3D Reconstruction of Curved Objects from Single 2D Line Drawings

Yingze Wang[1] Yu Chen[2] Jianzhuang Liu[1] Xiaoou Tang[1]
[1]Department of Information Engineering
The Chinese University of Hong Kong
{wyz007,jzliu,xtang}@ie.cuhk.edu.hk
[2]Department of Engineering
University of Cambridge
yc301@cam.ac.hk

Abstract

An important research area in computer vision is developing algorithms that can reconstruct the 3D surface of an object represented by a single 2D line drawing. Previous work on 3D reconstruction from single 2D line drawings focuses on objects with planar faces. In this paper, we propose a novel approach to the reconstruction of solid objects that have not only planar but also curved faces. Our approach consists of four steps: (1) identifying the curved faces and planar faces in a line drawing, (2) transforming the line drawing into one with straight edges only, (3) reconstructing the 3D wireframe of the curved object from the transformed line drawing and the original line drawing, and (4) generating the curved faces with Bezier patches and triangular meshes. With a number of experimental results, we demonstrate the ability of our approach to perform curved object reconstruction successfully.

1. Introduction

Single 2D line drawings provide a straightforward, easy way of illustrating 3D objects. The human vision system has the ability to interpret 2D line drawings as 3D objects without difficulty. How to bestow this ability on a computer vision system is an important topic. A number of publications have been devoted to this research in the major computer vision literature [8], [15], [20], [21], [22], [23], [25], [29], [30], [31], [32], [33]. The applications of this research include: flexible sketching input for conceptual designers who tend to prefer pencil and paper to mouse and keyboard [3], [19], [22]; providing rich databases to object recognition systems and reverse engineering algorithms for shape reasoning [2], [3]; automatic conversion of industrial wireframe models to solid models [2], [14]; interactive generation of 3D models from images [13], [20], [35]; friendly user interface for 3D object retrieval [5], [27] .

Here a 2D line drawing is defined as the parallel or nearly-parallel projection of a wireframe object in a generic view where all the edges (including silhouettes) and vertices of the object are visible, and the line drawing can be represented by a single edge-vertex graph. A line drawing with hidden lines shown makes it possible to reconstruct a complete object, including its back, from the line drawing. Such line drawings are generated by the user (designer) or come from existing industrial wireframe models. In what follows, we call an object with only planar faces a planar object, and call an object with at least one curved face a curved object. A planar solid is a polyhedron.

There have been a number of papers discussing 3D reconstruction from single 2D line drawings [6], [8], [15], [19], [20], [25], [28], [29], [30], [32], [33], [35], [36]. However, they handle only line drawings of planar objects. Reconstruction of curved objects is obviously a more challenging problem. In this paper, we propose an approach to the 3D reconstruction from line drawings of solids with not only planar but also curved faces. Given a line drawing LD_a, we define some rules to differentiate between curved faces and planar faces based on the result of face identification. Then LD_a is transformed into another line drawing LD_b with only straight lines. From LD_a and LD_b, the reconstruction of the 3D wireframe of the curved object is carried out using several regularities. Finally, the curved faces are recovered by developing Bezier surface patches and triangular meshes.

2. Related Work

Related work on the interpretation of line drawings can be classified into three groups: (a) line labeling, (b) 3D reconstruction from multiple views of wireframe models, and (c) face identification and 3D reconstruction from single line drawings with hidden lines visible. Line labeling focuses on finding a set of consistent labels from a line drawing without hidden lines in order to test if it is legal, and/or on 3D reconstruction based on such a labeled line drawing [9], [10], [11], [24], [32], [33]. Methods in the second group try to reconstruct a 3D CAD model from its multiple orthographic projections [1], [17], [37]. More information can be found from three orthographic views for the reconstruction task than from a single projected view. Our work belongs to the third group.

Face identification from a line drawing is a necessary step. An object consists of faces. If the face configuration of an object is known before the reconstruction of its 3D geometry, the complexity of the reconstruction will be reduced significantly. Finding the faces from a line drawing is not a trivial problem. Much effort has been made in this area [2], [3], [15], [18], [21], [22], [23], [31].

3D reconstruction from a line drawing is usually formulated as an optimization problem using some regularities. Marill [25] presented his method based on a very simple regularity (criterion): minimizing the standard deviation of the angles (MSDA) in the reconstructed object. Shoji et al. [30] presented the criterion of minimizing the entropy of angle distribution. Leclerc and Fischler's approach [15] considers not only the MSDA, but also the regularity of face planarity for planar object reconstruction. The methods in [28] and [36] concentrate on the reconstruction of symmetric polyhedra by developing a regularity of model symmetry. Lipson and Shpitalni [19] took Leclerc and Fischler's work further using more regularities for the reconstruction such as line parallelism, isometry, corner orthogonality, and skewed face orthogonality. Liu et al. [20] proposed to carry out the optimization in a low-dimensional space for more robust object reconstruction. Chen et al. [8] used a divide-and-conquer strategy to deal with complex object reconstruction.

The above 3D reconstruction methods handle only planar objects. Although little work is available to tackle curved objects, some efforts [5], [7], [16], [26] do try to deal with this problem. In [5], Cao et al. used 3D objects reconstructed from 2D line drawings as the input for 3D object retrieval. The method needs the user to manually add lines on some curved faces so that the line drawing of an approximate planar object can be obtained. Although [7] and [16] can deal with simple curved objects, they focus on architectural modeling where the faces are mainly planar. Both need the user to help derive camera parameters and construct a model in a progressive way: one primitive after another in [7] and one polygon after another in [16]. The work [26] also uses a progressive way to do 3D reconstruction of objects with simple curved faces. One limitation of [26] is that it requires the edges of an object exhibit pronounced angular trends so that an underlying axis system can be found. Compared with these methods, our work can automatically reconstruct complex curved solid objects.

3. Assumptions and Terminology

The following three assumptions are made before the reconstruction problem is formulated. (a) A line drawing, represented by a single edge-vertex graph, is the parallel or near-parallel projection of a wireframe manifold object in a generic view where all the vertices and edges of the object are visible. (b) Every curved edge of a line drawing is the projection of a 3D planar curve. (c) All the faces of a manifold that a line drawing represents are available.

A line drawing in a generic view means that no two vertices appear at the same position, no two edges overlap in the 2D projection plane, 3D non-collinear edges are not projected as collinear edges, and no 3D curves are projected as straight lines. In this paper, we focus on a class of most common solids, called manifolds (see below for their definition), and we refer to a polyhedron as a planar manifold. The second assumption requires that each 3D curve is on a 3D plane. This is reasonable because given a 2D curve in a line drawing, we interpret it as a 3D planar curve in most cases. There have been a number of algorithms developed for face identification from line drawings such as [18] and [22], as mentioned in Section 2. This is not the focus of this paper and is why we have the third assumption. For better understanding the content in the following sections, we summarize the terms that appear in this paper.

- **Manifold**. A manifold, or more rigorously 2-manifold, is a solid where every point on its surface has a neighborhood topologically equivalent to an open disk in the 2D Euclidean space. A basic property of a manifold is that each edge is shared exactly by two faces [22].
- **Face**. A face is one of the surface patches of a manifold bounded by edges.
- **Generalized polyhedron**. A generalized polyhedron is represented by a line drawing whose edges are all straight lines, transformed from a line drawing of a curved manifold. It is not a real polyhedron since it has non-planar generalized faces that are defined below.
- **Generalized face**. A generalized face, only existing in a generalized polyhedron, is a face that is not subject to the planarity constraint. It corresponds to a curved face in the line drawing of a curved manifold.
- **Silhouette**. A silhouette is defined as an edge with one adjacent face in front and the other at the back and it is satisfied that the two faces are C^1 continuous along such an edge in 3D space. At every point along a silhouette, the surface orientation is normal to the line of sight and to the tangent to the silhouette [4]. It can be identified by the two faces adjacent to it and the property of its vertices [10].
- **Artificial line**. An artificial line[1] is a line used to indicate that two cycles lie on the same plane or the same curved surface.
- **Edge set of a face**. The edge set $Edge(f)$ of a face f is the set of all the edges of f.
- **Singular point**. The singular points of a curve are defined as the points having the maximal distance to the line passing through the two endpoints of the curve.
- **Concurvity**. Two edges with a common vertex are called **concurved** if they are C^1 continuous at the common vertex. This concurvity is the generalizetion of collinearity.

Many of these terms are illustrated with the line drawings in Fig. 1. Curved edges adc and abc are concurved at vertices a and c. Two artificial lines hl and fj indicate the coplanarity of the cycles (e, f, g, h, e) and (i, j, k, l, i). Edges ae,

[1] Artificial lines have been used in solid modeling to indicate the co-surface of two cycles in a line drawing [2], [8], [22]. Without them, it is impossible to determine the 3D geometric relation between the two cycles (see Fig. 1 for example).

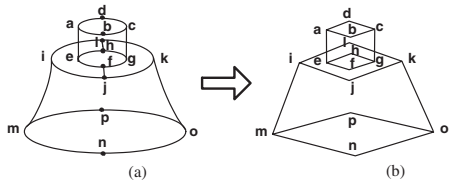

Figure 1. Illustration of some terms.

cg, im, and ko in Fig. 1(a) are four silhouettes of the line drawing. Points d and b are the singular points of curves adc and abc, respectively. Fig. 1(b) shows the line drawing of the generalized polyhedron corresponding to the curved manifold represented by the line drawing in Fig. 1(a). There are four generalized faces in it, such as (a, b, c, g, f, e, a). How to transform a line drawing representing a curved solid into a line drawing representing a polyhedron or generalized polyhedron is discussed in Section 4.2.

4. Reconstruction of Curved Manifolds

This section discusses our main work on the reconstruction of curved manifolds. First, we give some rules to distinguish between curved faces and planar faces. Second, we present the scheme to transform the line drawing of a curved manifold into the line drawing of a generalized polyhedron. Third, we develop new regularities for curved object reconstruction. Fourth, we discuss how to reconstruct the 3D wireframe of the curved object. Finally, we create the curved faces with Bezier surface patches and triangular meshes from the obtained 3D wireframe.

4.1. Distinguishing between curved and planar faces

Before 3D reconstruction, it is helpful to find whether a face is curved or planar. In [11], several labeling rules are proposed for discriminating between curved and planar faces in a line drawing. However, these rules cannot be applied to our problem since [11] deals with line drawings without hidden lines (besides, it does not consider 3D reconstruction). In this section, we propose four rules for the face labeling problem. Before giving the rules, we present several properties.

Property 1 *Two faces that share a straight edge can be either planar or curved.*

Property 2 *At least one of the two faces that share a curved edge is curved.*

Property 3 *Both faces that share a silhouette are curved.*

The above three properties are obvious for manifolds. When two or more disjoint cycles are on the same surface, artificial lines are used to indicate this co-surface property [2], [8], [22]. Artificial lines are easily identified, and when they are removed, these cycles become faces. Note that some such cycles denote holes, but they are still considered as faces in face identification after the artificial lines are removed. Finally, the real visible face can be known from the 2D geometric relation among these co-surface cycles [2], [8], [22]. Here these cycles are still called faces, which have the following property.

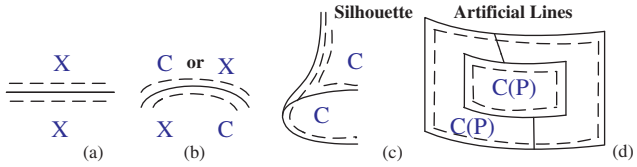

Figure 2. Illustration of Properties 1–4. (a) A straight edge. (b) A curved edge. (c) A silhouette. (d) Two faces connected by two artificial lines.

Property 4 *Two or more co-surface faces indicated by artificial lines are all planar or all curved faces.*

For each face f of a line drawing, we use $L(f) \in \{P, C, X\}$ to denote if f is planar (P), curved (C), or of unknown planarity (X). The four properties are illustrated in Fig. 2. With them, we have the following rules for labeling faces. Let f_1 and f_2 be two faces of a line drawing, and an edge $e \in Edge(f_1) \cap Edge(f_2)$.

Rule 1 *If e is a straight edge and the face f_i, $i = 1, 2$, is unlabeled, then $L(f_i) = X$.*

Rule 2 *If e is a curved edge and $L(f_i) = P$, $i = 1, 2$, then $L(f_{3-i}) = C$.*

Rule 3 *If e is a silhouette, then $L(f_1) = L(f_2) = C$.*

Rule 4 *If f_1 and f_2 are connected by two artificial lines and $L(f_i) = C$ or P, $i = 1, 2$, then $L(f_{3-i}) = L(f_i)$.*

Based on Rules 1–4, we develop Algorithm 1 to find the optimal face labeling configuration. The optimal face labeling is defined as the labeling configuration with the maximum number of planar faces in the interpretation of a line drawing, because planar faces are most common in man-made objects. This criterion is already used in [11]. The algorithm is initialized by assuming a face to be planar. The four rules are then used to deduce the labels of other faces. For any face whose label cannot be decided from any of its edges (labeled by X), we relabel it as planar (P) according to the criterion of maximizing the number of planar faces. The labeling algorithm repeats until all the faces are labeled with either curved (C) or planar (P), and Step 10 can prevent the deadlock of the face labeling. The optimal face labeling configuration is obtained by initializing the algorithm from every face of the line drawing and adopting the labeling configuration with the maximum number of planar faces.

Although we cannot prove that Algorithm 1 can label all line drawings correctly, our experiments show that all the examples we try are successfully labeled. In some cases, multiple solutions occur, such as the one shown in Fig. 3.

Algorithm 1 Identifying planar and curved faces.

Input: A line drawing with its edge set $\mathcal{E}$ and face set $\mathcal{F}$.

1. **for** each face $f \in \mathcal{F}$:
2. **if** f has no silhouettes, **then** set $L(f) = P$; **else goto** 1.
3. **for** each face $f_1 \in \mathcal{F}$, $f_1 \neq f$, set its initial label $L(f_1) = Null$.
4. **for** each edge $e \in \mathcal{E}$, set $visit(e) = 0$.
5. **while** not all the faces are labeled with either C or P **do**
6. Save the previous labeling configuration $LC'(f) = \{L(f_2)|f_2 \in \mathcal{F}\}$.
7. **for** each edge $e \in \mathcal{E}$ in the line drawing, try to label its neighboring faces according to Rules 1–4. If both of its neighboring faces are labeled, set $visit(e) = 1$.
8. **for** each face $f_3 \in \mathcal{F}$, **if** $L(f_3) = X$ and $visit(e) = 1$ for all the edges $e \in Edge(f_3)$, **then** set $L(f_3) = P$.
9. Obtain the current labeling configuration $LC(f) = \{L(f_4)|f_4 \in \mathcal{F}\}$.
10. **if** $LC(f) = LC'(f)$, **then** randomly select a face f_5 from those labeled by X and set $L(f_5) = P$.

Output: The configuration $LC^* \in \{LC(f)|f \in \mathcal{F}\}$ with the maximum number of planar faces.

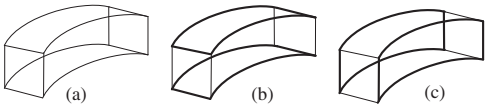

Figure 3. A line drawing (a) with two correct labeling configurations (b) and (c). Only curved faces are marked with bold edges.

4.2. Transformation of Line Drawings

Transforming a line drawing with curved edges into one with straight edges only is an important step for reconstructing the curved object. This transformed line drawing represents a polyhedron or generalized polyhedron. With this line drawing and the original line drawing, we can reconstruct the 3D wireframe of the curved object by combing some previous regularities for planar objects and our new regularities for curved objects.

Our scheme of line drawing transformation is described as follows. When there is only one curve between two vertices, as shown in Fig. 4(a), the curve is straightened (see Figs. 4(b) and (c)). When there is more than one curved edge between two vertices, such as the curves abc and adc between a and c in Fig. 4(d), the curved edges cannot be straightened directly because otherwise the two edges become one. In this case, we first find a singular point in a curve and then replace the curve with two straight lines (Fig. 4(e)). In Fig. 4(c), the line drawing represents a polyhedron, while in Fig. 4(f), the line drawing denotes a generalized polyhedron. In both cases, the face configurations are the same as their corresponding original line drawings, which guarantees that the 3D wireframes of the (generalized) polyhedra can be good approximations of the 3D wireframes of the curved objects.

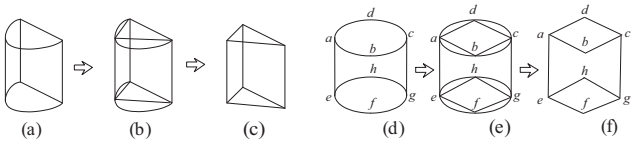

Figure 4. Examples of the transformation of line drawings, where (c) is a polyhedron and (f) is a generalized polyhedron.

4.3. Regularities

Many previous regularities developed for handling planar object reconstruction can be used to reconstruct a polyhedron such as the one in Fig. 4(c). However, a generalized polyhedron with generalized faces does not exist actually (see Fig. 4(f) for example). Our strategy is that based on the transformed line drawing and the original line drawing, use some previous regularities and the regularities proposed in this section to recover the 3D wireframe of the curved object.

Curve Parallelism. Before defining curve parallelism as a new regularity for curved object reconstruction, we first define two terms.

Definition 1 *Given a differentiable curve* $\mathbf{C}(t) = (x_1(t), x_2(t), \cdots, x_n(t))^T$, $t \in [0,1]$, *of* $\mathcal{R}^n$, *the normalized arc-length parametrization of the curve* $\mathbf{C}(t)$ *is defined as* $\mathbf{G}(s) : [0,1] \rightarrow \mathbf{C}(t)$, *where*

$$s = \frac{\int_0^s \|\mathbf{G}'(u)\| \mathrm{d}u}{\int_0^1 \|\mathbf{G}'(u)\| \mathrm{d}u} \tag{1}$$

for $s \in [0,1]$ *and* $\mathbf{G}'(u)$ *is the first derivative of* $\mathbf{G}(u)$.

Note that (1) is the condition that s satisfies; it is not used to determine the arc length s.

Definition 2 *The parallelism between two curves* $\mathbf{C}_1(t)$ *and* $\mathbf{C}_2(t)$ *in* $\mathcal{R}^n$ *is defined as*

$$r_{1,2} = \max\left\{ \int_0^1 \frac{\mathbf{G}_1'(s)^T \mathbf{G}_2'(s)}{\|\mathbf{G}_1'(s)\| \cdot \|\mathbf{G}_2'(s)\|} \mathrm{d}s, \int_0^1 \frac{\mathbf{G}_1'(s)^T \mathbf{G}_2'(1-s)}{\|\mathbf{G}_1'(s)\| \cdot \|\mathbf{G}_2'(1-s)\|} \mathrm{d}s \right\}, \tag{2}$$

where $\mathbf{G}_1(s)$ *and* $\mathbf{G}_2(s)$ *are the normalized arc-length parameterizations of* $\mathbf{C}_1(t)$ *and* $\mathbf{C}_2(t)$, *respectively, and* $\mathbf{G}_1'(s)^T$ *is the transpose of* $\mathbf{G}_1'(s)$.

In the two definitions, n takes 2 or 3 to denote 2D or 3D curves. It is easy to see that $-1 \leq r_{1,2} \leq 1$. The geometric meaning of $r_{1,2}$ is the sum (integration) of the normalized dot products of the tangent vectors at the corresponding points between $\mathbf{C}_1$ and $\mathbf{C}_2$. When $\mathbf{C}_1$ and $\mathbf{C}_2$ are parallel perfectly, such as $\mathbf{C}_2$ being a copy of $\mathbf{C}_1$ shifted to another position, $r_{1,2} = 1$. Two unparallel curves result in a small $r_{1,2}$. Let $r_{1,2}^{2D}$ and $r_{1,2}^{3D}$ be the parallelism values between two curves in

the 2D sketch plane and 3D space, respectively. Then the term used to enforce the constraint of curve parallelism is

$$\alpha_{CP} = \sum_{i,j} w_{i,j}^{CP}(1 - r_{i,j}^{3D})^2, \tag{3}$$

where the weighting factor

$$w_{i,j}^{CP} = \frac{1}{1 + exp\left[-\sigma_1(r_{i,j}^{2D} - 0.8)\right]}, \tag{4}$$

and σ_1 is a parameter to control the effect of $w_{i,j}^{CP}$. When α_{CP} is minimized, the regularity requires that two parallel curves in the 2D line drawing are also parallel in 3D space. In our experiments, we choose $\sigma_1 = 100$.

Generalized Face Perpendicularity. Face perpendicularity is first introduced as a regularity to inflate a flat line drawing into a 3D shape in [19]. It requires adjacent faces to be perpendicular. In this work, we generalize it for generalized polyhedra. Using the vertices of each curved face, we find a best-fitting plane and enforce it to be perpendicular to its adjacent planar faces. For example, in Fig. 4(f), the best-fitting plane obtained from the four vertices of the curved face (a, e, f, g, c, b, a) is required to be perpendicular to the faces (a, d, c, b, a) and (e, h, g, f, e). The following term is used to enforce this regularity:

$$\alpha_{GFP} = -\sum_{i=1}^{K}[\sin^{-1}(\mathbf{n}_{i1} \cdot \mathbf{n}_{i2})]^2, \tag{5}$$

where $\mathbf{n}_{i1}$ and $\mathbf{n}_{i2}$ denote all the possible combinations of the unit normals of the best-fitting planes and their corresponding adjacent planar faces, and K is the number of the combinations.

Curve Concurvity. Curve concurvity is a generalization of the regularity of line collinearity in [19]. This regularity requires that two concurved edges in the 2D sketch plane are also concurved in 3D space. According to the definition in Section 3, curve concurvity means that two edges have the same tangents at their common end. In our work, we use Bezier curves to represent all the curved edges in a line drawing. Checking the concurvity between two curves is reduced to verifying if the two corresponding control points and the common vertex are collinear. Fig. 5 shows two concurved edges e_1 and e_2. If both e_1 and e_2 are curved, then the control points p_{12}, p_{21}, and the common vertex v are collinear; if e_1 is curved and e_2 is straight, then the control point p_{12} and the vertices v and v_2 are collinear. We hence use the following term to describe this regularity:

$$\alpha_{CC} = \sum_{i=1}^{N} \sum_{\substack{j,k \in \mathcal{E}(i) \\ j \neq k}} w_{ijk}^{CC} \left(\frac{\|(\mathbf{P}_j - \mathbf{P}_i) \times (\mathbf{P}_k - \mathbf{P}_i)\|}{\|\mathbf{P}_j - \mathbf{P}_i\| \cdot \|\mathbf{P}_k - \mathbf{P}_i\|} \right)^2, \tag{6}$$

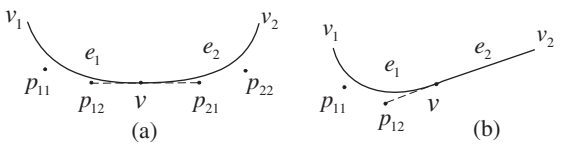

Figure 5. Illustration of curve concurvity.

where N is the number of vertices of the line drawing; $\mathcal{E}(i)$ is the set of all the edges ending at vertex i; $\mathbf{P}_i$ is the 3D point of vertex i; $\mathbf{P}_j$ ($\mathbf{P}_k$) is the 3D point of the other end of the edge j (k) if edge j (k) is a straight line, or the first control point neighboring to vertex i of edge j (k) if edge j (k) is a curve; the weighting factor

$$w_{ijk}^{CC} = \frac{1}{1 + exp\left[-\sigma_2(\beta_{ijk} - 8\pi/9)\right]}, \tag{7}$$

where β_{ijk} is the angle between two vectors $(\overline{\mathbf{P}}_j - \overline{\mathbf{P}}_i)$ and $(\overline{\mathbf{P}}_k - \overline{\mathbf{P}}_i)$ with $\overline{\mathbf{P}}_i$, $\overline{\mathbf{P}}_j$, and $\overline{\mathbf{P}}_k$ being the available 2D projections of $\mathbf{P}_i$, $\mathbf{P}_j$, and $\mathbf{P}_k$, respectively, and σ_2 is a parameter to control the effect of w_{ijk}^{CC}. When $\overline{\mathbf{P}}_i$, $\overline{\mathbf{P}}_j$, and $\overline{\mathbf{P}}_k$ are nearly collinear, w_{ijk}^{CC} is larger and close to 1; otherwise w_{ijk}^{CC} is close to 0. In our experiments, we choose $\sigma_2 = 100$.

4.4. 3D Wireframe Reconstruction

We use both the original line drawing and the transformed line drawing of the (generalized) polyhedron to reconstruct the 3D wireframe of the curved object. Several regularities used in previous work for planar object reconstruction are applied to the transformed line drawing, which are minimizing the standard deviation of angles in the reconstructed object (α_1) [25], face planarity (α_2) [15] (effective on the planar faces but not the generalized faces), line parallelism (α_3) [19], and corner orthogonality (α_4) [19]. The three new regularities (denoted as α_5, α_6, and α_7) proposed in Section 4.3 and the regularity isometry (α_8) [19] are applied to the original line drawing. Isometry can be used for both straight and curved edges. It requires that the lengths of the edges in the 3D wireframe are uniformly proportional to their lengths in the line drawing. It is used to avoid that a reconstructed object becomes too distorted but still projects to the same line drawing.

During the 3D reconstruction, previous methods for a planar object only need to use the z coordinates (depths) of the vertices to compute the regularity terms (the x and y coordinates of the vertices are available). In our work, however, not only the depths but also the 3D curves are required to compute all the regularity terms. Next, we discuss how to determine the curves during the reconstruction.

Let a 3D curve described in the parameterized form be $\mathbf{C}(t) = (x(t), y(t), z(t))^T$. Under the parallel projection, the 2D image of $\mathbf{C}(t)$ is simply $\overline{\mathbf{C}}(t) = (x(t), y(t))^T$. Now we need to recover $z(t)$ from $\overline{\mathbf{C}}(t)$. In general, given $\overline{\mathbf{C}}(t)$ and several specific 3D points on $\mathbf{C}(t)$, it is impossible to determine other 3D points on it. However, in our case where the

3D curve is planar, determining it becomes possible.

Given the 2D curve $\overline{\mathbf{C}}(t) = (x(t), y(t))^T$ and the coordinate of its one endpoint $\mathbf{P}_0 = (x_0, y_0, z_0)^T$ in 3D space, if we can find the unit normal vector $\mathbf{n} = (n_x, n_y, n_z)^T$ of the plane the 3D curve $\mathbf{C}(t) = (x(t), y(t), z(t))^T$ lies on, we can recover the depth of the curve by

$$z(t) = z_0 - (n_x(x(t) - x_0) + n_y(y(t) - y_0))/n_z. \quad (8)$$

Note that $n_z \neq 0$; otherwise the 3D curve is projected onto a straight line, which contradicts the assumption that the line drawing is in a generic view. In this paper, every curved edge in a line drawing is planar. It can be formed either by the intersection of a curved face and a planar face or by the intersection of two curved faces. In the former case, the normal of the plane on which the 3D curve lies can be determined directly since the 3D planar face is available during the reconstruction of the 3D wireframe.

When a curved edge is formed by two curved faces, there are an infinite number of planes passing through the two endpoints of the curve. In this case, we find suitable planes such curved edges lie on by minimizing the regularities where these 3D curves are involved. Therefore, in addition to the depths of all the vertices, the unit normal vectors of these planes are used as part of the variables of the objective function defined as follows:

$$F(z_1, z_2, \cdots, z_N, \mathbf{n}_1, \mathbf{n}_2, \cdots, \mathbf{n}_M) = \sum_{i=1}^{8} \lambda_i \alpha_i. \quad (9)$$

where λ_{1-8} are eight weighting factors determined by experiments, α_{1-8} are the eight regularity terms, z_{1-N} are the depths of all the N vertices of the line drawing, and $\mathbf{n}_{1-M}$ are the unit normal vectors of the M planes on which each of the M curved edges is the intersection of two curved faces. We use the hill-climbing method presented in [15] to minimize F. Note that in (9), each unit normal vector $\mathbf{n}_i$ has only one independent variable due to the two relations $\|\mathbf{n}_i\|^2 = 1$ and $\mathbf{n}_i^T(\mathbf{P}_{i1} - \mathbf{P}_{i2}) = 0$ where $\mathbf{P}_{i1}$ and $\mathbf{P}_{i2}$ are the two vertices of the ith curve and are available during the optimization (reconstruction) process.

4.5. Generating Curved Faces

After obtaining the 3D wireframe of the curved object, we fill in the cycles denoting the curved faces with smooth surface patches. A Bezier patch is generated for a curved face with three or four edges and a triangle mesh is used to create a curved face with more than four edges.

Bezier and Coons patches [12] are suitable for patches with four boundaries (patches with three boundaries are a special case). A curved patch parameterized by $\mathbf{S}(u, v)$, $0 \leq u, v \leq 1$, can be approximated by its four boundary curves $\mathbf{S}(0, v)$, $\mathbf{S}(1, v)$, $\mathbf{S}(u, 0)$, and $\mathbf{S}(u, 1)$ as follows:

$$\begin{aligned}\mathbf{S}^*(u, v) = {} & (1-u)\mathbf{S}(0, v) + u\mathbf{S}(1, v) + (1-v)\mathbf{S}(u, 0) \\ & + v\mathbf{S}(u, 1) - (1-u)v\mathbf{S}(0, 1) - u(1-v)\mathbf{S}(1, 0) \\ & - (1-u)(1-v)\mathbf{S}(0, 0) - uv\mathbf{S}(1, 1). \end{aligned} \quad (10)$$

$\mathbf{S}^*(u, v)$, called the Coons patch, passes through the four boundaries and the four corners exactly once. Note that this kind of surface patches gives not only an easy way of interpolation from the boundaries, but also good results in accordance with human visual observation.

Since the boundary curves are represented by Bezier curves in our work, motivated by the Coons patch, we use a similar manner to interpolate all the control points $\mathbf{P}_{i,j}$ of the Bezier patch $\mathbf{S}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{P}_{i,j} B_{i,n}(u) B_{j,m}(v)$ based on the known control points, $\mathbf{P}_{0,j}$, $\mathbf{P}_{n,j}$, $\mathbf{P}_{i,0}$, and $\mathbf{P}_{i,m}$, of the boundaries where $0 \leq i \leq n, 0 \leq j \leq m$. The interpolation equation is written as

$$\begin{aligned}\mathbf{P}_{i,j} = {} & (1 - \frac{i}{n})\mathbf{P}_{0,j} + \frac{i}{n}\mathbf{P}_{n,j} + (1 - \frac{j}{m})\mathbf{P}_{i,0} + \frac{j}{m}\mathbf{P}_{i,m} \\ & - (1 - \frac{i}{n})(1 - \frac{j}{m})\mathbf{P}_{0,0} - (1 - \frac{i}{n})\frac{j}{m}\mathbf{P}_{0,m} \\ & - \frac{i}{n}(1 - \frac{j}{m})\mathbf{P}_{n,0} - \frac{i}{n}\frac{j}{m}\mathbf{P}_{n,m}. \end{aligned} \quad (11)$$

It is easy to verify that the control points chosen in such a way generate a Bezier patch equivalent to the Coons patch.

When a curved face is bounded by more than four edges, we use a triangle mesh to generate it. The mesh generation is formulated as a quadratic optimization problem. An initial isotropic mesh is first built from the boundary edges of each curved face, and then the mesh is refined by minimizing the following objective function:

$$\begin{aligned} Q(\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_K) = {} & \lambda \sum_i \sum_{j \in \mathcal{N}(i)} \|\mathbf{u}_i - \mathbf{u}_j\|^2 \\ & + \gamma \sum_{i \notin \mathcal{S}} \sum_{j \in \mathcal{N}(i) \setminus \mathcal{S}} \|\mathbf{c}_i - \mathbf{c}_j\|^2 + \sum_{i \in \mathcal{S}} \|\mathbf{u}_i - \mathbf{u}'_i\|^2, \end{aligned} \quad (12)$$

where $\mathbf{u'}_i$, $\mathbf{u}_i$, and $\mathbf{c}_i$, $i = 1, 2, \cdots, K$, are the initial positions, the new positions, and the curvatures of all K mesh points, respectively; $\mathcal{N}(i)$ is the set of mesh points connected to the ith point in the mesh; $\mathcal{S}$ is the set of mesh points located on the 3D wireframe; λ and γ are weighting factors. On the right hand side of (12), the first term enforces the smoothness on the mesh, the second term is used to maintain the continuity of the curvature in the mesh which is approximated by the discrete graph Laplacian $\mathbf{c}_i = \mathbf{u}_i - \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{u}_j$, $i \notin \mathcal{S}$ [34], and the last term is the fitting constraint that requires the mesh to fit the points on the wireframe well. By minimizing Q, we find the positions of all the mesh points. This optimization problem has a closed-form solution.

4.6. The Complete 3D Reconstruction Algorithm

The outline of the complete 3D reconstruction algorithm is summarized in Algorithm 2.

When a line drawing represents a complex object, we use the divide-and-conquer technique in [8] to separate it into simpler line drawings and transform them into (generalized) polyhedra. After reconstructing the 3D shapes from these line drawings, we merge them into a complete object.

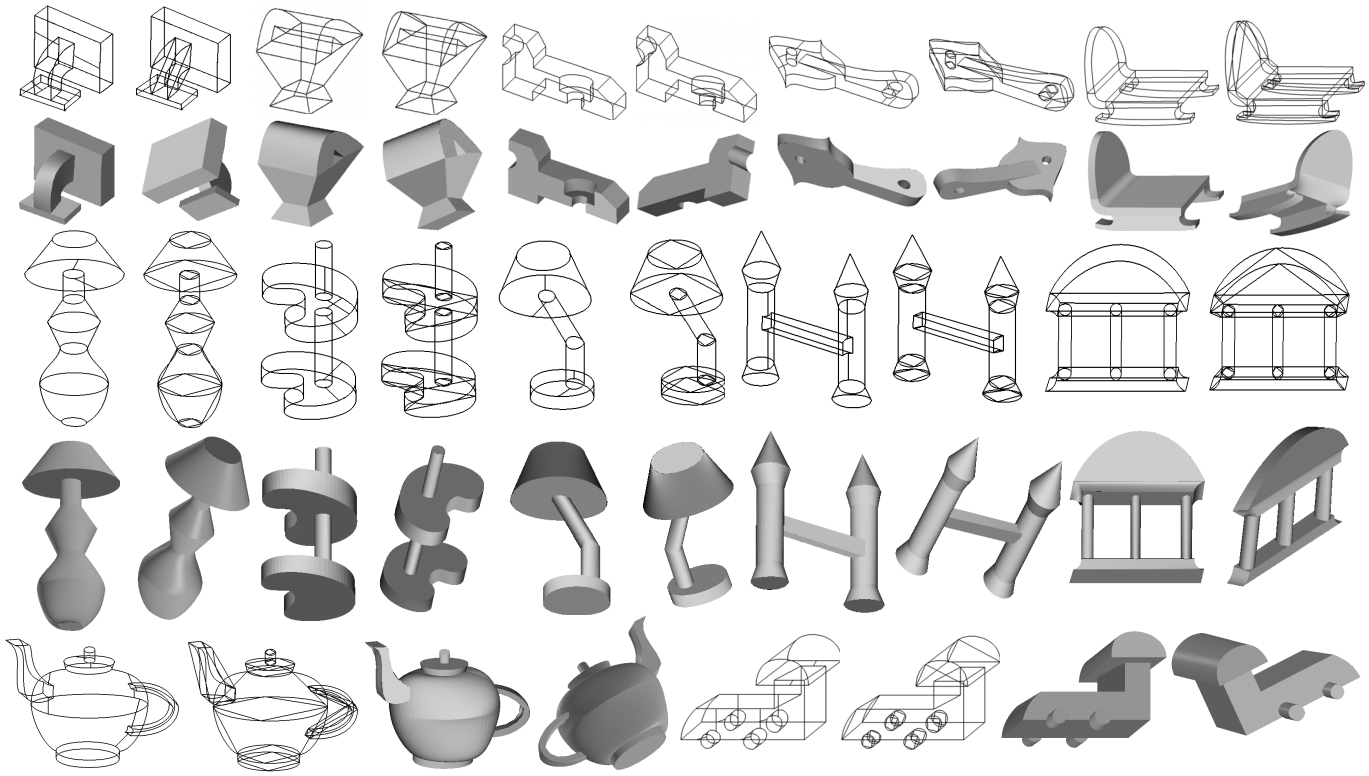

Figure 6. A set of line drawings, their transformed line drawings, and their reconstructed results, each shown from two viewpoints.

Algorithm 2 3D curved object reconstruction.

1. Distinguish between the planar and curved faces in the line drawing by Algorithm 1.
2. Transform the line drawing into one representing a (generalized) polyhedron.
3. Reconstruct the 3D wireframe of the curved object.
4. Generate the surface patches of the curved faces.

5. Experimental Results

A number of manifold objects with both planar and curved faces have been drawn to test our approach. The algorithm is implemented in Visual C++, running on a 3.2 GHz Pentium IV PC. The weighting factors α_{1-8} are chosen to be 100, 1, 80, 20, 80, 80, 80, and 15, respectively. The parameters λ and γ in (12) are set to 0.05 and 0.2. These parameters are obtained from a few tests first and then fixed in the reconstruction of all the objects.

Fig. 6 shows a set of line drawings and their reconstruction results. For each line drawing, we also show the transformed line drawing superimposed on its original one. Each 3D reconstruction result is displayed in two views. From Fig. 6, we can see that the results accord with our visual perception very well. Note that some results may not look so perfect, such as the teapot. It is part of our future work to fine-tune the results. In addition, our algorithm still cannot handle line drawings representing free-form objects such as a human body.

The computational time of our algorithm varies with different line drawings depending on the complexity of them. For those in Fig. 6, it ranges from 8.0 seconds to 127 seconds. Among the four steps given in Section 4.6, Steps 3 and 4 take almost all the computational time.

6. Conclusions

In this paper, we have proposed a novel approach to 3D curved manifold object reconstruction from single 2D line drawings. It includes four steps: (1) discriminating between curved faces and planar faces in the line drawing, (2) transforming the line drawing of a curved manifold into the line drawing of a (generalized) polyhedron, (3) reconstructing the 3D wireframe of the curved object based on the transformed line drawing and the original line drawing, and (4) generating the curved faces with Bezier patches and triangular meshes. A number of examples given in the experimental section clearly indicate the ability of our approach to perform 3D curved object reconstruction. The previous methods of curved object reconstruction from line drawings [5], [7], [16], [26] need human interaction and can only handle simple objects. In contrast, our approach can reconstruct complex curved objects automatically. Our future work includes fine-tuning the results and developing more regularities for curved object reconstruction.

Acknowledgements

This work was supported by two grants from the Research Grants Council of the Hong Kong SAR, China (Project No. CUHK 414306 and 415408).

References

[1] S. Ablameyko, V. Bereishik, A. Gorelik, and S. Medvedev. 3D object reconstruction from engineering drawing projections. *Computing & Control Engineering Journal*, 10(6):277–284, 1999.

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

[3] S. Bagali and J. Waggenspack. A shortest path approach to wireframe to solid model conversion. *Proc. 3rd Symp. Solid Modeling and Applications*, pages 339–349, 1995.

[4] H. Barrow and J. Tenenbaum. Interpreting line drawings as three-dimensional surfaces. *Artificial Intelligence*, 17:75–116, 1981.

[5] 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*, pages 105–108, 2006.

[6] 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.

[7] X. Chen, S. Kang, Y. Xu, J. Dorsey, and H. Shum. Sketching reality: Realistic interpretation of architectural designs. *ACM Trans. Graph.*, 27(2):1–15, 2008.

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

[9] M. Cooper. The interpretations of line drawings with contrast failure and shadows. *IJCV*, 43(2):75–97, 2001.

[10] M. Cooper. Wireframe projections: Physicl realisability of curved objects and unambiguous reconstruction of simple polyhedra. *IJCV*, 64(1):69–88, 2005.

[11] M. Cooper. A rich discrete labeling scheme for line drawings of curved objects. *IEEE Trans. PAMI*, 30(4):741–745, 2008.

[12] A. Davies and P. Samuels. An introduction to computational geometry for curves and surfaces. *New York: Oxford University Press Inc.*, 1996.

[13] P. Debevec, C. Yaylor, and J. Malik. Modeling and rendering architecture from photographs: a hybrid geometry and image-based approach. *SIGGRAPH 96 Conf. Proc.*, pages 11–20, 1996.

[14] J. Hojnicki and P. White. Converting CAD wireframe data to surfaced representations. *Computer in Mechanical Engineering*, pages 19–25, 1988.

[15] 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.

[16] S. Lee, D. Feng, and B. Gooch. Automatic construction of 3d models from architectural line drawings. *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 123–130, 2008.

[17] R. Lequette. Automatic construction of curvilinear solid from wireframe views. *Computer-Aided Design*, 20(4):171–180, 1988.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

[24] J. Malik. Interpreting line drawings of curved objects. *IJCV*, 1(1):73–103, 1987.

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

[26] M. Masry, D. Kang, and H. Lipson. A freehand sketching interface for progressive construction of 3d objects. *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, page 30, 2007.

[27] S. Ortiz. 3D searching starts to take shape. *Computers*, 37(8):24–26, 2004.

[28] A. Piquer, R. Martin, and P. Company. Using skewed mirror symmetry for optimisation-based 3D line-drawing recognition. *Proc. 5th IAPR Int. Workshop on Graphics Recognition*, pages 182–193, 2003.

[29] L. Ros and F. Thomas. Overcoming superstrictness in line drawing interpretation. *IEEE Trans. PAMI*, 24(4):456–466, 2002.

[30] K. Shoji, K. Kato, and F. Toyama. 3-D interpretation of single line drawings based on entropy minimization principle. *CVPR*, 2:90–95, 2001.

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

[32] K. Sugihara. Mathematical structures of line drawings of polyhedrons—toward man-machine communication by means of line drawings. *IEEE Trans. PAMI*, 4(5):458–469, 1982.

[33] K. Sugihara. A necessary and sufficient condition for a picture to represent a polyhedral scene. *IEEE Trans. PAMI*, 6(5):578–586, 1984.

[34] G. Taubin. A signal processing approach to fair surface design. *Proc. SIGGRAPH*, 7:351–358, 1995.

[35] A. Turner, D. Chapman, and A. Penn. Sketching space. *Computers & Graphics*, 24:869–879, 2000.

[36] A. Vicent, P. Calleja, and R. Martin. Skewed mirror symmetry in the 3D reconstruction of polyhedral models. *Journal of WSCG*, 11(3):504–511, 2003.

[37] M. A. Wesley and G. Markowsky. Fleshing out projections. *IBM Journal of Research And Development*, 25(6):934–954, 1981.