these properties, we can develop an algorithm to detect the internal faces of a line drawing, which is summarized in Algorithm 1. It is a depth-first search algorithm with the properties incorporated to guide the search. The properties can cut most fruitless branches during the search, and thus considerably speed up the algorithm.

Algorithm 1. Depth-first search of internal faces.
Input: An line drawing $\mathcal{L} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$ where $\mathcal{V}$, $\mathcal{E}$, and $\mathcal{F}$ are the sets of vertices, edges, and real faces, respectively, the adjacency lists $AdjList(v)$ for every vertex $v \in \mathcal{V}$, a 2D array $Shortest(v_1, v_2)$ indicating the length of the shortest path between any two vertices v_1 and v_2, and the maximum search depth D_{max}.

```
1) Initialization: $\mathcal{F}^* \leftarrow \emptyset$; $Label(v) \leftarrow 0$, for every vertex
   $v \in \mathcal{V}$;
2) for every edge $\{u, v\} \in \mathcal{E}$ do
    a) $index \leftarrow 1$; $Label(u) \leftarrow 1$; $Label(v) \leftarrow 1$;
    b) $Path(0) \leftarrow v$; $Path(1) \leftarrow u$;
    c) for every vertex $w_1 \in AdjList(u)$ and $w_1 \neq v$
       do $INTERNAL(u, w_1)$;
    d) $Path(0) \leftarrow u$; $Path(1) \leftarrow v$;
    e) for every vertex $w_2 \in AdjList(v)$ and $w_2 \neq u$
       do $INTERNAL(v, w_2)$;
3) Detect if there are incompatible internal faces in $\mathcal{F}^*$
   according to Property 4;
```

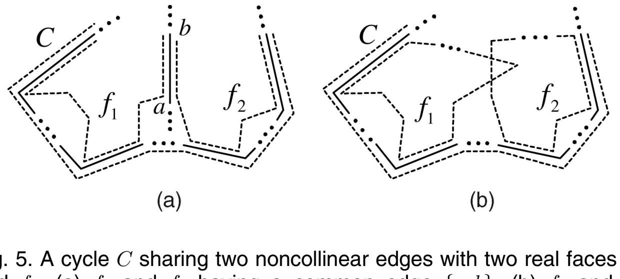


Fig. 5. A cycle C sharing two noncollinear edges with two real faces f_1 and f_2. (a) f_1 and f_2 having a common edge $\{a, b\}$. (b) f_1 and f_2 overlapping.

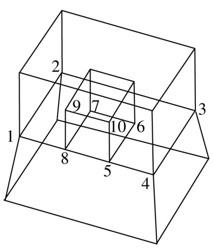


Fig. 6. An example where multiple solutions exist.

```
Output: The set of internal faces in $\mathcal{F}^*$ if there are no incompatible internal faces, or the sets of internal faces if there are incompatible internal faces.
procedure $INTERNAL(u, v)$
  4) $index \leftarrow index + 1$; $Label(v) \leftarrow 1$;
  5) $Path(index) \leftarrow v$;
  6) Check if edge $\{u, v\}$ intersects any previous edges in $Path$ (according to Property 3). If yes, goto 9;
  7) With the set of real faces that share two or more edges with the current path in $Path$, check if this path can form an internal face according to Properties 7 and 8. If no, goto 9;
  8) for every vertex $w \in AdjList(v)$ and $w \neq u$ do
     a) if $Label(w) = 0$ and $index + Shortest(Path(0), w) \leq D_{max}$, then extend the path by calling $INTERNAL(v, w)$;
     b) else if $w = Path(0)$ (a cycle is obtained in this case) then
        i) Check if edge $\{v, w\}$ intersects any previous edges in $Path$ (according to Property 3). If yes, goto 9;
        ii) With all the chords of the cycle in $Path$, check if this cycle can form an internal face according to Property 5. If no, goto 9;
        iii) With all the real faces, check if this cycle can form an internal face according to Property 6. If no, goto 9;
        iv) Put this cycle into $\mathcal{F}^*$ if it is not a real face and is not in $\mathcal{F}^*$ yet;
  9) $index \leftarrow index - 1$; $Label(v) \leftarrow 0$;
end of $INTERNAL$.
```

In the algorithm, an array $Path$ is used to keep the vertices in the current search path; the variable $index$ gives the position of the last-added vertex in $Path$ during the search. A binary label $Label(v)$ is used for every vertex v to denote whether v is in $Path$ or not. In practical applications, we can often set the maximum length D_{max} of internal faces to avoid fruitless search. D_{max} is used with a 2D array $Shortest$ in step 8(a), which indicates the smallest number of edges in a path between any two vertices.

The algorithm starts the search from every edge in the line drawing (see step 2), and calls the procedure $INTERNAL$ recursively to detect possible internal faces. Obviously, this search does not miss any internal faces. In addition, all of our experiments show that the cycles output are indeed internal faces, which suggests that the properties not only speed up the search but also effectively distinguish internal faces from other cycles. The complexity of the algorithm depends on the structure of a line drawing and is exponential in the worst case. However, our experiments in Section 5 indicate that its computational time is acceptable even for complex line drawings.

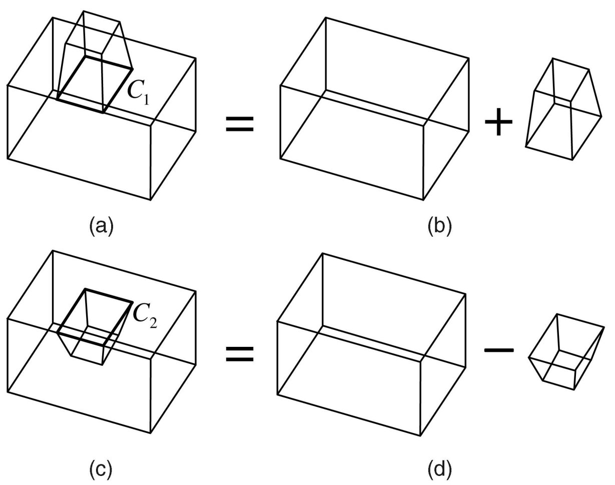


Fig. 7. Union and subtraction that generate internal faces. (a) A manifold. (b) Two manifolds whose union results in the manifold in (a). (c) Another manifold. (d) Two manifolds with which the subtraction of the smaller one from the bigger one results in the manifold in (c). C_1 and C_2 are two internal faces.

For some line drawings, there exist incompatible internal faces, resulting in multiple solutions from a line drawing (see step 3 and "Output" in the algorithm). One example is shown in Fig. 6, which has 15 real faces. From this line drawing, Algorithm 1 finds three internal faces, $C_1 = (1, 2, 3, 4, 5, 6, 7, 8, 1)$, $C_2 = (1, 2, 3, 4, 5, 8, 1)$, and $C_3 = (5, 8, 9, 10, 5)$. Since C_1 and C_2 are incompatible according to Property 4, the algorithm finally outputs two solutions: One is C_1 and C_3 and the other is C_2 and C_3. Note that when C_1 is an internal face, C_1 and the real face $(5, 6, 7, 8, 5)$ are on the same plane; when C_2 is an internal face, C_2 and this real face are on different planes.

The reader may wonder why C_3 in Fig. 6 is also an internal face. In fact, holes and cavities may also generate internal faces, but the definition of internal faces needs to be extended a little. Here, a hole is one that passes through the surface of a manifold while a cavity does not. The manifold in Fig. 1 has a hole, while the manifold in Fig. 6 has a cavity. Next, we discuss this extension with two simple objects in Figs. 7a and 7c.

In Section 2, we define an internal face as a face inside a manifold only with its edges visible on the surface, and it is formed by gluing two manifolds together. One such example is shown in Fig. 7a, where C_1 (denoted by the bold cycle) is the internal face. Obviously, the object in Fig. 7a can be considered as the union (gluing) of the two smaller objects in Fig. 7b. In another case, the object shown in Fig. 7c can be considered as the subtraction of the smaller object from the bigger object shown in Fig. 7d. Comparing the two line drawings (a) and (c), we can see that they have very similar structures and the same topology. The region enclosed by C_1 is invisible and is the common part of the two separate objects in Fig. 7b. There is also such a region enclosed by C_2

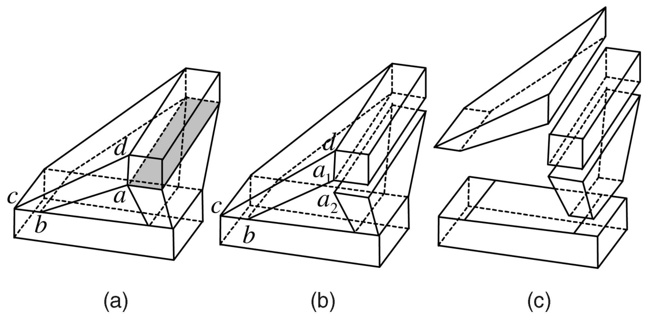


Fig. 8. An example of decomposing a line drawing along its internal faces. (a) The line drawing with one internal face shadowed. (b) The correct partition along this internal face. (c) Partitions along the four internal faces. The hidden edges are shown in dashed lines for easier observation.

(the bold cycle) that is invisible since it is not a real face and is the common part of the two objects in Fig. 7d. Therefore, we also call C_2 an internal face. Note that Algorithm 1 also finds it as an internal face because Properties 1-8 do not prevent it from being so. Now we see that an internal face can be formed either by gluing two manifolds together or by cutting a manifold from another. On the other hand, it is easy to know that gluing two manifolds or cutting a manifold from another may not necessarily result in an internal face.

The cycle $C_3 = (5, 8, 9, 10, 5)$ in Fig. 6 is similar to the cycle C_2 in Fig. 7c. In Fig. 6, since C_3 is compatible with both C_1 and C_2 while C_1 and C_2 are incompatible, Algorithm 1 obtains the two solutions $\{C_1, C_3\}$ and $\{C_2, C_3\}$. Both of them are valid but lead to different decompositions of the line drawing, which is discussed at the end of the next section.

3.4 Decomposition along Internal Faces of Type 2

From an internal face, we decompose the line drawing by recovering the two touching faces that form the internal face. Given a line drawing and its identified real and internal faces, it is not a trivial problem to decompose the line drawing. The main difficulties are: 1) the 3D geometry of the manifold is not available yet, 2) in the 2D projection, the lines connecting to an internal face can be in any direction with respect to the internal face, and 3) when a line drawing is decomposed into two sides along an internal face, for a line that is connected to the internal face in the original line drawing, it is not obvious to which side this line should be connected. For example, the correct decomposition of the line drawing along the shadowed internal face in Fig. 8a is given in Fig. 8b. If the edge $\{a, b\}$ is not connected to a_1 but to a_2, then a wrong decomposition occurs because the real face (a, b, c, d, a) is broken after such a decomposition.

Through the observation of numerous line drawings, we find that the human decomposition of a line drawing along an internal face f^* always satisfies the following two properties:

Property 9. *All of the real faces connected to f^* are partitioned into two sets, $\mathcal{F}_0(f^*)$ and $\mathcal{F}_1(f^*)$.*

Property 10. *Two real faces sharing a common edge connected to f^* (not including any edge of f^*) both appear in either $\mathcal{F}_0(f^*)$ or $\mathcal{F}_1(f^*)$.*

These two properties are easy to verify based on the definition of an internal face. Since f^* is formed by two parts of one manifold or two manifolds, $\mathcal{F}_0(f^*)$ and $\mathcal{F}_1(f^*)$ contain the real faces connected to f^* of the two parts, respectively. Besides, two real faces sharing a common edge connected to f^* belong to one of the two parts instead of belonging to both parts.

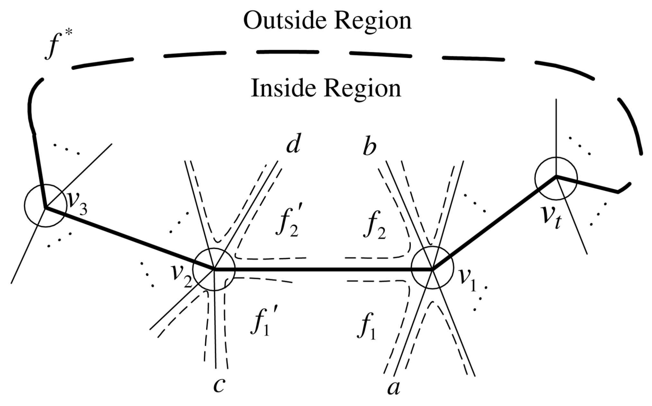


Fig. 9. Part of a line drawing where the internal face f^* is denoted by the bold solid and dashed edges, and all of the neighborhoods $\mathcal{N}_{v_i}$, $1 \le i \le t$, are stretched into 2D disks.

From the decompositions of the line drawings in Fig. 1a and Fig. 8a, it is easy to see that the resulting line drawings in Fig. 1b and Fig. 8c satisfy the two properties. Mathematically, we formulate such a decomposition in the following definition and call it *a partition along an internal face.*

Definition 1. *Let f^* be an internal face, $\mathcal{F}(f^*) = \{f_i\}_{i=1}^m$ be the set of all the m real faces connected to f^*, and $\mathcal{E}(f^*) = \{e_i\}_{i=1}^n$ be the set of all the n edges connected to f^*. A* ***partition along*** *f^* is to find a face set partition $P_{\mathcal{F}(f^*)} = \{\mathcal{F}_0(f^*), \mathcal{F}_1(f^*)\}$ and an edge set partition $P_{\mathcal{E}(f^*)} = \{\mathcal{E}_0(f^*), \mathcal{E}_1(f^*)\}$ simultaneously such that for any $e \in \mathcal{E}_s(f^*)$, it holds that $e \notin Edge(f), \forall f \in \mathcal{F}_{1-s}(f^*)$, where $s = 0, 1$. This partition along f^* is denoted by $P_{f^*} = (P_{\mathcal{F}(f^*)}, P_{\mathcal{E}(f^*)})$.*

The following Property 11 shows that such a partition along an internal face is unique, and the proof of it leads to an algorithm to find this partition.

Property 11. *The partition along an internal face of a line drawing representing a manifold exists and is unique.*

Proof. Let $f^* = (v_1, v_2, \ldots, v_t, v_1)$ be the internal face with t vertices. Let $\mathcal{N}_{v_i}$, $1 \le i \le t$, be a neighborhood around v_i on the surface of the manifold such that $\mathcal{N}_{v_i}$ is topologically equivalent to a 2D open disk and is small enough so that only the edges connected to v_i are contained in $\mathcal{N}_{v_i}$. According to the definition of a manifold, every $\mathcal{N}_{v_i}$ can be stretched into a 2D disk where the real faces passing through v_i are located side by side around v_i and do not overlap, as shown in Fig. 9. Next, we show four properties after all the $\mathcal{N}_{v_i}$ have been stretched into 2D disks.

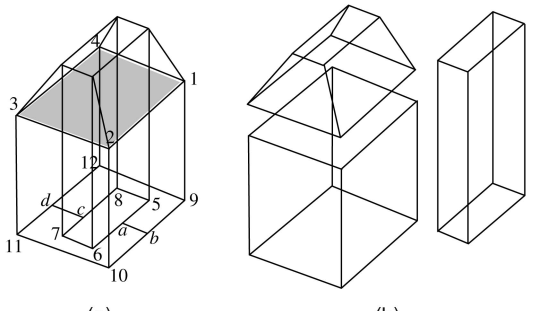
1. At v_i, $1 \le i \le t$, the two edges of f^* partition the real faces passing through v_i into two nonempty sets $\mathcal{F}_0(v_i)$ and $\mathcal{F}_1(v_i)$. This is because f^* is formed by gluing two manifolds or two parts of one manifold (or cutting one from another) and the real faces in $\mathcal{F}_0(v_i)$ and $\mathcal{F}_1(v_i)$ belong to the two manifolds (parts), respectively.

2. When all of the $\mathcal{N}_{v_i}$, $1 \le i \le t$, are stretched into 2D disks and drawn in the 2D plane, f^* separates the 2D plane into inside and outside regions (see Fig. 9). The two faces both passing through edge $\{v_i, v_{i+1}\}$ in the same region are the same face where $1 \le i \le t$ and $v_{t+1} = v_1$. Without loss of generality, consider v_1 and v_2 in Fig. 9. We will prove that $f_2 = (v_2, v_1, b, \dots, v_2)$ and $f_2' = (v_1, v_2, d, \dots, v_1)$ are the same face, and so are $f_1 = (a, v_1, v_2, \dots, a)$ and $f_1' = (c, v_2, v_1, \dots, c)$. Since each edge is shared by exactly two real faces, f_2 can only pass through edge $\{v_2, d\}$ or edge $\{v_2, c\}$. However, f_2 cannot pass through $\{v_2, c\}$ because if we walk clockwise on the surface of the object along the edges of a planar real face, the region enclosed by these edges is always on our right-hand side. Thus, f_2 and f_2' are the same face, and so are f_1 and f_1'.
3. If all of the real faces and the edges connected to f^* in the outside region form two sets, $\mathcal{F}_0(f^*)$ and $\mathcal{E}_0(f^*)$, respectively, and all of the real faces and the edges connected to f^* in the inside region form another two sets, $\mathcal{F}_1(f^*)$ and $\mathcal{E}_1(f^*)$, respectively, then $P_{f^*} = (P_{\mathcal{F}(f^*)}, P_{\mathcal{E}(f^*)})$ with $P_{\mathcal{F}(f^*)} = \{\mathcal{F}_0(f^*), \mathcal{F}_1(f^*)\}$ and $P_{\mathcal{E}(f^*)} = \{\mathcal{E}_0(f^*), \mathcal{E}_1(f^*)\}$ is a partition along f^*. From Fig. 9, it is easy to see that every edge connected to f^* (not including the edges of f^*) appears in two real faces that both belong to the same face set, either $\mathcal{F}_0(f^*)$ or $\mathcal{F}_1(f^*)$. This fact and the above property 1 that indicates $\mathcal{F}_0(f^*) \neq \emptyset$ and $\mathcal{F}_1(f^*) \neq \emptyset$ verify this property 3, which shows the existence of a partition P_{f^*} along f^*.
4. The partition P_{f^*} defined above is unique. Let us consider whether there are other partitions along f^*. If one or more but not all of the real faces are removed from $\mathcal{F}_s(f^*)$ and put into $\mathcal{F}_{1-s}(f^*)$, where $s = 0, 1$, we can see from Fig. 9 that at least two edges connected to f^* will violate Property 10 that a partition along f^* should satisfy, i.e., the two faces sharing such an edge do not both appear in either $\mathcal{F}_0(f^*)$ or $\mathcal{F}_1(f^*)$. If all of the real faces connected to f^* are put in $\mathcal{F}_0(f^*)$ or $\mathcal{F}_1(f^*)$, this does not form a partition along f^*. □

After the partition along f^*, f^* becomes a new real face in the two parts separated from f^*. It is easy to see from Fig. 9 that every point on the edges of f^* has a neighborhood that is topologically equivalent to an open disk in the 2D euclidean space. For example, for a point in the middle of edge $\{v_1, v_2\}$ in Fig. 9, the disk is formed by points in f_2 and the new real face f^*. It is also true for every point inside f^*. Therefore, we have the following property:

Property 12. *After the partition along an internal face, the line drawing (line drawings) still represents (represent) a manifold (manifolds).*

The proof of Property 11 with Fig. 9 already provides all of the elements for developing a simple algorithm to find the partition $P_{f^*} = (P_{\mathcal{F}(f^*)}, P_{\mathcal{E}(f^*)})$ along an internal face f^*. The outline of the algorithm is given in Algorithm 2.



(a) (b)

Fig. 10. (a) A line drawing with a hole passing through the internal face $(1, 2, 3, 4, 1)$. (b) The decomposition result of the line drawing.

Algorithm 2. Partition along an internal face.

```
1) Set $\mathcal{F}_0(f^*)$, $\mathcal{F}_1(f^*)$, $\mathcal{E}_0(f^*)$, and $\mathcal{E}_1(f^*)$ to be empty sets;
2) Put all the real faces connected to $f^*$ into $\mathcal{F}_1(f^*)$;
3) Remove any one face from $\mathcal{F}_1(f^*)$ and put it into $\mathcal{F}_0(f^*)$;
4) for every $f \in \mathcal{F}_1(f^*)$ do
5)     if $f$ and any real face in $\mathcal{F}_0(f^*)$ share an edge connected to $f^*$ (not including the edges of $f^*$) then
6)         Remove $f$ from $\mathcal{F}_1(f^*)$, put it into $\mathcal{F}_0(f^*)$, and goto 4;
7) Put all the edges that are connected to $f^*$ and in the faces in $\mathcal{F}_0(f^*)$ and $\mathcal{F}_1(f^*)$ into $\mathcal{E}_0(f^*)$ and $\mathcal{E}_1(f^*)$, respectively.
```

Since there may be more than one internal face in a line drawing, the algorithm is run repeatedly until all the internal faces have been split. For the line drawing in Fig. 8a, four partitions along the four internal faces decompose it into four simpler line drawings as shown in Fig. 8c.

In Fig. 10, an example is given to show how to decompose a line drawing of an object with a hole passing through an internal face. The hole passes through the object and the internal face $(1, 2, 3, 4, 1)$. Two artificial lines $\{a, b\}$ and $\{c, d\}$ indicate that two cycles $(5, 6, 7, 8, 5)$ and $(9, 10, 11, 12, 9)$ are coplanar. Our algorithm can decompose the line drawing into three simpler ones, as shown in Fig. 10b. Note that when the "hole object" (the right one in Fig. 10b) is removed from the original line drawing, there is no hole in the internal face $(1, 2, 3, 4, 1)$.

Now let us see what decomposition results look like when Algorithm 1 finds multiple solutions from a line drawing. Take the line drawing shown in Fig. 6 as an example in which $\{C_1, C_3\}$ and $\{C_2, C_3\}$ are two solutions with $C_1 = (1, 2, 3, 4, 5, 6, 7, 8, 1)$, $C_2 = (1, 2, 3, 4, 5, 8, 1)$, and $C_3 = (5, 8, 9, 10, 5)$. For convenient observation, it is redrawn in Fig. 11a.

Consider the first solution $\{C_1, C_3\}$ and suppose that Algorithm 2 does the partition beginning with C_1. Then the result is shown in Fig. 11b. Note that since C_3 has been broken in Fig. 11b, a partition based on it is impossible. Furthermore, since the two line drawings in Fig. 11b have no internal faces, the result in Fig. 11b is final. Now suppose that Algorithm 2 starts with C_3 instead of C_1. The first partition result is shown in Fig. 11c. Again C_1 has been broken after the first partition and a partition based on it is impossible. However, from the bigger line drawing in Fig. 11c, Algorithm 1 can find an

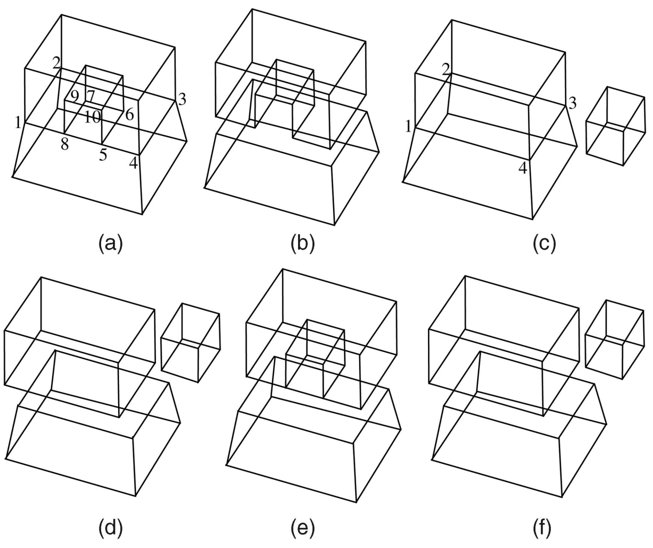


Fig. 11. (a) The original line drawing. (b) Partition result along C_1 from (a). (c) Partition result along C_3 from (a). (d) Further partition result along C_4 from (c). (e) Partition result along C_2 from (a). (f) Further partition result along C_3 from (e).

internal face $C_4 = (1, 2, 3, 4, 1)$. Therefore, Algorithm 2 performs the second partition along C_4 and the final result is given in Fig. 11d.

Consider the second solution $\{C_2, C_3\}$ and suppose that Algorithm 2 carries out the partition along C_2 first. The result is shown in Fig. 11e. Since C_3 is not broken in the upper line drawing in Fig. 11e, further partition along it is performed and the final result is generated as shown in Fig. 11f. If Algorithm 2 starts the partition along C_3 first and then along C_2, the same final result as the one in Fig. 11f will be obtained.

From this example, we can see that even though we have proven that the partition along any internal face is unique, we may obtain multiple decompositions of a line drawing when the line drawing has more than one internal face. In this example, since the results in Figs. 11d and 11f are the same, there are only two different decomposition results, as shown in Figs. 11b and 11d. It should be mentioned that although we have two different decompositions from this line drawing, after combining the manifolds reconstructed from these separate line drawings by our 3D reconstruction method described in the next section, the two combined 3D manifolds based on the two decompositions can be expected to have similar 3D shapes.

4 3D Reconstruction

After decomposing a line drawing along its internal faces of type 1 or 2 (see Section 3.1), we obtain several simpler line drawings, each representing a part of the 3D manifold. Our strategy to obtain the manifold is to reconstruct the 3D shapes from these simpler line drawings first and then merge these 3D shapes together.

As most of the previous methods for 3D reconstruction from a line drawing, we consider that a line drawing is a parallel or near parallel projection of the edges and vertices of a 3D manifold in a generic view. Thus, the x and y coordinates of each vertex are already known, and only the depth (z coordinate) needs to be derived. Since the cycles of the real faces are already available too, the surface of the 3D manifold is recovered if the depths of all the vertices are obtained.

The five steps to reconstruct a complete object from a line drawing are listed in Algorithm 3. Among previous methods, the one in [9] can handle 3D reconstruction from simple line drawings most efficiently. So we use it to carry out steps 3 and 5, with five regularities used for the 3D geometry reconstruction, which are minimizing the standard deviation of angles in the reconstructed object, face planarity, line parallelism, isometry, and corner orthogonality [1], [2], [9], [18]. Step 5 is performed on the complete object with the merged object as the initial shape. Our experiments show that using this step usually generates a better result. When there are multiple decompositions of a line drawing, as discussed in Section 3.4, we can either do the reconstruction from all of these decompositions and output multiple complete manifolds, or just pick any one decomposition to carry out the 3D reconstruction. In the remainder of this section, we give the details of step 4.

Algorithm 3. 3D reconstruction from a line drawing.

```
1) Find all the internal faces of an input line drawing;
2) Decompose the line drawing along the internal
   faces;
3) Reconstruct the 3D objects from these decomposed
   line drawings independently;
4) Merge these 3D objects to form a complete object;
5) Fine-tune the complete object.
```

When all of the 3D simple manifolds are available, the next step is to combine them in an appropriate way so that a complete 3D object is obtained. The basic idea of our merging process is to well match two manifolds' real faces that correspond to one internal face of the original line drawing.

Suppose that two 3D manifolds O_a and O_b share an internal face f^* with K vertices in the original line drawing, the depths of all O_a's vertices are $z_{a1}, z_{a2}, \ldots, z_{aN_a}$, and the depths of all of O_b's vertices are $z_{b1}, z_{b2}, \ldots, z_{bN_b}$. Without loss of generality, also suppose that $z_{a1}, z_{a2}, \ldots, z_{aK}$ are the depths of f^*'s vertices in O_a, and $z_{b1}, z_{b2}, \ldots, z_{bK}$ are the depths of f^*'s vertices in O_b, where z_{ai} corresponds to z_{bi}, $1 \le i \le K$. Since O_a and O_b are reconstructed independently, we usually have a large difference between z_{ai} and z_{bi}, $1 \le i \le K$, and different sizes of f^* in O_a and O_b. We align them according to the depth means (μ_a and μ_b) and the standard deviations (σ_a and σ_b) of f^* in O_a and O_b, where

$$\mu_j = \frac{1}{K}\sum_{i=1}^{K} z_{ji}, \quad j = a, b, \tag{1}$$

$$\sigma_j = \sqrt{\frac{1}{K}\sum_{i=1}^{K}(z_{ji} - \mu_j)^2}, \quad j = a, b. \tag{2}$$

While fixing O_b, we modify the depths of all the vertices of O_a by

$$z'_{ai} = \mu_b + \frac{\sigma_b}{\sigma_a}(z_{ai} - \mu_a), \quad i = 1, 2, \ldots, N_a. \tag{3}$$

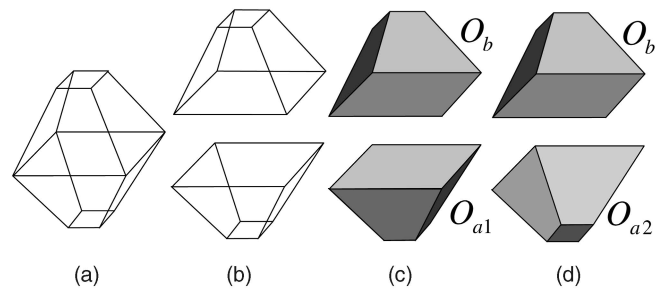


Fig. 12. (a) A line drawing. (b) Two decomposed line drawings. (c) Incompatible objects O_{a1} and O_b. (d) Compatible object O_{a2} and O_b.

Finally, O_a and O_b are merged by forcing their corresponding vertex depths of f^* to be the same:

$$z''_{ai} = z''_{bi} = \frac{z'_{ai} + z_{bi}}{2}, \quad i = 1, 2, \dots, K. \tag{4}$$

Our visual system can interpret a line drawing as a 3D object in two ways, which is well known as the Necker cube reversal perception, and this phenomenon also exists in 3D reconstruction from a line drawing [2]. One example is shown in Fig. 12, where the lower line drawing in Fig. 12b may lead to one of the two 3D objects O_{a1} in Fig. 12c and O_{a2} in Fig. 12d. Incompatible combination of O_{a1} and O_b happens. To solve this problem, we can turn O_{a1} into O_{a2} by multiplying by -1 all the depths of the vertices of O_{a1}. Before doing this, we need to check if two objects O_a and O_b are compatible. Let

$$s = \operatorname{sgn}\left(\sum_{i=1}^{K}(z_{ai} - \mu_a)(z_{bi} - \mu_b)\right). \tag{5}$$

If $s = 1$, O_a and O_b are compatible; if $s = -1$, O_a and O_b are not. Step 4 can be generalized to the case in which O_a and O_b share more than one internal face.

It is not difficult to understand adding objects together. In the last section, we also talk about subtracting one object from another. One example is in Fig. 7d. In fact, the subtraction and the addition of objects in 3D reconstruction work the same way. We always merge objects from their common faces (internal faces). Since we already know the face topology from the original line drawing, we can show the surface of the reconstructed 3D object correctly. For the object in Fig. 7c, the area enclosed by the cycle C_2 is not on the surface of the object because it is not a face.

5 Experimental Results

This section shows a set of examples to demonstrate the performance of our approach. The algorithms are implemented using Visual C++, running on a PC with an Intel Core(TM)2 Quad CPU Q6600 @ 2.4 GHz (only one CPU is used). The maximum search depth D_{max} in Algorithm 1 is set to 10.

For some line drawings, there are multiple decompositions. Let us take the one in Fig. 11a as an example. Fig. 13 shows the two reconstruction results (each displayed in three views) based on the two decompositions in Figs. 11b and 11f. Although the two decompositions are different, the two final complete objects, as expected, do look similar.

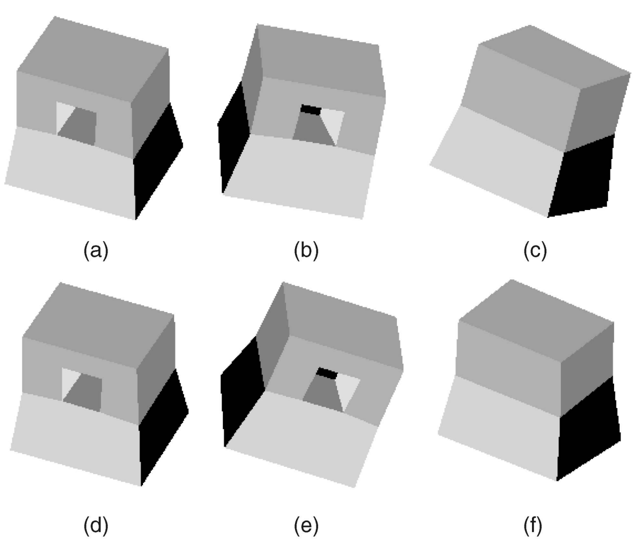


Fig. 13. (a)-(c) The reconstruction result based on the decomposition in Fig. 11b, displayed in three views. (d)-(f) Another reconstruction result based on the decomposition in Fig. 11f, displayed in three views.

The two objects in the views in Figs. 13a and 13d result in the same projection (the original line drawing), which is called the original view. Even though the two shapes look similar, they are not exactly the same. We may use the following formula to measure the difference between them in their original view:

$$\delta(O_a, O_b) = \min_{\varepsilon}\left\{\frac{1}{n}\sum_{i=1}^{n}|z_{ai} - (z_{bi} + \varepsilon)|\right\}, \tag{6}$$

where O_a and O_b denote the two objects, n is the number of vertices of each object, and z_{ai}'s and z_{bi}'s are the z coordinates (depths) of the two objects, respectively, with z_{ai} corresponding to z_{bi}, $1 \le i \le n$. Since the corresponding x and y coordinates of the objects in their original view are the same, we only need to compare the differences between their corresponding z coordinates. Note that if a line drawing comes from the projection of an object $O(z_1, z_2, \dots, z_n)$, then it is also the projection of the object shifted along the z axis by some amount ε, i.e., $O(z_1 + \varepsilon, z_2 + \varepsilon, \dots, z_n + \varepsilon)$. This is why the ε in (6) is used. To compute δ, the range of ε is easy to determine according to the ranges of the z coordinates of the two objects.

For the two objects in Figs. 13a and 13d, the size of their projections (i.e., the line drawing in Fig. 11a) is 161×185. The average difference δ of their z coordinates is 6.3.

Fig. 14 shows a set of complex line drawings. There is only one decomposition from line drawing (a), (e), (f), or (g), but there are 8, 4, 8, and 4 decompositions from line drawings (b), (c), (d), and (h), respectively. In Fig. 14, only one decomposition of each line drawing is given, together with the reconstructed 3D object displayed in two views. From these results, we can see that our algorithm successfully decomposes the line drawings and generates desired 3D objects. It should be emphasized that the objects in Fig. 14 are more complex than the objects given in the previous related papers, in terms of the numbers of real faces and internal faces.

The method in [9] can do 3D reconstruction of more complex objects than other previous methods. However, it

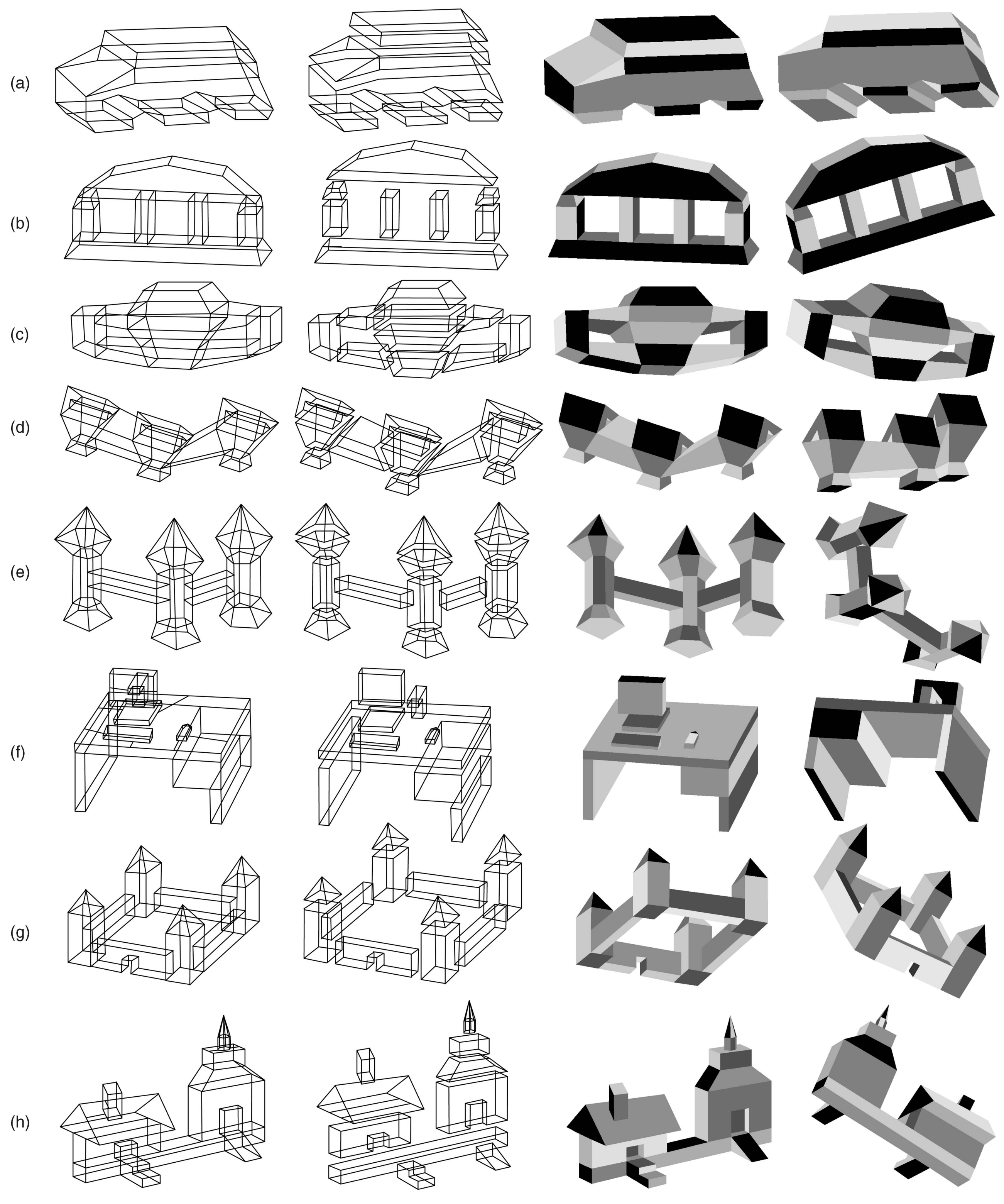


Fig. 14. A set of complex line drawings and their decomposition and reconstruction results.

is still unable to handle these complex line drawings due to too many real and internal faces in each object. From [9], we know that the dimension of the object search space depends on the number of the internal faces of an object. A high dimensional search space renders the search for desired objects more difficult [9]. For example, from the line drawing (e) in Fig. 14, the algorithm in [9] generates an unexpected result as shown in Fig. 15. On the contrary, the 3D objects are easy to reconstruct from the decomposed simple line drawings, and the combination of them into one is not difficult.

From the 3D reconstruction results in Fig. 14, we can see that some are not perfect. For example, in the two second views of the results reconstructed from Figs. 14g and 14h, the

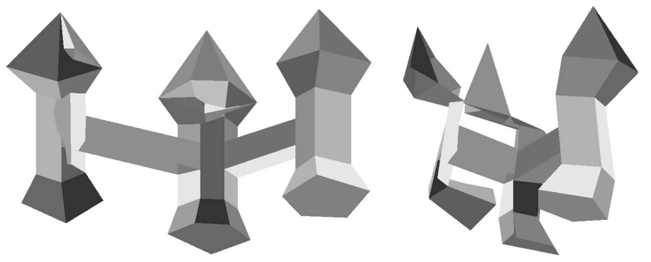


Fig. 15. A failed result in two views reconstructed from the line drawing (e) in Fig. 14 by the algorithm in [9].

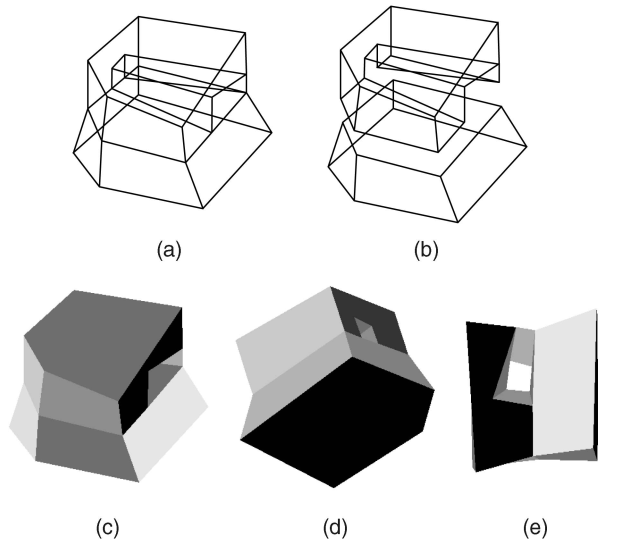


Fig. 16. (a) The line drawing of an irregular object. (b) Decomposition result. (c)-(e) Reconstruction result displayed in three views.

TABLE 1
Times (Seconds) Taken by Searching for the Internal Faces (T1) and Fine-Tuning the Complete Objects (T2) for the Line Drawings in Fig. 14

	a	b	c	d	e	f	g	h
T1	3	4	11	9	8	2	3	6
T2	3	6	5	8	21	14	15	32

four walls of the castle are a little twisted and the two ends of the base of the house are not of the same thickness. These imperfections come from several factors: inaccurate sketches of line drawings, the superstrictness problem in this reconstruction problem [9], approximately optimal solutions obtained by the optimization algorithm used in steps 3) and 5) in Algorithm 3, and the imperfect regularities used to construct the objective function for 3D reconstruction [9], [18].

Common man-made objects such as those in Fig. 14 are usually symmetric or regular with parallel or perpendicular faces. Our algorithm can also cope with irregular objects. Fig. 16a shows the line drawing of such an object. The decomposition and reconstruction results are shown in Figs. 16b, 16c, 16d, and 16e.

The computational time of Algorithm 3 varies with different drawings, depending on their complexity. The main computational cost comes from steps 1 and 5. Table 1 lists the times taken by these two steps for all the line drawings in Fig. 14.

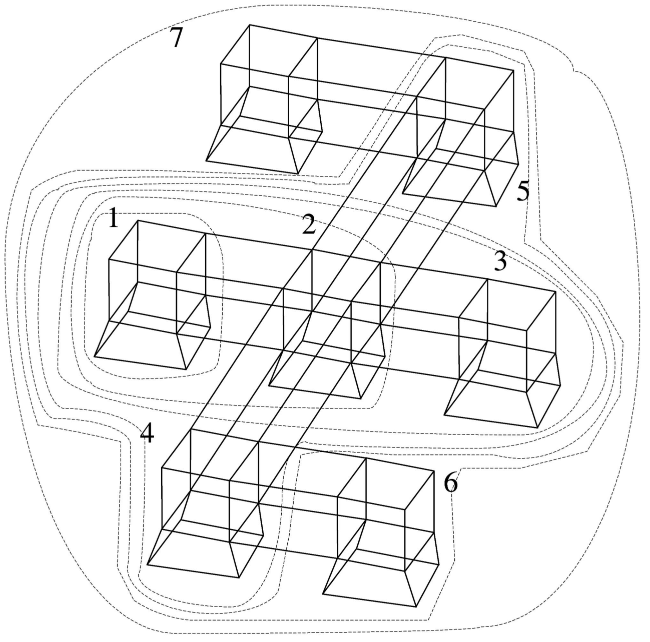


Fig. 17. Seven line drawings of increasing size denoted by seven loops.

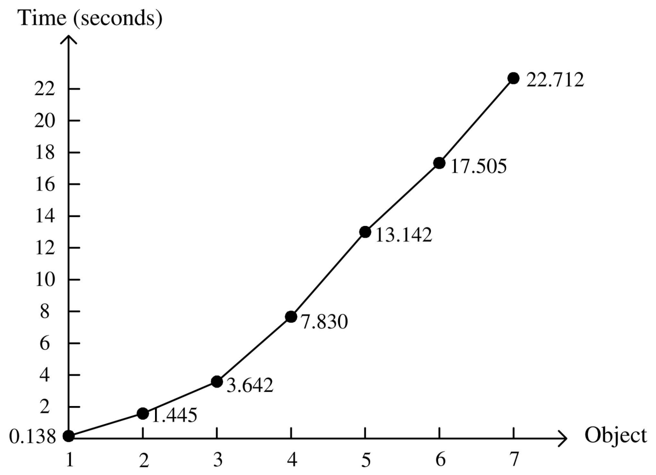


Fig. 18. The time used by Algorithm 1 to find the internal faces of the line drawings in Fig. 17.

The next experiment is to show how the computational time of Algorithm 1 varies for line drawings of increasing size. In Fig. 17, seven objects O_{1-7} (line drawings) are denoted by seven loops with $O_i \subset O_{i+1}$, $1 \leq i \leq 6$. They are shown this way to save space. The time used to find the internal faces of each object is plotted in Fig. 18. It can be seen that the time is not an exponential function of the size for these line drawings, indicating the usefulness of the proposed properties and the efficiency of Algorithm 1 for internal face identification, even though the numbers of non-self-intersecting cycles (potential internal faces) of these line drawings increase exponentially. For O_{1-5}, these numbers are 79, 5,310, 368,465, 4,479,584, and 50,108,386, respectively.

There are two kinds of objects that Algorithm 1 cannot decompose: objects without internal faces, such as the one in Fig. 19a, and objects whose internal faces are self-intersecting, such as the one in Fig. 19b with the self-intersecting internal face $(1, 2, 3, 4, 1)$. Here, we explain more about the second case. In this paper, we reconstruct 3D planar-faced

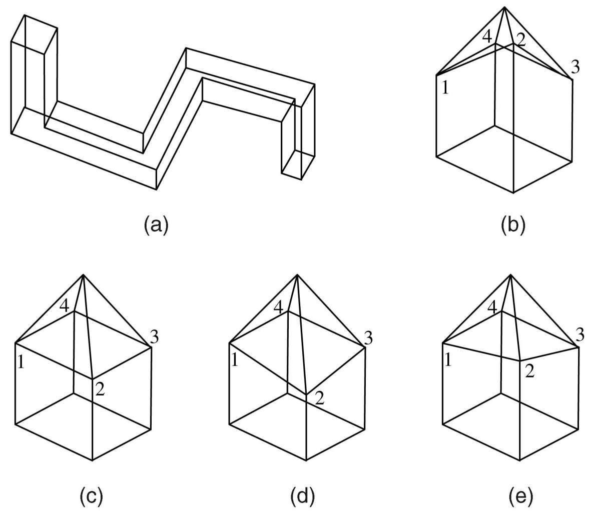


Fig. 19. (a) An object without internal faces. (b) An object with a self-intersecting internal face. (c)-(e) Objects with non-self-intersecting internal faces.

manifold objects from line drawings. It is reasonable to assume that internal faces are also planar and thus non-self-intersecting. If internal faces are allowed to be self-intersecting, two problems will occur: 1) Without the non-self-intersecting constraint, many more cycles in a complex line drawing can be the candidates of internal faces and it is difficult to determine the internal faces. 2) When a line drawing is decomposed, their internal faces become real faces in the decomposed line drawings. These self-intersecting real faces may cause previous 3D reconstruction methods to fail because they are based on planar non-self-intersecting faces. In Figs. 19c, 19d, and 19e, the internal faces $(1, 2, 3, 4, 1)$ are not self-intersecting and our algorithm can handle them easily. Dealing with the cases as in Figs. 19a and 19b is part of the future work.

6 Conclusion

In this paper, we have proposed a novel divide-and-conquer approach to complex 3D planar-faced manifold reconstruction from single line drawings. Our strategy is to

1. identify the internal faces of an input line drawing,
2. decompose the line drawing into simpler ones along its internal faces,
3. reconstruct the 3D shapes from these simpler line drawings, and
4. merge the shapes into a complete object.

The experiments show that our approach can handle more complex objects than previous methods.

Future work includes 1) the improvement of the computational efficiency of the approach, 2) the beautification of the reconstructed objects, and 3) the extension of the work to handle more general planar-faced objects and objects with curved faces.

Acknowledgments

This work was supported by grants from Natural Science Foundation of China (No. 60975029, 61070148), the Research Grants Council of the Hong Kong SAR, China (Project No. CUHK 415408), and the Shenzhen Bureau of Science Technology & Information, China (No. JC200903180635A).

References

[1] Y. Leclerc and M. 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.
[2] 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.
[3] 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, Sept. 1984.
[4] K. Sugihara, *Machine Interpretation of Line Drawings.* MIT Press, 1986.
[5] L. Ros and F. Thomas, "Overcoming Superstrictness in Line Drawing Interpretation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 4, pp. 456-466, Apr. 2002.
[6] J. Liu and Y. 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, Oct. 2001.
[7] J. Liu, Y. Lee, and W.K. Cham, "Identifying Faces in a 2D Line Drawing Representing a Manifold Object," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 12, pp. 1579-1593, Dec. 2002.
[8] J. Liu and X. Tang, "Evolutionary Search for Faces from Line Drawings," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 6, pp. 861-872, June 2005.
[9] J. Liu, L. Cao, Z. Li, and X. Tang, "Plane-Based Optimization for 3D Object Reconstruction from Single Line Drawings," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 30, no. 2, pp. 315-327, Feb. 2008.
[10] Y. Chen, J. Liu, and X. Tang, "A Divide-and-Conquer Approach to 3D Object Reconstruction from Line Drawings," *Proc. IEEE Int'l Conf. Computer Vision,* 2007.
[11] K. Shoji, K. Kato, and F. Toyama, "3-D Interpretation of Single Line Drawings Based on Entropy Minimization Principle," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 90-95, 2001.
[12] 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.
[13] 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.
[14] S. Bagali and J. Waggenspack, "A Shortest Path Approach to Wireframe to Solid Model Conversion," *Proc. Third Symp. Solid Modeling and Application,* pp. 339-349, 1995.
[15] P. Company, M. Contero, J. Conesa, and A. Piquer, "An Optimisation-Based Reconstruction Engine for 3D Modeling by Sketching," *Computers & Graphics,* vol. 28, pp. 955-979, 2004.
[16] P. Company, A. Piquer, M. Contero, and F. Naya, "A Survey on Geometrical Reconstruction as a Core Technology to Sketch-Based Modeling," *Computer & Graphics,* vol. 29, no. 6, pp. 892-904, 2005.
[17] P. Debevec, C. Yaylor, and J. Malik, "Modeling and Rendering Architecture from Photograph: A Hybrid Geometry and Image-Based Approach," *Proc. ACM SIGGRAPH '96,* pp. 11-20, 1996.
[18] 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.
[19] P. Min, J. Chen, and T. Funkhouser, "A 2D Sketch Interface for a 3D Model Search Engine," *Proc. ACM SIGGRAPH '02, Technical Sketches,* p. 138, 2002.
[20] A. Turner, D. Chapman, and A. Penn, "Sketching Space," *Computer and Graphics,* vol. 24, pp. 869-879, 2000.
[21] A. Piquer, R.R. Martin, and P. Company, "Using Skewed Mirror Symmetry for Optimisation-Based 3D Line-Drawing Recognition," *Proc. Fifth IAPR Int'l Workshop Graphics Recognition,* pp. 182-193, 2003.
[22] S. Ortiz, "3D Searching Starts to Take Shape," *Computer,* vol. 37, no. 8, pp. 24-26, 2004.

[23] L. Cao, J. Liu, and X. Tang, "3D Object Retrieval Using 2D Line Drawing and Graph Based Relevance Feedback," *Proc. ACM Int'l Conf. Multimedia,* pp. 105-108, 2006.

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

[25] D. Huffman, "Impossible Objects as Nonsense Sentences," *Machine Intelligence,* vol. 6, pp. 295-323, 1971.

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

[27] M.C. Cooper, "Wireframe Projections: Physical Realisability of Curved Objects and Unambiguous Reconstruction of Simple Polyhedra," *Int'l J. Computer Vision,* vol. 64, no. 1, pp. 69-88, 2005.

[28] M.C. Cooper, "Constraints between Distant Lines in the Labelling of Line Drawings of Polyhedral Scenes," *Int'l J. Computer Vision,* vol. 73, no. 2, pp. 195-212, 2007.

[29] M.C. Cooper, "A Rich Discrete Labeling Scheme for Line Drawings of Curved Objects," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 30, no. 4, pp. 741-745, Apr. 2008.

[30] H. Li, Q. Wang, L. Zhao, Y. Chen, and L. Huang, "nD Object Representation and Detection from Single 2D Line Drawing," *Lecture Notes in Computer Science,* vol. 3519, pp. 363-382. Springer, 2005.

[31] P.A.C. Varley and R.R. Martin, "Estimating Depth from Line Drawings," *Proc. Seventh ACM Symp. Solid Modeling and Application,* pp. 180-191, 2002.

[32] L. Cao, J. Liu, and X. Tang, "3D Object Reconstruction from a Single 2D Line Drawing without Hidden Lines," *Proc. IEEE Int'l Conf. Computer Vision,* vol. 1, pp. 272-277, 2005.

[33] L. Cao, J. Liu, and X. Tang, "What the Back of the Object Looks Like: 3D Reconstruction from Line Drawings without Hidden Lines," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 30, no. 3, pp. 507-517, Mar. 2008.

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

[35] H. Shimodaira, "A Shape-from-Shading Method of Polyhedral Objects Using Prior Information," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 4, pp. 612-624, Apr. 2006.

[36] P.A.C. Varley and R.R. Martin, "A System for Constructing Boundary Representation Solid Models from a Two-Dimensional Sketch—Topology of Hidden Parts," *Proc. First UK-Korea Workshop Geometric Modeling and Computer Graphics,* pp. 113-128, 2000.

[37] M.A. Armstrong, *Basic Topology.* Springer, 1983.

[38] D.E. LaCourse, *Handbook of Solid Modeling.* McGraw-Hill, 1995.

Jianzhuang Liu received the BE degree from Nanjing University of Posts and Telecommunications, P.R. China, in 1983, the ME degree from Beijing University of Posts and Telecommunications, P.R. China, in 1987, and the PhD degree from The Chinese University of Hong Kong in 1997. From 1987 to 1994, he was a faculty member in the Department of Electronic Engineering, Xidian University, P.R. China. From August 1998 to August 2000, he was a research fellow at the School of Mechanical and Production Engineering, Nanyang Technological University, Singapore. Then he was a postdoctoral fellow at The Chinese University of Hong Kong for several years. He is now an assistant professor in the Department of Information Engineering, The Chinese University of Hong Kong. His research interests include computer vision, image processing, machine learning, and graphics. He is a senior member of the IEEE.

Yu Chen received the BE degree from the Department of Electronics Engineering, Tsinghua University, P.R. China, in 2006, and the MPhil degree from the Department of Information Engineering, The Chinese University of Hong Kong, in 2008. He is currently working toward the PhD degree in the Department of Engineering, University of Cambridge, United Kingdom. His research interests include computer vision, image processing, and machine learning. He is a student member of the IEEE.

Xiaoou Tang received the BS degree from the University of Science and Technology of China, Hefei, in 1990, the MS degree from the University of Rochester, New York, in 1991, and the PhD degree from the Massachusetts Institute of Technology, Cambridge, in 1996. He is a professor in the Department of Information Engineering and Associate Dean (Research) on the Faculty of Engineering of the Chinese University of Hong Kong. He worked as the group manager of the Visual Computing Group at Microsoft Research Asia from 2005 to 2008. He received the Best Paper Award from the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009. He was a program chair of the IEEE International Conference on Computer Vision (ICCV) 2009 and is an associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* and the *International Journal of Computer Vision (IJCV)*. His research interests include computer vision, pattern recognition, and video processing. He is a fellow of the IEEE.