Object Cut:
Complex 3D Object Reconstruction Through Line Drawing Separation

Tianfan Xue[1], Jianzhuang Liu[1,2], and Xiaoou Tang[1,2]
[1]Department of Information Engineering, The Chinese University of Hong Kong
[2]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China
{xtf009,jzliu,xtang}@ie.cuhk.edu.hk

Abstract

This paper proposes an approach called object cut to tackle an important problem in computer vision, 3D object reconstruction from single line drawings. Given a complex line drawing representing a solid object, our algorithm finds the places, called cuts, to separate the line drawing into much simpler ones. The complex 3D object is obtained by first reconstructing the 3D objects from these simpler line drawings and then combining them together. Several propositions and criteria are presented for cut finding. A theorem is given to guarantee the existence and uniqueness of the separation of a line drawing along a cut. Our experiments show that the proposed approach can deal with more complex 3D object reconstruction than state-of-the-art methods.

1. Introduction and Related Work

Reconstructing a 3D object from its 2D line drawing is a classic topic in computer vision. The visual system of human beings can interpret a 2D line drawing and perceive its 3D model easily. In order to emulate this ability by a computer vision system, many methods have been proposed in the literature [2], [3], [4], [6], [8], [9], [12], [14]. The applications of 3D reconstruction from 2D line drawings include: user-friendly sketch interface for conceptual 3D designers in CAD systems, 3D query creation for 3D object retrieval, converting existing industrial wireframe models to solid models, and generating 3D objects from images with user sketches [1], [8], [9], [15]. In this paper, we consider line drawings with hidden lines visible. This kind of line drawings allows the reconstruction of complete and complex objects, including their invisible parts. A line drawing with hidden lines visible can be obtained by the sketch of the user on the screen directly or by the scan of the sketch on a piece of paper.

When a line drawing becomes complex, most previous algorithms fail in the reconstruction, getting trapped in local minima due to the large number of variables in the objective functions [12]. Among previous methods, the one in [12] can handle most complex objects. In [12], the authors propose to separate a complex line drawing into multiple simpler line drawings, then independently reconstruct the 3D objects from these line drawings, and finally merge them to form a complete object. This approach well solves the problem mentioned above. Its key step is how to separate a complex line drawing. The authors propose to do it from the *internal faces* of the line drawing. An internal face is a face inside an object with only its edges visible on the surface and these edges are all from the line drawing. However, this method may fail when a complex object has no or too few internal faces. One example is given in Fig. 1(a) where there is only one internal face (Fig. 1(b)), from which the separation is shown in Fig. 1(c). We can see that the bigger object in Fig. 1(c) is still complex.

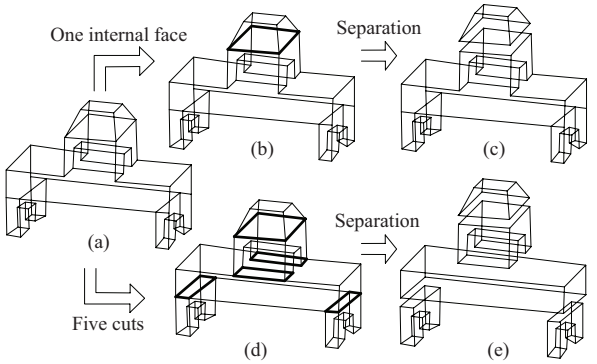

Fig. 1. Comparison between the method in [12] and object cut. (a) A line drawing. (b) Only one internal face. (c) Separation result by [12]. (d) Five cuts. (e) Separation result by object cut.

In this paper, we propose a novel approach, called *object cut*, to decompose a complex line drawing into multiple simpler line drawings. We use cuts but not internal faces to partition a line drawing. An internal face is a special case of a cut. One example is shown in Fig. 1. From Fig. 1(a), our algorithm can find five cuts (Fig. 1(d)), based on which the line drawing is separated into five simpler ones (Fig. 1(e)).

Note that only one of these cuts is an internal face since the other four cuts contain edges not from the original line drawing. Obviously, the reconstruction problem from the line drawings in Fig. 1(e) is easier to handle than those in Fig. 1(c). We develop several propositions and criteria for cut finding, and present a theorem showing the existence and uniqueness of the separation of a line drawing along a given cut. Our experimental results indicate that our approach can deal with 3D reconstruction of more complex objects than previous methods.

2. Assumption and Terminology

In this paper, we consider the reconstruction of the same kind of objects as that in [12], i.e., planar-faced manifolds, which are the most common solids (see below for their definition). A line drawing, represented by a single edge-vertex graph, is the parallel or near parallel projection of the edges of a manifold in a generic view with all its edges and vertices visible. Same as [12], we also assume that the faces of a manifold are available from its line drawing. Finding faces from a line drawing with hidden lines visible has been well studied in previous work [1], [7], [10], [11], [13], [17]. Next, we list the terms used in the rest of this paper, most of which are exemplified in Fig. 2.

- **Manifold**. A manifold, or more specifically 2-dimentional manifold, is a solid where every point on its surface has a neighborhood topologically equivalent to an open disk in the 2D Euclidean space. Manifolds considered in this paper are made up of flat faces. In this kind of manifolds, each edge is shared exactly by two faces [5].
- **Face**. A face is a flat patch of a manifold bounded by edges.
- **Internal Face**. An internal face is a face inside a manifold only with its edges visible on the surface. It is not a real face but is formed by gluing two manifolds together [12].
- **Degree of a vertex**. The number of edges adjacent to a vertex is called the degree of the vertex.
- **Artificial line**. An artificial line is a line used to indicate the coplanarity of two cycles [12].
- **Chord**. A chord of a cycle is an edge or a virtual line inside the cycle that connects two nonadjacent vertices of the cycle. A virtual line does not appear as an edge in the original line drawing.
- **Neighboring face**. If a face f_1 has a sharing edge with another face f_2, then f_2 is called a neighboring face of f_1. If an edge is on the boundary of a face, then the face is called a neighboring face of the edge.
- **Rotation direction of a face**. For a manifold consisting of planar faces, there exists an assignment of a rotation direction for each face simultaneously such that the rotation directions of any two neighboring faces on their sharing edge(s) are opposite [16].

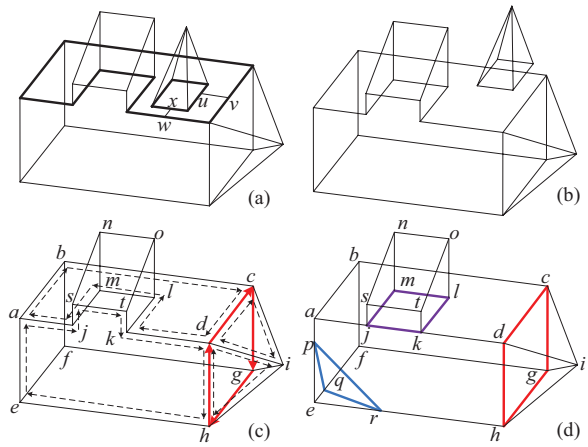

Fig. 2. Examples of most of the terms. In Fig. 2(a), edges (u, v) and (w, x) are two artificial lines. Fig. 2(b) shows the resulted sub-manifolds after removing them. In Fig. 2(c), cycle $(a, j, s, t, k, d, h, e, a)$ is a face. Face (d, i, h, d) is a neighboring face of face (c, i, d, c). The dashed arrows show the rotation directions of four faces $(a, j, s, t, k, d, h, e, a)$, $(a, b, c, d, k, l, m, j, a)$, (c, i, d, c), and (d, i, h, d). Note that on edges (d, k), (a, j), (d, c), (d, i), and (d, h), the rotation directions of their two neighboring faces are opposite. The bold (red) arrows show the rotation direction of cut (c, g, h, d, c), which is arbitrarily assigned. Face $(a, j, s, t, k, d, h, e, a)$ is inconsistent with cut (c, g, h, d, c), but face (d, i, h, d) is consistent with it. Edge (e, f) is a chord of cycle (a, b, f, g, h, e, a), and a virtual line connecting vertices e and g is also a chord of the cycle. In Fig. 2(d), cycles (p, q, r, p), (c, g, h, d, c), and (m, l, k, j, m) are cuts. Face $(a, j, s, t, k, d, h, e, a)$ is a neighboring face of cut (c, g, h, d, c). Face (c, d, i, c) is connected to cuts (c, g, h, d, c). Edge (f, g) is connected to cut (c, g, h, d, c).

- **Cut**. A cut is a planar cycle on the surface of a manifold, formed by cutting the manifold with a plane, consisting of some edges of the manifold and/or new edges on the surface of the manifold, with its enclosed region not on the surface of the manifold. In this paper, we also assign a rotation direction to a cut.
- **Connected faces and edges of a cut**. If a face has at least one sharing point with a cut, then the face is called connected to the cut. If an edge has a sharing point with a cut and is not an edge of the cut, then the edge is called connected to the cut.
- **Neighboring face of a cut**. If a cut has at least one sharing edge with a face, the face is called a neighboring face of the cut.
- **Consistency of a face with a cut**. Given a cut and one of its neighboring faces, if they have the same rotation direction on their sharing edge(s), they are called consistent; otherwise, they are called inconsistent.
- **Partition of a set**. Given a non-empty set S, a partition of S is denoted by $P(S) = (S_0, S_1)$ where S_0 and S_1

are two non-empty sets with $S_0 \cup S_1 = S$ and $S_0 \cap S_1 = \phi$.

Artificial lines, added by the designer, are used to indicate the coplanarity of two cycles in solid modeling [1], [3], [11], [12]. One example is shown in Fig. 2(a) where two artificial lines (u, v) and (w, x) indicate that the two bold cycles are coplanar. Without them, it is impossible to know the geometric relation between the two objects in Fig. 2(b). Detecting artificial lines is an easy task according to the connection between an artificial line and the edges it connects to [12]. After removing the artificial lines in Fig. 2(a), the line drawing becomes two line drawings without artificial lines (Fig. 2(b)). In the next Sections 3 and 4, we consider line drawings without artificial lines.

3. Finding Cuts from a Line Drawing

The main difference between a cut and an internal face is that the former can have part or all of its boundary not from the original line drawing, while the latter consists of edges all from the original line drawing. An internal face is a special case of a cut. The strict requirement of internal faces makes the partition algorithm in [12] fail when a complex manifold is without, or with too few, internal faces (see Fig. 1 for example).

According to the definition of a cut, there is an infinite number of cuts on a manifold. We should find those cuts that really simplify the reconstruction problem. Internal faces are good cuts to partition a line drawing, such as the one in Fig. 1(b) and the cut (c, g, h, d, c) in Fig. 2(d). However, the cut (p, q, r, p) in Fig. 2(d) is not a good cut, and a cut that separates a rectangular solid into two rectangular solids is not a good cut either, because they do not simplify the reconstruction. In the next two sub-sections, we present some propositions and criteria for finding good cuts.

3.1. Propositions for Finding Cuts

Given a cycle on the surface of a manifold, determining whether the cycle is a cut is not a trivial problem due to the lack of 3D geometry in a 2D line drawing. Here we present three propositions to eliminate cycles that cannot be cuts.

Proposition 1. *A cycle is not a cut if it is self-intersecting.*

When we cut a manifold with a plane to form a cut, the cut becomes two visible planar faces, which are not self-intersecting obviously.

Proposition 2. *A cycle is not a cut if it has a chord inside it and the chord is on the surface of the manifold.*

Proof. As shown in Fig. 3, suppose that the cycle $(\ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_{j-1}, v_j, v_{j+1}, \ldots)$ is a cut with a chord (v_i, v_j). Let v be a point in the middle of the chord. If the chord is an edge of the original line drawing, then v inside the cut is on the surface of the manifold. If the chord

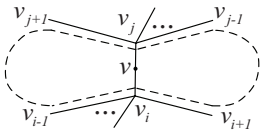

Fig. 3. Part of a line drawing where the cycle has a chord (v_i, v_j).

is not an original edge but is on a face, then v inside the cut is still on the surface of the manifold. Both cases contradict the definition of a cut. □

Proposition 3. *A cycle is not a cut if it has two non-collinear edges belonging to a face and there is an overlapping region between it and the face in the 2D line drawing plane.*

Proof. Since both a cut and a face are planar, if the cut has two non-collinear edges on the face, they must be coplanar. Furthermore, when they have an overlapping region in the 2D line drawing plane, they overlap in the 3D space. Thus, this region inside the cut is on the surface of the manifold, which contradicts the definition of a cut. □

Note that Proposition 3 implies that a face is not a cut.

3.2. Searching for Good Cuts

Through observation of common manifold objects, we find that good cuts that separate a complex manifold into simpler ones usually follow the following three criteria:

Criterion 1. *A good cut should have as few new edges as possible.*

Criterion 2. *A good cut should have as few edges as possible.*

Criterion 3. *A new edge added to form a good cut is usually parallel (or near parallel) to an original edge of the face which the new edge is on.*

We have Criterion 1 because adding too many new edges into the line drawing introduces many new faces on the surface of the object, making the reconstruction more complicated. It also gives priority to finding internal faces. Criterion 2 comes from the observation that a cut with too many edges may lead to an untidy partition of the line drawing. Criterion 3 is based on the fact that a large man-made object is often formed by regular/symmetric smaller objects.

With the criteria and the propositions, we next develop a shortest cycle algorithm to search for good cuts on a search graph. A search graph is constructed based on a line drawing where new edges are added connecting vertices in each face. Fig. 4 shows an example where the dashed lines are part of the new edges. Note that a new edge connecting two vertices on different faces is not added because it is not on the surface of the object. Besides, we do not add new vertices to make the problem too complicated since our current method can already obtain excellent results that are in accordance with our visual perception of the object partition (see the experiments).

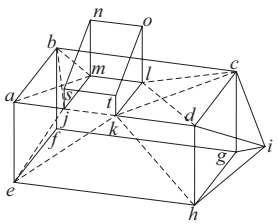

Fig. 4. A searching graph where only part of the new edges (dashed lines) are shown for clarity.

In a search graph, each edge has a weight defined by

$$\omega(e) = \begin{cases} 1, \text{ if } e \text{ is from the original line drawing,} \\ \alpha \cdot (\min_{e' \in S(e)} \gamma(e, e')) + \beta, \text{ otherwise,} \end{cases} \quad (1)$$

where e is an original or new edge of the search graph, α and β are two parameters used to balance the criteria, and we set $\alpha = \beta = 3$ in this paper. $S(e)$ is an edge set consisting of all the edges of the face on which the new edge e lies, and $\gamma(e, e')$ is a function evaluating the parallelism between two edges e and e' defined as:

$$\gamma(e, e') = 1 - \exp(-0.05\theta^2_{e,e'}), \quad (2)$$

where $\theta_{e,e'} \in [0, 90]$ is the angle between e and e'. Take three edges, $e_1 = (a, b)$, $e_2 = (j, k)$, and $e_3 = (k, h)$ in Fig. 4, for example. We have $\omega(e_1) = 1$, $\omega(e_2) \approx 3$, and $\omega(e_3) \approx 6$. Given an original edge of the search graph, the shortest cycle passing through this edge is the one that has the minimum sum of the weights of the edges on this cycle. However, not every shortest cycle can be a cut. The three propositions given in Section 3.1 help to eliminate cycles that cannot be a cut. For example, the shortest cycle passing through edge (c, d) in Fig. 4 is (c, d, i, c), but it cannot be a cut according to Proposition 3 (it is a face actually).

Algorithm 1 lists the steps to find a best cut from a line drawing. In the algorithm, $\omega(\mathfrak{c})$ denotes the total weight of a cut $\mathfrak{c}$. In step 5, the returned shortest path p connected with edge e_i forms a cut, and $\mathfrak{c}$ is always the shortest cut found so far. The procedure $ExtDijkstra$ is an extended version of Dijkstra's shortest path algorithm for finding a shortest path on a weighted graph. The main difference between them is that Propositions 1–3 are incorporated into the search in $ExtDijkstra$. Proposition 1 is used earlier than Propositions 2 and 3 so that most of the paths that cannot be a cut are determined as soon as possible. Since the conditions in Propositions 2 and 3 need to be tested when a cycle is formed, they are used later. Note that in step 8, each path in $PathSet(v_{i,2})$ connected with edge $(v_{i,1}, v_{i,2})$ forms a cycle. Another difference between $ExtDijkstra$ and Dijkstra's algorithm is that we need to keep the geometric positions of the vertices and edges of G' so that Propositions 1–3 are used correctly.

Only one cut is obtained by each running of Algorithm 1. When the current best cut is found, we use the partition method proposed in the next section to separate the line drawing along this cut. Then Algorithm 1 is run again on each separated line drawing. This procedure is repeated until some condition is satisfied. The stopping condition we use is

$$\mu = \frac{\text{sum of the edge weights of the best cut}}{\text{number of faces of the line drawing}} > 1 . \quad (3)$$

The ratio μ is large when the line drawing is simple (i.e., with fewer faces) and the cut is not good (i.e., with a large weight).

Algorithm 1 Finding a best cut

Input: A Line Drawing $G = (V, E)$
Initialization: The best cut $\mathfrak{c} \leftarrow null$; $\omega(\mathfrak{c}) \leftarrow \infty$

1. Create the searching graph $G' = (V, E')$ from G
2. Calculate the weight of each edge of E'
3. **for** each edge $e_i = (v_{i,1}, v_{i,2}) \in E$
4. Call $ExtDijkstra(v_{i,1}, v_{i,2})$
5. **if** a shortest path p is found and $\omega(p, e_i) < \omega(\mathfrak{c})$, **then** $\mathfrak{c} \leftarrow (p, e_i)$

Output: The best cut $\mathfrak{c}$
procedure $ExtDijkstra(v_{i,1}, v_{i,2})$

1. Remove edge $(v_{i,1}, v_{i,2})$ from E'
2. $d(v) \leftarrow \infty$, $PathSet(v) \leftarrow \phi, \forall v \in V$; $d(v_{i,1}) \leftarrow 0$; $PathSet(v_{i,1}) \leftarrow \{(v_{i,1})\}$; $S \leftarrow \phi$, $Q \leftarrow V$
3. **while** $v_{i,2} \notin S$
4. Move vertex $u \in Q$ with minimum $d(u)$ from Q to S
5. **for** each edge $(u, v) \in E'$ with $v \in Q$
6. Extend the paths in $PathSet(u)$ by (u, v); add them to $PathSet(v)$; remove the paths from $PathSet(v)$ that cannot form a cut according to Proposition 1
7. **if** $PathSet(v) = \phi$, **then** $d(v) \leftarrow \infty$; **else** $d(v) \leftarrow \min_{p \in PathSet(v)} \omega(p)$
8. Remove the paths from $PathSet(v_{i,2})$ that cannot form a cut when they are connected with edge $(v_{i,1}, v_{i,2})$ according to Propositions 2 and 3
9. Add edge $(v_{i,1}, v_{i,2})$ back to E'; **if** $PathSet(v_{i,2}) = \phi$, **return** $null$; **else return** the shortest path p in $PathSet(v_{i,2})$

end of $ExtDijkstra(v_{i,1}, v_{i,2})$

4. Separation of a Line Drawing from Cuts

To separate a manifold into simpler ones, cuts of the manifold are found first and then the manifold is separated from the cuts. In Fig. 2(d), some cuts separate faces of the original manifold into sub-faces, which generates a new representation of the manifold. We call this representation an extended manifold, which is defined below.

Definition 1. *Let C be a set of cuts. Let F, E and V be the face set, edge set, and vertex set of the original manifold,*

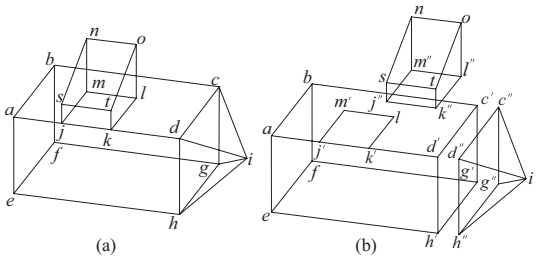

Fig. 5. (a) An extended manifold. (b) Three separated manifolds.

respectively. The extended manifold is an object with its face set $F_{ext}(C)$, edge set $E_{ext}(C)$, and vertex set $V_{ext}(C)$ defined as follows: 1) For each face $f \in F$, if C separates f into two or more sub-faces $\{f_i\}$, then $\{f_i\} \subset F_{ext}(C)$; otherwise, $f \in F_{ext}(C)$. Besides, $F_{ext}(C)$ contains no other faces not from these two cases. 2) $E_{ext}(C) = E \cup \{$edges of the cuts in $C\}$. 3) $V_{ext}(C) = V$.

Since we do not consider cuts with new vertices, $V_{ext}(C)$ is the same as V. It is easy to see that an extended manifold is still a manifold, because all the new edges are on the surface of the original manifold and each edge of the extended manifold is still passed through by exactly two faces of the extended manifold. Some faces of the original manifold are broken into sub-faces by the cuts. For example, Fig. 5(a) is an extended manifold from the manifold in Fig. 2(c) with a cut set $C = \{(c, g, h, d, c), (m, l, k, j, m)\}$. Newly generated faces (a, e, h, d, k, j, a) and (s, j, k, t, s) are the result of cut (m, l, k, j, m). Note that cut (c, g, h, d, c) does not generate any new faces.

Separating an extended manifold along a cut in a 2D line drawing is not trivial. The difficulties include the lack of 3D geometry in a line drawing and the fact that the faces and edges connected to a cut can appear in any directions with respect to the cut. Besides, when the line drawing is separated into two sides along a cut, it is not obvious which side an edge connected to the cut should be put in.

Even though there exist these difficulties, humans are quite easy to obtain a unique separation along a cut. For example, given the extended manifold shown in Fig. 5(a) with the two cuts (c, g, h, d, c) and (m, l, k, j, m), humans always generate the separations in Fig. 5(b). We can see that the faces connected to a cut are separated into two non-empty sets, each of which contains the faces on one side. Besides, for each edge connected to the cut, its two neighboring faces always appear on one side only. This is because if the two neighboring faces appear on different sides, the line drawing cannot be separated into two sides along the cut. Next, a formal definition of the separation is given, which is called *a partition along a cut*.

Definition 2. *Given a cut $\mathfrak{c}$, let the set of all the faces connected to $\mathfrak{c}$ be $F(\mathfrak{c})$, and let the set of all the edges connected to $\mathfrak{c}$ be $E(\mathfrak{c})$. A partition of the extended manifold along $\mathfrak{c}$ is to find a face set partition $P(F(\mathfrak{c})) = (F_0(\mathfrak{c}), F_1(\mathfrak{c}))$ and an edge set partition $P(E(\mathfrak{c})) = (E_0(\mathfrak{c}), E_1(\mathfrak{c}))$ simultaneously such that for any edge $e \in E_m(\mathfrak{c})$, it holds that $e \notin Edge(f)$, $\forall f \in F_{1-m}(\mathfrak{c})$, $m = 0, 1$, where $Edge(f)$ denotes the set of all the edges of face f.*

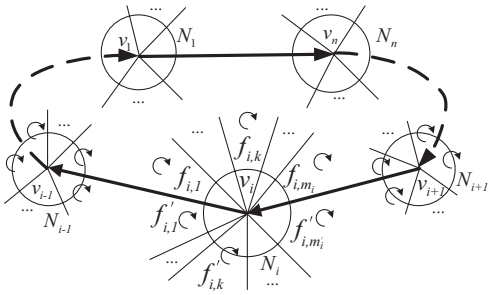

Fig. 6. Part of a line drawing where a cut with n vertices is shown in bold lines, and all the neighborhoods $N_i, i = 1, 2, \ldots, n$, are stretched into 2D open disks.

For example, given the cut $\mathfrak{c} = (m, l, k, j, m)$ in Fig. 5(a), the face set partition along $\mathfrak{c}$ is: $F_0(\mathfrak{c}) = \{(a, e, h, d, k, j, a), (a, b, c, d, k, l, m, j, a)\}$ and $F_1(\mathfrak{c}) = \{(s, j, m, n, s), (s, j, k, t, s), (t, k, l, o, t), (n, m, l, o, n)\}$, and the edge set partition along $\mathfrak{c}$ is: $E_0(\mathfrak{c}) = \{(a, j), (k, d)\}$, $E_1(\mathfrak{c}) = \{(s, j), (n, m), (o, l), (t, k)\}$. Next we give a theorem showing that the partition along a cut exists and is unique[1].

Theorem 1. *The partition of an extended manifold along a cut exists and is unique.*

Proof. Recall the rotation direction of a face. We can assign a rotation direction to every face of the manifold[2] such that any two neighboring faces have opposite rotation directions on their sharing edge(s). Let $\mathfrak{c} = (v_1, v_2, v_3, \ldots, v_n, v_1)$ be a cut with n vertices, and N_i be a neighborhood around v_i on the surface of the manifold such that N_i is topologically equivalent to a 2D open disk and small enough with only the edges connected to v_i contained in N_i. According to the definition of a manifold, every N_i can be stretched into a 2D open disk where the faces passing through v_i are located side by side around v_i without overlap. Besides, we arbitrarily assign a rotation direction to the cut, as shown in Fig. 6, which is helpful to create the face set partition and the edge set partition. Next we state and verify five properties when all the N_i's are stretched into 2D disks.

1) In N_i, two edges (v_{i-1}, v_i) and (v_i, v_{i+1}) of the cut separate the faces passing through v_i into two non-empty sets: $\{f_{i,1}, f_{i,2}, \ldots, f_{i,m_i}\}$ on one side and $\{f'_{i,1}, f'_{i,2}, \ldots, f'_{i,m'_i}\}$ on the other side (see Fig. 6). Neither

[1] In [12], a theorem is given to show that the partition along an internal face exists and is unique. Theorem 1 in this paper is an extension of it.

[2] In this proof, the term *manifold* is used to denote the extended manifold for conciseness.

Algorithm 2 Partition of a line drawing along a cut $\mathfrak{c}$

1. Create the extended manifold with $\mathfrak{c}$
2. $E_0(\mathfrak{c}) \leftarrow \phi$; $E_1(\mathfrak{c}) \leftarrow \phi$; $F_0(\mathfrak{c}) \leftarrow \phi$; $F_1(\mathfrak{c}) \leftarrow \{$faces connected to $\mathfrak{c}$ in the extended manifold$\}$
3. Pick a face $f \in F_1(\mathfrak{c})$ and move it to $F_0(\mathfrak{c})$
4. **for** each face $f \in F_1(\mathfrak{c})$
5. **if** f shares an edge connected to $\mathfrak{c}$ with a face in $F_0(\mathfrak{c})$, **then** move f to $F_0(\mathfrak{c})$
6. **for** each edge e connected to $\mathfrak{c}$
7. **if** e is on a face in $F_m(\mathfrak{c})$, $m = 0$ or 1, **then** $E_m(\mathfrak{c}) \leftarrow e$

set can be empty because otherwise, N_i is not topologically equivalent to a 2D open disk.

2) All the rotation directions of the faces in N_i are either clockwise or counter-clockwise. Without loss of generality, suppose that the rotation direction of $f_{i,1}$ is clockwise (Fig. 6). Since two faces have opposite rotation directions on their sharing edge(s), the rotation direction of $f_{i,2}$ is also clockwise, and so are the rotation directions of $f_{i,3}, f_{i,4}, \ldots, f_{i,m_i}$. Similarly, all the rotation directions of $f'_{i,1}, f'_{i,2}, \ldots, f'_{i,m'_i}$ are also clockwise.

3) Among the four faces, $f_{i,1}$, f_{i,m_i}, $f'_{i,1}$, f'_{i,m'_i}, which pass through the cut and v_i, two of them lying on one side of the cut are consistent with the cut, while the other two lying on the other side are inconsistent with the cut[3]. In Fig. 6, $f_{i,1}$ and f_{i,m_i} lying on the upper side of the cut are consistent with the cut, while $f'_{i,1}$, f'_{i,m'_i} lying on the lower side are inconsistent with the cut. This property is a direct result of property 2).

4) Suppose that $f_{i,1}$ and f_{i,m_i} are consistent with the cut, and $f'_{i,1}$ and f'_{i,m'_i} are inconsistent with the cut, for $i \in \{1, 2, \ldots, n\}$. To create a face set partition, let $F_0(\mathfrak{c}) = \bigcup_{i=1}^{n}\{f_{i,1}, f_{i,2}, \ldots, f_{i,m_i}\}$ and $F_1(\mathfrak{c}) = \bigcup_{i=1}^{n}\{f'_{i,1}, f'_{i,2}, \ldots, f'_{i,m'_i}\}$. That is, $F_0(\mathfrak{c})$ contains the faces consistent with the cut and the faces lying on the same side as these consistent faces, and $F_1(\mathfrak{c})$ contains all other faces passing through v_1, v_2, ..., or v_n. To create an edge partition, let $E_0(\mathfrak{c})$ be the set of edges whose neighboring faces belong to $F_0(\mathfrak{c})$, and $E_1(\mathfrak{c})$ be the set of edges whose neighboring faces belong to $F_1(\mathfrak{c})$. Since for any $e \in E(\mathfrak{c})$, the two neighboring faces of e lie on the same side of the cut, they must belong to the same face partition set, i.e., either $F_0(\mathfrak{c})$ or $F_1(\mathfrak{c})$. Therefore $P(F(\mathfrak{c})) = (F_0(\mathfrak{c}), F_1(\mathfrak{c}))$ and $P(E(\mathfrak{c})) = (E_0(\mathfrak{c}), E_1(\mathfrak{c}))$ form a partition along $\mathfrak{c}$ that satisfies Definition 2. This property and the property 1) show the existence of a partition along $\mathfrak{c}$.

5) The above partition along $\mathfrak{c}$ is unique. To verify this property, suppose that there is another face set partition $(F'_0(\mathfrak{c}), F'_1(\mathfrak{c}))$ along $\mathfrak{c}$ with $F'_0(\mathfrak{c}) \neq F_0(\mathfrak{c})$ (and thus $F'_1(\mathfrak{c}) \neq F_1(\mathfrak{c})$). Since none of $F'_0(\mathfrak{c})$ and $F'_1(\mathfrak{c})$ is empty, there must exist an edge $e \in E(\mathfrak{c})$, the two neighboring faces of which belong to different face partition sets. In this case, the line drawing cannot be separated into two sides because e appears in both sides, which shows that $(F'_0(\mathfrak{c}), F'_1(\mathfrak{c}))$ is not a valid face set partition. □

[3]It is possible that there is only one face on one side. In this case, the two faces merge into one.

Algorithm 3 3D reconstruction

1. Detect and delete all artificial lines
2. For each separated line drawing, find its best cut and partition it along the cut; repeat this step until $\mu > 1$
3. Reconstruct 3D manifolds from the separated line drawings
4. Combine the 3D manifolds to obtain a complete object

Theorem 1 and Definition 2 already provide a method to partition a line drawing along a cut, which is given in Algorithm 2. After the partition along $\mathfrak{c}$, $\mathfrak{c}$ becomes two new faces on the two sides of the partition. In Fig. 5(b), three objects are generated from the partitions along the two cuts (m, l, k, j, m) and (c, g, h, d, c) in Fig. 5(a). Note that the newly face (m', l', k', j', m') is merged with the original face $(a, j', m', l', k', d, c, b, a)$ in the biggest object in Fig. 5(b) by deleting the edges connected to the vertices (m' and l') of degree 2. The following theorem shows that separated line drawings still represent manifolds.

Theorem 2. *After the partition along a cut, the line drawing (line drawings) still represents (represent) a manifold (manifolds).*

Proof. After the partition[4], the new line drawing (line drawings) is (are) formed by the faces of the extended manifold and the two new faces from the cut. We only need to verify that every point on the new faces and their edges has a neighborhood topologically equivalent to a 2D open disk. Obviously, every point inside each new face satisfies this requirement. Let p be a point on an edge of a new face (the cut), say, at the middle of edge (v_i, v_{i-1}) in Fig. 6 without loss of generality. It is easy to find such a neighborhood around p, which is formed by points in $f_{i,1}$ or $f'_{i,1}$, edge (v_i, v_{i-1}), and the new face. □

[4]Note that one partition may or may not separate a line drawing into two disjoint line drawings.

5. 3D Reconstruction from a Line Drawing

After partitioning a line drawing along its cuts, we reconstruct 3D manifolds from these separated line drawings and then obtain the complete large 3D object through the combination of these smaller 3D manifolds.

It is not difficult to deal with 3D reconstruction from a separated line drawing because it is simple enough (see the experiments). We use the method in [9] to carry out this work. The basic idea of this method is to derive the z-coordinates of all the vertices by minimizing an objective function. Since a line drawing is considered as a parallel

projection of a manifold and its face topology is known, the 3D object is obtained when all the z-coordinates are derived. More details can be found from [9]. After constructing the smaller 3D objects from all the separated line drawings, we merge them together to have a complex large object using the method in [12].

Algorithm 3 lists our complete algorithm to do 3D reconstruction from a complex line drawing.

6. Experiments

A set of examples is given in this section to demonstrate the performance of our algorithm. The problem in [12] is that it may fail when a complex line drawing has too few internal faces. For example, it can only separate the line drawing in Fig. 1(a) into the two line drawings in Fig. 1(c). The larger line drawing in Fig. 1(c) is still too complex.

Fig. 7 presents a number of complex line drawings together with their partition and reconstruction results by our algorithm. From the second column of Fig. 7, we can see that our algorithm successfully finds good cuts to separate the line drawings, which are in accordance with our visual partitions. For the two objects in Figs. 7(a) and (b), our algorithm and the one in [12] obtain the same partition results. Note that besides the separations from the artificial lines, there is only one internal face in line drawing (d) or (e), and no internal face in line drawing (f), (g), or (h).

Because our algorithm can separate the complex line drawings into very simple line drawings based on the found cuts, the 3D reconstruction from these line drawings becomes much easier. From the third and forth columns in Fig. 7, we can see that the 3D objects are reconstructed very well. Besides, all the line drawings given in [12] can be dealt with by our algorithm because internal faces are special cases of cuts.

The computational time of Algorithm 3 depends on the complexity of a line drawing. It ranges from 10 to 112 seconds for the line drawings in Fig. 7. The algorithm is implemented using C++ and runs on a PC with 2.4GHz Intel Core2 CPU. Steps 3 and 4 consume the majority of the time, while steps 1 and 2 take about 1 second only for each of the line drawings.

7. Conclusions

In this paper, we propose to separate a complex line drawing from cuts, which include internal faces as a special case. We develop several propositions and criteria for cut finding. We also present a theorem that guarantees the existence and uniqueness of the partition of a line drawing along a cut. Our algorithm can tackle 3D reconstruction of more complex solid objects than previous algorithms. Future work includes the extension of this approach to 3D reconstruction of more general objects such as non-manifolds and objects with curved surfaces.

Acknowledgement

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

References

[1] S. Agarwal and J. Waggenspack. Decomposition method for extracting face topologies from wireframe models. *Computer-Aided Design*, 24(3):123–140, 1992.

[2] 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. PAMI*, 30(3):507–517, 2008.

[3] Y. Chen, J. Liu, and X. Tang. A divide-and-conquer approach to 3D object reconstruction from line drawings. *ICCV*, 2007.

[4] M. Cooper. *Line Drawing Interpretation*. Springer, 2008.

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

[6] Y. Leclerc and M. Fischler. An optimization-based approach to the interpretation of single line drawings as 3D wire frames. *IJCV*, 9(2):113–136, 1992.

[7] H. Li, Q. Wang, L. Zhao, Y. Chen, and L. Huang. nD object representation and detection from simple 2D line drawing. *LNCS*, 3519:363–382, 2005.

[8] H. Lipson and M. Shpitalni. Optimization-based reconstruction of a 3D object from a single freehand line drawing. *Computer-Aided Design*, 28(8):651–663, 1996.

[9] J. Liu, L. Cao, Z. Li, and X. Tang. Plane-based optimization for 3D object reconstruction from single line drawings. *IEEE Trans. PAMI*, 30(2):315–327, 2008.

[10] J. Liu and Y. Lee. A graph-based method for face identification from a single 2D line drawing. *IEEE Trans. PAMI*, 23(10):1106–1119, 2001.

[11] J. Liu, Y. Lee, and W. Cham. Identifying faces in a 2D line drawing representing a manifold object. *IEEE Trans. PAMI*, 24(12):1579–1593, 2002.

[12] J. Liu and X. Tang. Decomposition of complex line drawings with hidden lines for 3D planar-faced manifold object reconstruction. *IEEE Trans. PAMI*, to appear.

[13] J. Liu and X. Tang. Evolutionary search for faces from line drawings. *IEEE Trans. PAMI*, 27(6):861–872, 2005.

[14] T. Marill. Emulating the human interpretation of line-drawings as three-dimensional objects. *IJCV*, 6(2):147–161, 1991.

[15] P. Min, J. Chen, and T. Funkhouser. A 2D sketch interface for a 3D model search engine. *ACM SIGGRAPH, Technical Sketches*, page 138, 2002.

[16] H. Seifert. *Seifert and Threlfall: A Textbook of Topology*. Academic Press, 1980.

[17] M. Shpitalni and H. Lipson. Identification of faces in a 2D line drawing projection of a wireframe object. *IEEE Trans. PAMI*, 18(10):1000–1012, 1996.

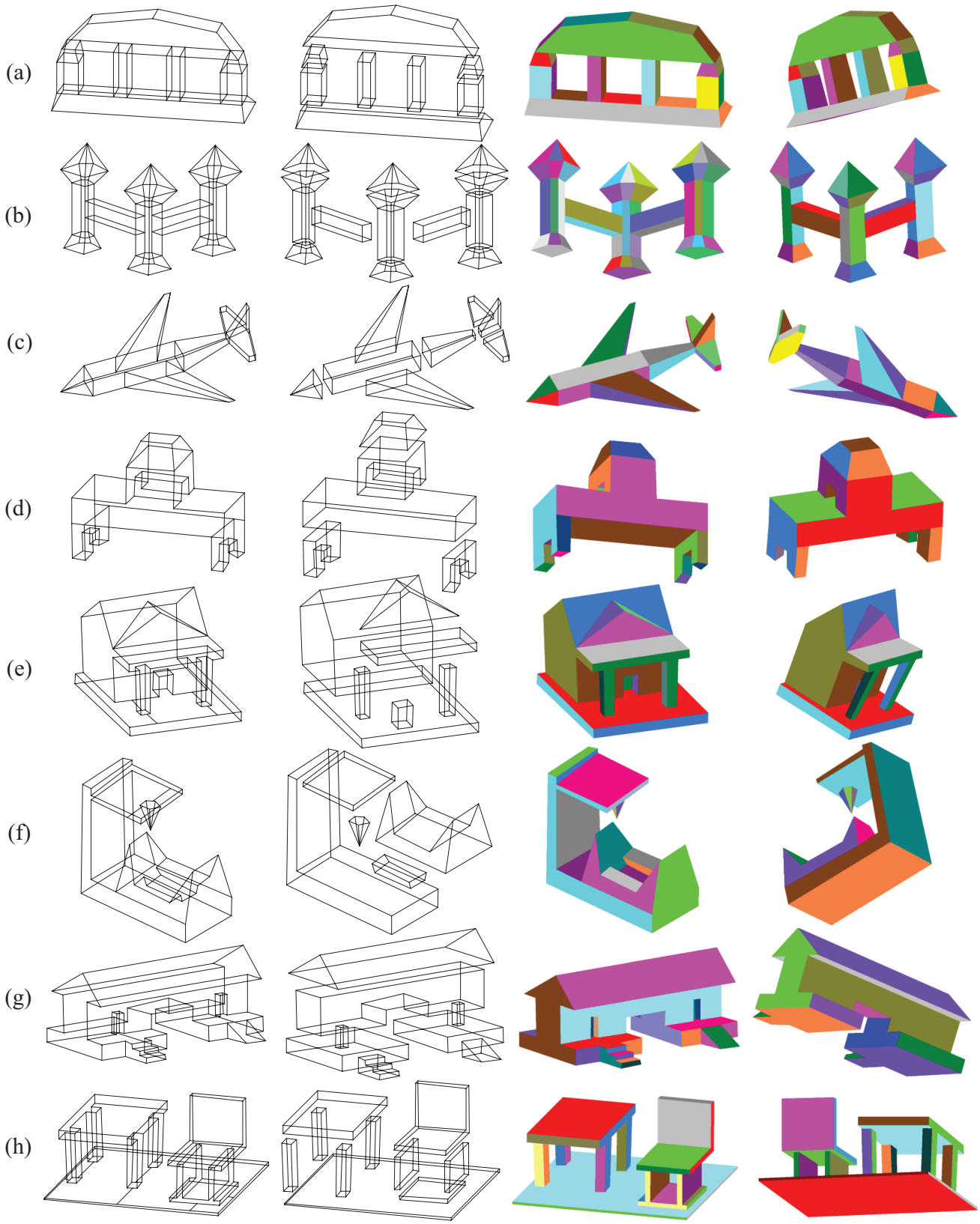

Fig. 7. Experimental results on a set of complex line drawings (a)–(h) by our algorithm. The second column shows the partitions of the line drawings. Each reconstructed 3D object is displayed in two views with its faces illustrated by different colors.