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

WANG, Yingze

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Information Engineering

July 2009

Abstract

3D object reconstruction from 2D line drawings is an important problem in both computer vision and graphics. It has a wide range of applications including flexible sketching input for designers who prefer to use pencil and paper, providing rich databases to object recognition systems, friendly user interface for 3D object retrieval, and interactive generation of 3D models from images.

A single 2D line drawing is defined as the 2D projection of the edges and vertices of a 3D object in a generic view. Many methods have been proposed to solve 3D reconstruction from single 2D line drawings. However, they can only handle the line drawings of planar objects. The reconstruction of curved objects is still a challenging problem. Few works have been developed to deal with this problem and they usually carry out the curved object reconstruction based on human interactions.

In this thesis, a novel approach is proposed to reconstruct solid objects that have not only planar but also curved faces. Our approach consists of four steps: (1) identifying the curved faces and the planar faces in a line drawing; (2) transforming the line drawing into a new one with straight edges only; (3) reconstructing a 3D wireframe of the curved object from the transformed line drawing and the original line drawing; (4) generating the curved faces with Bezier patches and triangular meshes. There is no strict restriction on the objects in our algorithm and we construct complex curved objects automatically. A number of experimental results are given to demonstrate the excellent performance of our approach on 3D curved object reconstruction.

摘要

在機器視覺和圖形學領域，一個重要的方向是研究一種算法，它能夠從物體輪廓的二維線畫圖重建物體的三維結構。該研究具有廣泛的應用，包括方便喜歡用鉛筆和紙繪圖的用戶自由設計草圖來描繪物體，爲物體識別系統提供豐富強大的數據庫，提供三維物體檢索的友好界面，方便用戶從單幅圖片交互的重建三維物體。

二維線畫圖是指三維物體從一般角度投影所產生的二維頂點和邊的線畫圖。目前基於二維線畫圖的三維物體重建方面有很多的研究。但是，他們都只能重建平面組成的物體。由曲面和平面組成的比較複雜的物體的重建仍然是一個非常具有挑戰性的問題。這方面的研究非常少，其中的方法通常都需要人工交互。

本論文提出了一個全新的方法，它可以重建由平面和曲面組成的複雜物體。該方法包含四個步驟：（１）辨別二維線畫圖中表示的曲面和平面；（２）將線畫圖轉化成爲僅含有直線的線畫圖；（３）通過轉化後的線畫圖和原線畫圖重建曲面物體的三維輪廓；（４）利用貝塞爾曲面和三角網格產生曲面。我們的算法不需要對物體有嚴格限制，可以自動重建複雜的曲面物體。大量的實驗結果證明該算法在三維曲面物體的重建方面具有良好的性能。

Acknowledgement

First of all, I would like to give my extremely gratitude to my supervisor Prof. Jianzhuang Liu, who has been instructing and encouraging me during the past two years. He has always been a nice and enthusiastic advisor, providing great guidance and valuable panoramic perspective, discussing with students to teach useful things to us. He is always there to meet and talk about my ideas, to proofread and mark up my writings and to ask me good questions to help me think through my problems. He gives me great help not only in research but also in my future career path.

I am also grateful to my co-supervisor Prof. Xiaoou Tang. His deep insight and significant suggestions give me great help in the way of research as well as my future plan. He hardly criticizes students, instead, encourages us with special and effective ways to direct us to think actively and independently in research.

A special thank goes to Chen Yu, who was my labmate and the second author of our accepted paper. He helped me much to complete this work as well as the challenge research lying behind it. For many tough problems, I always discussed with him and he provided many valuable comments and suggestions. Without his encouragement and constant help, I would not have finished this thesis.

Moreover, I want to show my sincerely deep appreciation to many guys in my lab. Chunjing, a good and so nice guy sitting beside me, is always there answering my many questions with no weariness. He has been an old brother, showing me ways to approach a research problem and encouraging me to be persistent to accomplish a goal. Weige, a really good guy who is a genius in C++ programming, is always generous to offer me help in computer programming. I extent my warm thanks to A Feng, Liu Ming, Boqing, Zhimin, Zhengou, Yiwen, Yueming, Deli, Zhangwei, Duhao, Huanzi, Huangting and Chenmo. I am proud and lucky for spending these happy two years with all of you in such a good lab.

Finally, I would like to thank my parents. They always encourage me and give me confidence to make me never give up. The last one that I want to show my deep appreciation is my boyfriend. Because of him, I never lose hope and insist on doing research in the future. I cannot imagine that how to get through the hard days during the two years without him. I love him, thanks

too much!

I really cherish the good time with all of you!

Contents

List of Figures

Chapter 1

Introduction

Three dimensional objects shown on images and the screen lose their depth information. Recovering of this lost information and reconstruction of 3D objects have always been of great concern and is a problem faced by different researchers. Nowadays, 3D reconstruction is used in a wide variety of applications. For examples, the architecture industry uses it to design 3D buildings and shows desired configuration and landscape through particular software. Researchers and designers reconstruct 3D objects in computer game, virtual reality, internet web pages, etc. Moreover, 3D reconstruction of blood vessels or other organs has been widely used in medical imaging field to facilitate doctors' diagnosing. Consequently, many researchers in computer vision and graphics have made their efforts in designing algorithms and new methods for 3D reconstruction.

There exists many different aspects of research in 3D reconstruction, including 3D reconstruction from video sequences, 3D reconstruction from single view or multiple view images, 3D reconstruction from engineering drawings of multiple views, etc. This thesis focuses on understanding a single 2D line drawing representing an object with planar and curved faces and reconstructing the 3D geometry of the object. This area belongs to the field *shape from line drawings* or *shape from contours*.

Single 2D line drawings provide a straightforward and 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. It would be very helpful if there is an efficient algorithm to make such a drawing used in computer-aided design (CAD), where tools that can directly convert a design sketch into a 3D model are extremely desirable. The reason for this is that creating 3D objects with current sophisticated CAD tools is still a tedious work, while a 2D line drawing is simple to draw and is the most direct way of demonstrating a 3D object. Current CAD tools cannot convert a line drawing into a 3D object, which prevents designers, especially conceptual designers from inputting or designing conveniently. Therefore, developing approaches that can understand single 2D line drawings and reconstruct the 3D models from the

sketches become more and more important and useful. A number of publications have been devoted to this research in the major computer vision literature [10], [29], [33], [35], [36], [38], [37], [34], [42], [43], [48], [51], [52], [54], [56], and in CAD and graphics [2], [3], [13], [14], [62], [63], [59]. This research has many applications including:

- flexible sketching input for conceptual designers who tend to prefer pencil and paper to mouse and keyboard in current CAD systems [3], [33], [36], [49];
- providing rich databases to object recognition systems and reverse engineering algorithms for shape reasoning [2], [3], [14], [57];
- automatic conversion of industrial wireframe models to solid models [2], [24];
- interactive generation of 3D models from images [20], [34], [59], [53];
- friendly user interface for 3D object retrieval [7], [45] [68].

Here a 2D line drawing is defined as the parallel or nearly-parallel projection of a 3D 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 3D object, including its back, from the line drawing. Fig. 1.1 shows an example of 3D reconstruction from line drawings. Such line drawings may be inputted from sketch images on paper, a digitizer tablet, and the screen drawn with a pen or a mouse, generated by the user (designer) or came from existing industrial wireframe models. To extract an edge-vertex graph in a scanned image, some procedures are required such as binarizing and thinning of the image, tracking and analyzing of the lines, curves and vertices. 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], [10], [29], [33], [34], [42], [47], [48], [51], [54], [56], [59], [65]. However, they handle only line drawings of planar objects, and most of them consider simple objects of genus 0 (without holes). Although there are few works which deal with curved objects [7], [9], [30], [44], these methods cannot reconstruct curved objects automatically and fail in dealing with complex objects. As a consequence, reconstruction of curved objects is still a challenging problem. Three non-collinear points determine a plane, but a curved surface often has much more degrees of freedom. Therefore, the reconstruction of curved objects owns a higher underconstrained nature.

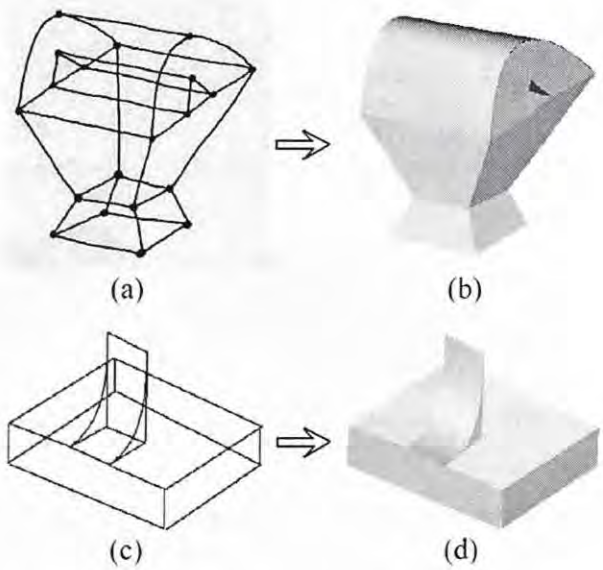


Figure 1.1: Two line drawings of solids and their reconstructed shapes. (a) A 2D line drawing obtained by scanning the sketch image on paper. (b) Reconstructed 3D object of (a). (c) A 2D line drawing coming from the screen drawn with a mouse. (d) Reconstructed 3D object of (c).

In this thesis, we propose an approach to the 3D reconstruction from line drawings of solids with not only planar but also curved faces. Previous methods for line drawing reconstruction usually fail for curved objects, so the main contribution of the thesis is that the algorithm can handle complex curved objects automatically. To the best of our knowledge, our approach can tackle objects with the most complex geometrical structures in this area. 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. Fig. 1.2 illustrates the process of our approach. It should be emphasized that although there are infinite possible 3D wireframes that can be projected to the same 2D line drawing, human observers trend to have quite definite 3D perception from the line drawing. Obviously, in this research, there is no a precise (but a rough) 3D object corresponding to a sketch. Whether or not a reconstruction is successful is judged by human observers.

We use those line drawings from previous papers and also draw more complex ones to verify the effectiveness of our new approach. The experiments

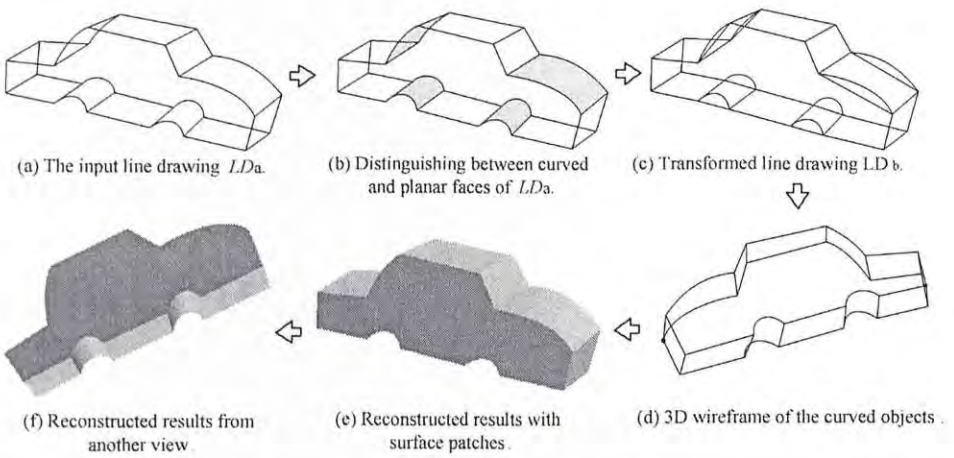


Figure 1.2: Illustration of our approach to the problem of curved objects reconstruction from a single line drawing.

show that our approach can tackle 3D reconstruction from much more complex line drawings with both planar and curved faces. This work has been accepted by IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2009.

Chapter 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 wire-frame 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. Methods in the second group try to reconstruct a 3D CAD model from its multiple orthographic projections. 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. In this section, we give a brief review on these related works.

2.1 Line labeling and realization problem

Since the early stage of computer vision, a large amount of works are about line labeling and 3D reconstruction based on a labeled drawing [12], [15], [16], [17], [18], [23], [25], [26], [41], [54], [56], [55], [66], [61]. Line labeling is meant to find a set of consistent labels from a line drawing and to provide a qualitative description of the scene by classifying the segments of a line drawing as the projection of concave, convex or contour edges. However it does not explicitly give the 3D structure represented by a line drawing. Early work on line labeling focuses on labeling polyhedra without hidden lines. Huffman et al. [25] first described a scheme for labeling line drawings, and independently by Clowes [12], in 1971. Huffman restricted his attention to the case where all faces are planar, that is, a "polyhedral world", so now there are only four possible labels $\{+, -, \vee, \wedge\}$. He also assumed that all vertices are trihedral, that is, they are formed by exactly three faces, and that there are no object alignments, which would result in a "crack" edge. In 1971, Waltz [66] exhibited a filtering algorithm with very good average running time (roughly linear in the number of segments). The algorithm achieved local consistency in the following way: given a junction, rule out all legal labelings of the junction

for which there is no labeling of the neighbor junctions which is compatible with it. Then, repeat this procedure until no further progress can be made. To label a line drawing, the algorithm need first achieve this local consistency and then achieve global consistency by tree searching with backtracking. Most of the line drawings labeling algorithms are generally suitable for polyhedra without hidden edges. Recently, Cooper extended this line labeling research to wireframes with hidden lines visible as well as curved objects [16], [17], [18]. However, the limitations of line labeling are that multiple consistent labeling solutions for one line drawing are possible [50].

Another area of the work is Realization, which involves the physical legitimacy of the interpreted scene for line labelings, and tries to recover the underlying 3D structure based on algebra test with linear equalities and inequalities [54], [56], [55], [60], [48]. Using line labeling schemes, all physical objects should be labelable. However, labelability is not a sufficient condition for physical realizability. Because there are always vertex position errors in the practical line drawing, which is either extracted from an image or drawn by a person, so it is often impossible to find a 3D object whose projection is exactly the line drawing. Small deviations of some vertices from their precisely projected 2D positions may cause the 3D vertices on the same planar face to be noncoplanar. A typical example is illustrated in Fig. 2.1. In Fig. 2.1 (b), it may seem to represent a truncated pyramid, but it does not, for the reason that three side edges (when extended) should meet at one common point for the real truncated while they do not as shown is Fig. 2.1 (c). Therefore, Fig. 2.1 (c) is not the projection of a truncated pyramid and Fig. 2.1 (b) does not represent a polyhedron correctly. However, human beings can understand with no difficulty what the line drawing represents and visualize this object as a real truncated pyramid. This problem is termed superstrictness. Consequently, the limitations of these methods [54], [56], [55], [60] are that such a formulation is superstrict and not robust; realizability can be efficiently checked only when a legal labeling is available.

2.2 3D reconstruction from multiple views

Papers in this group try to reconstruct a 3D CAD model from its multiple (three, in general) orthographic projections [1], [67], [27], [39], [11], [21], [28], [31], [40]. More information can be found from three orthographic views for the reconstruction task than from a single projected view, so this work is more easier compared to 3D reconstruction from a single line drawing. Traditionally, engineering objects are represented by three orthographic views: front, top and side views, which are the case for most of the engineering drawings. Liu et al. [39] used matrices to represent conic faces for the reconstruction of different objects such as planar, cylindrical and conical faces. He also gave the

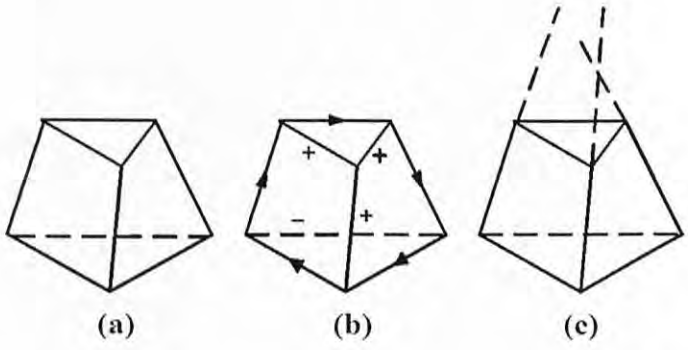


Figure 2.1: (a) A line drawing representing truncated pyramid. (b) A consistent labeling [54]. (c) The wrong projection of truncated pyramid.

proof that minimum number of views required to represent conics are three. The approach in [11] was based on constructive solid geometry and required three orthographic views. The technique is powerful in handling blind pockets, through pockets, circular pockets, through holes, blind holes, counter bored through holes, counter bored blind holes, etc. Dimri et al. [21] introduced a novel technique of reconstruction from x-sectional views. Handling of sectional views was also discussed by Wesley [67] but their approach is limited to full sectional views only. Technique of Dimri [21] took into account full sectional, half sectional, offset sectional. Because our work does not belong to 3D reconstruction from multiple views in this thesis, we won't give much detailed review of this group of research.

2.3 3D reconstruction from single line drawings

Research on 3D reconstruction from single line drawings, which our work belongs to, can be divided into two subproblems: face identification from the line drawings and 3D geometry reconstruction. Next, the methods in these two aspects will be reviewed in more details.

2.3.1 Face identification from the line drawings

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. Fig. 2.2 shows the faces of the two line drawings in Figs. 1.1(a) and (c).

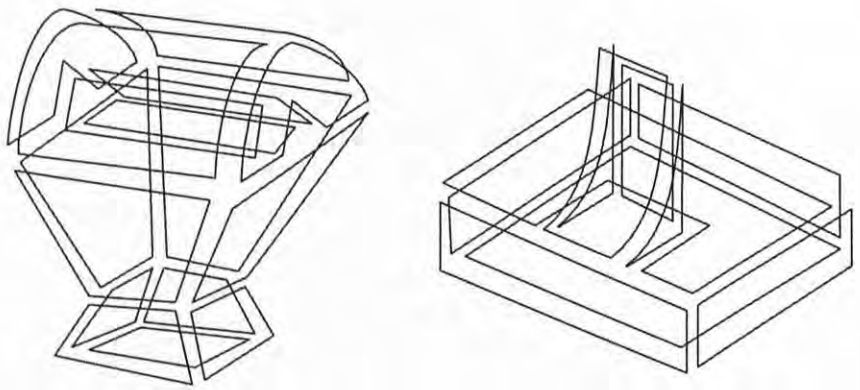


Figure 2.2: The faces of the two line drawings in Fig. 1.1.

In general, there are many cycles in a line drawing and only a small subset of them represent its faces, and the number of cycles grows exponentially with the number of edges. Thus finding the faces from a line drawing is not a trivial problem. Much effort has been made in this area [36], [3], [2], [29], [52], [38], [35], [37], [32].

A distinct decomposition method for extracting face topologies from wire-frame models was proposed by Agarwal and Waggenspack's method [2]. They employed a divide-and-conquer strategy to remove stars (tetrahedra, N-sided pyramids, or multiply connected stars) from a drawing. The faces of the drawing were obtained by combining triangles that were created from the stars. However this method failed in some occasions mentioned in [36]. Bagali and Waggenspack's approach [3] was based on an efficient shortest path algorithm for cycle generation. Their algorithm is fast, conceptually simpler and easy to implement, but limited to 3-connected drawings of genus 0. The recent work presented in [33] and [35] can handle a larger range of objects than previous methods. Both of them included two steps: finding a set of circuits that may be potential faces and searching for faces from this set. It needs to be emphasized that the two steps in each of the two methods correspond to two combinatorial problems. The number of circuits is generally exponential in the number of edges of a line drawing. Shpitalni and Lipson [33] presented two algorithms for the face identification problem. Their first algorithm was using the planar embedding algorithm to locate faces of a drawing. Although they put in more effort to find multiple interpretations of a drawing that was not 3-connected, the algorithm was still suitable only for manifolds of genus 0. Their second algorithm was an optimization-based procedure. The criterion they employed to formulate the face identification was based on the observation on face configuration and a basic theorem called the face adjacency theorem. The observation, serving as the criterion for the problem, is that, given a line drawing, human beings tend to choose a face configuration in which there are

as many edges as possible. The face adjacency theorem stated that two adjacent planar faces may coexist in the same object if and only if their common edges are collinear. This algorithm is suitable for a large set of drawings representing manifold and nonmanifold objects. However, it fails when handling the objects with internal faces. Liu and Lee [35] revisited the problem tackled by Shpitalni and Lipson and used the same criterion and face adjacency theorem to formulate the problem. They formulated the face identification as a maximum weight clique problem and developed a much faster algorithm to find faces in a line drawing. Their algorithm outputs the same results of face identification, and has the same problem, as Shpitalni and Lipsons. Liu et al. [38] and [37] proposed variable-length genetic algorithms with heuristic and geometric constraints incorporated for local search and tackled simultaneously the two combinatorial problems involved in the previous methods [33], [35].

The work in this thesis focuses on the 3D reconstruction of manifold objects with both planar and curved faces. Thus, we assume that given a line drawing, its face topology is known before the reconstruction of its 3D geometry. Here, the face topology denotes the set of circuits that represent all the faces of the 3D object. Among these face identification techniques, the work in [36] is most suitable for face identification in this thesis. Because, the previous approaches to the face identification from line drawings with hidden lines visible are still not satisfactory. For manifolds only, none of the previous algorithms can handle both the objects with the internal faces and with the holes. In addition, it seems that it is impossible to develop an efficient (polynomial) algorithm to handle drawings with genus ≥ 0. Liu et al. [36] proposed a new method based on a number of properties implied in line drawings representing manifold objects, used a tree search scheme to find the faces of manifolds. The two main steps in the method were 1) searching for cycles from a line drawing and 2) searching for faces from the cycles. In order to speed up the face identification procedure, a number of properties, most of which relate to planar manifold geometry in line drawings, were presented to identify most of the cycles that are or are not real faces in a drawing, thus reducing the number of unknown cycles in the second searching. Schemes to deal with manifolds of curved faces and manifolds each represented by two or more disjoint graphs were also proposed.

2.3.2 3D geometry reconstruction

To reconstruct the 3D object from single 2D line drawings, the main stream approach in the previous researches is to formulate the problem as an optimization problem based on different objective functions. Our work in this thesis belongs to this group.

3D planar object reconstruction from single line drawings

Marill [42] presented his method based on a simple criterion: minimizing the standard deviation of the angles in the reconstructed object, which is called the MSDA principle. This criterion can be used to inflate a 2D line drawing into a 3D shape. Marill's approach is tolerant of freehand sketching errors, but it can just reconstruct simple 3D objects, such as cubic, pyramid, stairs, etc. Motivated by the MSDA, Brown and Wang [6] proposed to minimize the standard deviation of the segment magnitudes (MSDSM) in the recovered planar object, and Shoji et al. [51] presented the criterion of minimizing the entropy of angle distribution (MEAD), and claimed that it is more general than both the MSDA and the MSDSM.

MSDA, MSDSM, and MEAD can only recover simple objects from line drawings and base on the criteria of regularities that humans perceive when interpreting 2D line drawings. Later, some researchers extended the criterions following this idea and incorporated more heuristic regularities in the reconstruction process. Leclerc and Fischler et al. [29] considered not only the MSDA, but also the regularity of face planarity for planar object reconstruction. By modifying Marill's objective function to explicitly favor planar-faced solutions, and by using a more competent optimization technique, this method performs better than MSDA, MSDSM, and MEAD. The methods in [47] and [65] concentrated on the reconstruction of symmetric polyhedra by developing a regularity of model symmetry. Lipson and Shpitalni [33] took Leclerc and Fischler's work further using more regularities for the reconstruction such as line parallelism, line verticality, isometry, corner orthogonality, skewed facial orthogonality and skewed facial symmetry, all of which are in accordance with human visual perception of line drawings. All these constraints are combined together to form an objective function to reconstruct more complex objects than all the previous methods. In this thesis, we proposed a new scheme handling the curved objects reconstruction, while we still keep using some regularities proposed in [33] and the idea of optimization-based reconstruction. Here, we give more detailed review of Lipson's method [33]. The regularities in [33] which we use in curved reconstruction are:

- **Line parallelism**: A parallel pair of lines in the sketch plane reflects parallelism in space. The term used to evaluate the parallelism of a line pair is

$$\alpha_{parallel} = w_{1,2}[\cos^{-1}(\hat{l_1} \cdot \hat{l_2})]^2 \qquad w_{1,2} = \mu_{0^0,7^0}[\cos^{-1}(\hat{l_1}' \cdot \hat{l_2}')] \quad (2.1)$$

  where $\hat{l_1}$ and $\hat{l_2}$ are the unit direction vectors of the first and second lines, respectively. A continuous compliance factor $\mu_{a,b}(x)$ is defined as $\mu_{a,b}(x) = \max[\ 0,\ 1.1 \cdot e^{-(\frac{x-a}{b})^2} - 0.1\ ]$. The weight $w_{1,2}$ given to this

regularity for a specific pair of lines depends on how the two lines are parallel in the sketch plane.

- **Corner orthogonality**: A junction of three lines that mathematically qualifies as a projection of a 3D orthogonal corner is orthogonal in space. To determine whether a junction of three lines in a plane qualifies as a projection of an orthogonal corner, a following test is applied [33], based on the fact that the projection of an orthogonal corner spans at least 90^0.

- **Isometry**: Lengths of entities in the 3D model are uniformly proportional to their lengths in the sketch plane. The term to account for non-uniformity corresponds to the standard deviation of scales.

$$\alpha_{isometry} = n \cdot \sigma^2(r_i = 1 \ldots N_e) \qquad r_i = \frac{length(entity_i)}{length'(entity_i)} \tag{2.2}$$

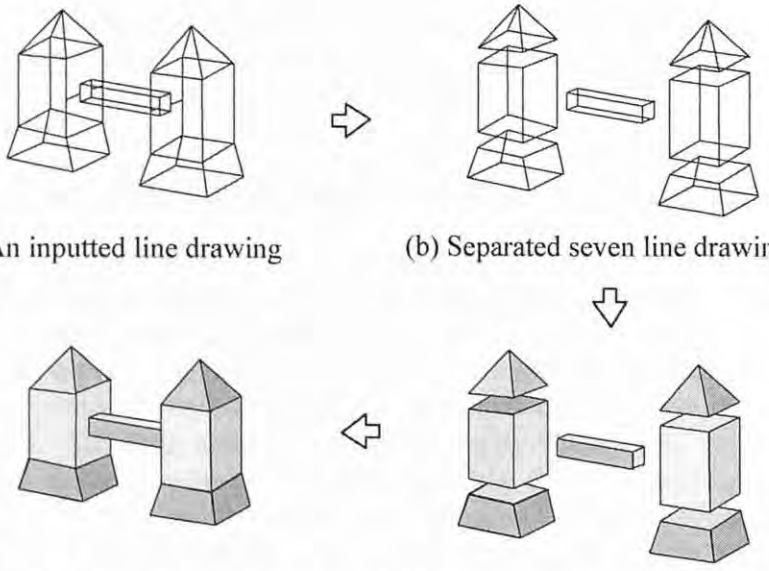
where n is the number of entities, N_e is the number of edges in the line drawing, r_i is the ratio between the current length of entity i and its length in the sketch plane, and σ is the standard deviation of the series of r_i.

Besides these three regularities, there are more regularities proposed in the [33]. When the reconstruction process begins, the given 2D edge-vertex graph is analyzed and image regularities are identified. For each regularity, the corresponding weighting coefficient is computed. A 3D configuration can be represented by a vector Z containing the z coordinates of the vertices. A compliance function $F(Z)$ can then be computed for any 3D configuration by summing the contributions of the regularity terms. Regularities are prefixed by a global balancing coefficient vector W. The final compliance function to be optimized takes the form

$$F(Z) = W^T \sum [\alpha] \tag{2.3}$$

where α is the vector containing all the constraints including $\alpha_{planarity}$, $\alpha_{parallel}$, $\alpha_{vertical}$, $\alpha_{isometry}$, $\alpha_{cornerskewedorthogonality}$, etc. The appropriate configuration of $z's$ (the vector Z) is sought using optimization.

Chen et al. [10] used a divide-and-conquer strategy shown in Fig. 2.3 to deal with complex objects reconstruction. In this thesis, we apply this method to deal with the complex line drawings with both planar and curved faces. The approach consists of three steps: 1) dividing a complex line drawing into multiple simpler line drawings based on the result of face identification; 2) reconstructing the 3D shapes from these simpler line drawings; 3) merging the 3D shapes into one complete object represented by the original line drawing.



(a) An inputted line drawing (b) Separated seven line drawings

(d) Combination of the seven parts (c) 3D reconstruction of the seven parts

Figure 2.3: Illustration of divide-and-conquer approach to the problem of line drawing reconstruction.

3D curved object reconstruction from single line drawings

The above 3D reconstruction methods handle only planar objects. Although few works are available to tackle curved objects, some efforts [7], [64], [9], [30], [44] do try to deal with this problem. In [7], 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. In [64], Varley et al's method cannot work directly on the line drawing of a curved object. Instead, it requires the user to create another line drawing of a polyhedral template first. When the polyhedron is reconstructed, some faces are bent to generate curved faces. Other limitation of this method is the templates cannot have holes. [30] and [9] focused on architectural modeling where the faces are mainly planar. Both methods need the user to help derive camera parameters, and the constructed models are actually 2.5D but not full 3D since no hidden lines are drawn. Both construct a model in a progressive way: one polygon after another in [30] and one primitive after another in [9]. Besides, [9] heavily relies on a primitive database and when the user finishes drawing a primitive, he needs to specify its type to help reconstruction. [44] also used a progressive way to do 3D reconstruction where the constructed models are full 3D since hidden lines are drawn. One limitation of [44] is that it requires the edges of an object exhibit pronounced angular trends so that an underlying axis system

can be found. From the experimental results in [30], [9], [44], it is easy to see that they are limited to simple curved surfaces that are symmetric in [9] or (parts of) cylinders or spheres in [30] and [44].

2.4 Our research topic and contributions

In previous sections, we have reviewed the contributions and problems of these previous researches related to interpretation of line drawings. This thesis mainly concentrates on the topics of 3D reconstruction of curved objects from single 2D line drawings in view of the limitations in the previous work: most of previous methods for line drawing reconstruction just concentrated on planar objects. Although there are little few works dealing with curved object, all of these methods cannot reconstruct the line drawings automatically, but in a progressive way using human interaction or template. Moreover, these methods fail in many complex curved objects and limit to very simple curved surfaces. To the best of our knowledge, no method has been published to tackle curved objects with more complex and general geometrical structure in an automatic way.

In light of this problem, we present a novel approach which is mainly devoted to handle complex curved objects. This method creates full 3D models and does not require the user to derive camera parameters or to draw an object with edges mainly in the axis directions. It works in an automatic way, so it can handle more complex objects in essence. The techniques, identifying curved/planar faces in a line drawing, creating a generalized polyhedron, and regularities for curved object reconstruction, are novel contributions in this thesis. Our approach can tackle the objects with more complex general curved surfaces among all the researches in this area.

Chapter 3

Reconstruction of Curved Manifold Objects

In this chapter, we first state the assumptions for the curved object reconstruction problem and the terms that will be used in what follows, then develop our approach to this problem, and finally give the corresponding algorithm.

3.1 Assumptions and terminology

The following four assumptions are made before the reconstruction problem is formulated.

- 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.
- Every curved edge of a line drawing is the projection of a 3D planar curve.
- All the faces of a manifold that a line drawing represents are available.
- Each face is either planar or curved. For a planar face, two principal curvatures are both zero at every point of surface, that is $\mathcal{K}_1 = 0$, $\mathcal{K}_2 = 0$. For a curved face, two principal curvatures can be nonzero. But we assume that the curved face we deal with can be transformed to piecewise surfaces whose definition is shown below.

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 theis, 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 [32] and [36], as mentioned in Section 2.3.1. This is not the focus of this thesis 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 thesis.

- **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 [36].
- **Face**. A face is one of the surface patches of a manifold bounded by edges.
- **Edge**. An edge of a line drawing is the intersection of two non-coplanar real faces, which can be a curve or straight line. An edge e is also denoted by $\{v_{e_1}, v_{e_2}\}$ where v_{e_1} and v_{e_2} are two vertices of e.
- **Cycle**. A cycle is formed by a sequence of vertices $v_0, v_1, \cdots, v_n$, where $n \geq 3$, $v_0 = v_n$, where n vertices are distinct, and there exists an edge connecting v_i and v_{i+1} for $i = 0, 1, \cdots, n-1$. A cycle is denoted by $\{v_0, v_1, \cdots, v_n\}$. A face is a cycle.
- **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.
- **Developable surface**. At least one of the principal curvatures is zero at every point.
- **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 [16].
- **Piecewise surface**. The surface is segmented from a curved face including silhouette. The slope of the tangent at every point along silhouette in piecewise surface has the same sign (negative or positive).

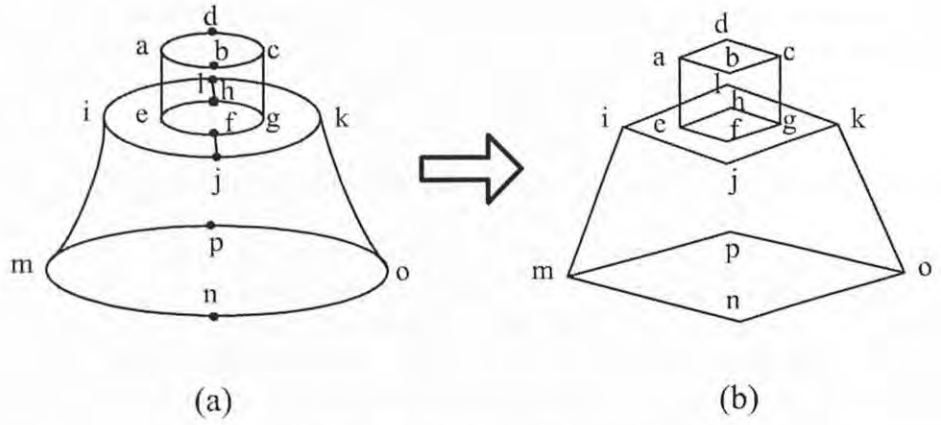


Figure 3.1: Illustration of some terms.

- **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.
- **Farthest point**. The farthest points of a curve are defined as the points having the maximal distance to the line passing through the two end-points of the curve.
- **Singular point**. Singular point on a curve $f(x, y) = 0$ is the point if the x and y partial derivatives of f are both zero at this point.
- **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.
- **Genus** The genus of a surface can be considered as the number of holes that pass through it completely.

Many of these terms are illustrated with the line drawings in Fig. 3.1. 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*, *cg*, *im*, and *ko* in Fig. 3.1(a) are four silhouettes of the line drawing. Points *d* and *b* are the farthest points of curves *adc* and *abc*, respectively. Fig. 3.1(b) shows the line drawing of the generalized polyhedron corresponding to the curved manifold represented by the line drawing in Fig. 3.1(a). There

[1]Artificial lines have been used in solid modeling to indicate the co-surface of two cycles in a line drawing [2], [10], [36]. Without them, it is impossible to determine the 3D geometric relation between the two cycles (see Fig. 3.1 for example).

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 3.2.2.

3.2 Reconstruction of curved manifold objects

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.

3.2.1 Distinguishing between curved and planar faces

Before 3D reconstruction, it is helpful to find whether a face is curved or planar. In [18], 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 [18] 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], [10], [36]. 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], [10], [36]. Here these cycles are still called faces, which have the following property.

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

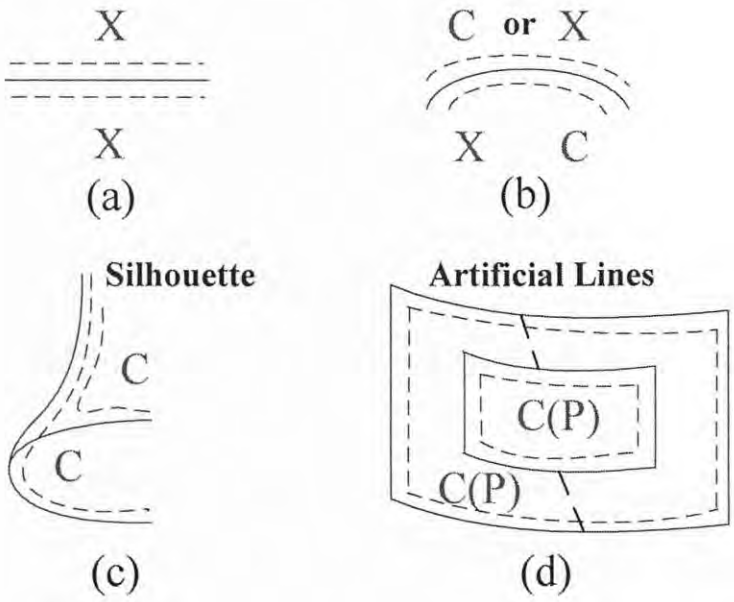


Figure 3.2: Illustration of Properties 1–4. (a) A straight edge. (b) A curved edge. (c) A silhouette. (d) Two faces connected by two artificial lines.

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. 3.2. 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 [18]. 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

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.

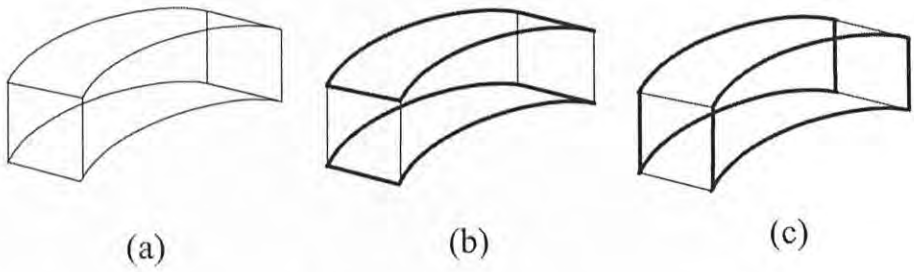


Figure 3.3: A line drawing (a) with two correct labeling configurations (b) and (c). Only curved faces are marked with bold edges.

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 each 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. 3.3.

3.2.2 Transformation of Line Drawings

Transforming a line drawing with curved edges into the 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 one, we can reconstruct the 3D wireframe of the curved object by combining previous regularities for planar objects and our new regularities for curved objects. Based on the careful observations, we find that there are many 3D objects whose boundaries contain silhouettes with curve lines. For these objects, we have to add some curves on the undevelopable surface before transforming the line drawing to make the reconstructed results more accurate and closer to human perception. After this operation, we then carry out our transformation to a line drawing without curve.

Transformation of undevelopable surface

If a curved face whose boundary contains silhouette is an undevelopable surface, we add some curves automatically to transform it into some piecewise surfaces.

As in Fig. 3.4 shown, the back and front curved faces (p_0, p_1, Q_1, Q_0) are

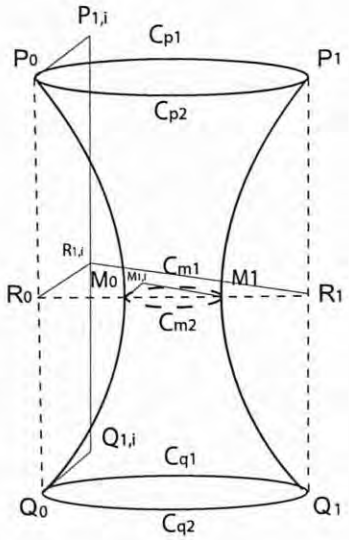


Figure 3.4: Transformation of an undevelopable surface into piecewise surfaces

both undevelopable faces. Firstly, we calculate the singular point of one silhouette. We should mention that the silhouettes may not be totally symmetrical. The Bezier curve in R^2 is described in the parameterized form $C(t) = (x(t), y(t))$. The singular points are those points where

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \frac{\mathrm{d}y(t)}{\mathrm{d}t} = 0. \tag{3.1}$$

In Fig. 3.4, we calculate singular points of curve P_0Q_0, that is M_0.

Secondly, we find the corresponding point in the other silhouette. The corresponding point M_1 as shown in Fig. 3.4 should satisfy the below equation,

$$k = \frac{\|P_0R_0\|}{\|P_0Q_0\|} = \frac{\|P_1R_1\|}{\|P_1Q_1\|}, \tag{3.2}$$

where k represents the proportion of two line segments' length. As seen in the Fig. 3.4, we can get some equations from the geometric properties.

$$\overrightarrow{M_0R_0} = a\overrightarrow{M_0R_1}, \tag{3.3}$$

$$\overrightarrow{M_0R_0} = M_0 - (P_0 + k(Q_0 - P_0)), \tag{3.4}$$

$$\overrightarrow{M_0R_1} = M_0 - (P_1 + k(Q_1 - P_1)), \tag{3.5}$$

where a is the coefficient.

We can put the coordinates of M_0, P_0, P_1, Q_0, Q_1 into the equation(3.3), (3.4) and (3.5) to get an quadratic equation with k. By solving the equation,

we can get the value of k at the interval of [0,1]. When k is known, we can get coordinates of R_0,R_1, and finally M_1.

At last, control points of each curve which we add can be calculated. As we use Bezier curve to represent curves, the existing curves' equations are

$$C_{p1}(t) = \sum_{i=0}^{n} P_{1,i} B_{i,n}(t), \tag{3.6}$$

$$C_{q1}(t) = \sum_{i=0}^{n} Q_{1,i} B_{i,n}(t), \tag{3.7}$$

where $P_{1,i}$ and $Q_{1,i}$ are the control points of curves $C_{p1}(t)$ and $C_{q1}(t)$ respectively. Also $P_{1,0} = P_0$, $P_{1,n} = P_1$ and $Q_{1,0} = Q_0$, $Q_{1,n} = Q_1$.

In order to calculate the control points of curve C_{m1}, we introduce new points $R_{1,i}$ firstly, as shown in Fig. 3.4,

$$R_{1,i} = (1-k)P_{1,i} + kQ_{1,i}, \tag{3.8}$$

It is easy to prove that triangle $R_0R_{1,i}R_1$ is similar to triangle $M_0M_{1,i}M_1$. Then we use $R_{1,i}$ to get control points $M_{1,i}$,

$$M_{1,i} = M_0 + \frac{\|M_1 - M_0\|}{\|R_1 - R_0\|}(R_{1,i} - R_0), \qquad i = 0, 1, ...n. \tag{3.9}$$

Then we can find its corresponding Bezier curve C_{m1},

$$C_{m1}(t) = \sum_{i=0}^{n} M_{1,i} B_{i,n}(t), \tag{3.10}$$

Similarly, Bezier curve C_{m2} can be found in the same way. Thus, we automatically calculate the two curves and add them on the line drawing in order to transform undevelopable surface into two piecewise surfaces. By adding curves, the original Bezier curve P_0Q_0 is split into two Bezier curves P_0M_0 and M_0Q_0, so is curve P_1Q_1. [22] presents the subdivision algorithm of Bezier curve. By using this method, we could get new Bezier curves which are continuous at the split point. If the silhouette contains more than one singular point, we could add more curves in the similar way. After this transformation of undevelopable surface, a new line drawing which is regarded as the input line drawings has been generated for reconstruction. We apply the Algorithms 1 described in Section 3.2.1 to distinguish between curved and planar faces, and then transform it into a line drawing without curve.

Transformation to a line drawing without curve

Our scheme of line drawing transformation is described as follows. When there is only one curve between two vertices, as shown in Fig. 3.5(a), the curve is

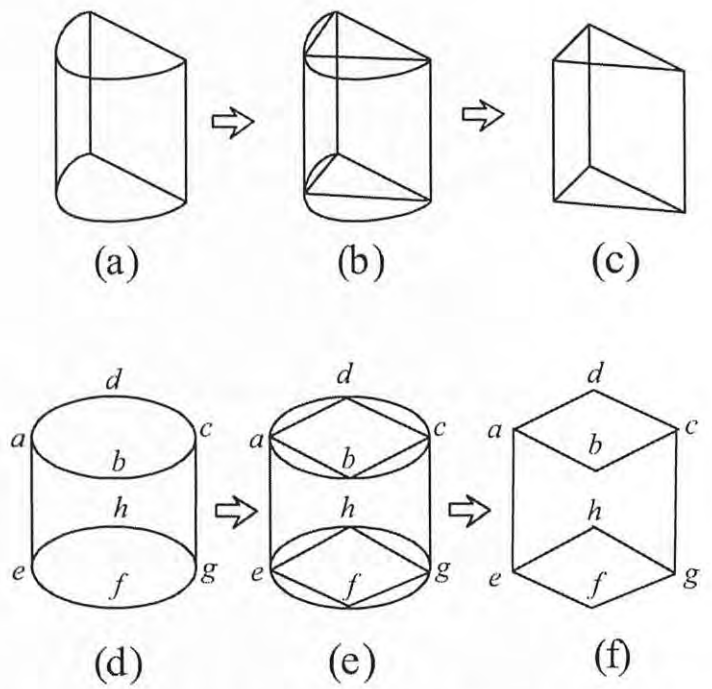


Figure 3.5: Examples of the transformation of line drawings, where (c) is a polyhedron and (f) is a generalized polyhedron.

straightened (see Figs. 3.5(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. 3.5(d), the curved edges cannot be straightened directly because otherwise the two edges become one. In this case, we first find a farthest point in a curve and then replace the curve with two straight lines (Fig. 3.5(e)). In Fig. 3.5(c), the line drawing represents a polyhedron, while in Fig. 3.5(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.

3.2.3 Regularities

Many previous regularities developed for handling planar object reconstruction can be used to reconstruct a polyhedron such as the one in Fig. 3.5(c). However, a generalized polyhedron with generalized faces does not exist actually (see Fig. 3.5(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 3.1. *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*

$$s = \frac{\int_0^s \|\mathbf{G}'(u)\| \mathrm{d}u}{\int_0^1 \|\mathbf{G}'(u)\| \mathrm{d}u} \tag{3.11}$$

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

Note that (3.11) is the condition that s satisfies; it is not used to determine the arc length s.

Definition 3.2. *The parallelism between two curves* $\mathbf{C}_1(t)$ *and* $\mathbf{C}_2(t)$ *in* $\mathcal{R}^n$ *is defined as*

$$r_{1,2} = \max\left\{ \int_0^1 \frac{\mathbf{G}_1'(s)^T \mathbf{G}_2'(s)}{\|\mathbf{G}_1'(s)\| \cdot \|\mathbf{G}_2'(s)\|} \mathrm{d}s, \quad \int_0^1 \frac{\mathbf{G}_1'(s)^T \mathbf{G}_2'(1-s)}{\|\mathbf{G}_1'(s)\| \cdot \|\mathbf{G}_2'(1-s)\|} \mathrm{d}s \right\}, \tag{3.12}$$

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

$$\alpha_{CP} = \sum_{i,j} w_{i,j}^{CP} (1 - r_{i,j}^{3D})^2, \tag{3.13}$$

where the weighting factor

$$w_{i,j}^{CP} = \frac{1}{1 + exp\left[-\sigma_1 (r_{i,j}^{2D} - 0.8)\right]}, \tag{3.14}$$

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 [33]. 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. 3.5(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:

$$\alpha_{GFP} = \sum_{i=1}^{K} [\sin^{-1}(\mathbf{n}_{i1} \cdot \mathbf{n}_{i2})]^2, \tag{3.15}$$

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 [33]. This regularity requires that two concurved edges in the 2D sketch plane are also concurved in 3D space. According to the definition in Section 3.1, 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. 3.6 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:

$$\alpha_{CC} = \sum_{i=1}^{N} \sum_{\substack{j,k \in \mathcal{E}(i) \\ j \neq k}} w_{ijk}^{CC} \left(\frac{\|(\mathbf{P}_j - \mathbf{P}_i) \times (\mathbf{P}_k - \mathbf{P}_i)\|}{\|\mathbf{P}_j - \mathbf{P}_i\| \cdot \|\mathbf{P}_k - \mathbf{P}_i\|} \right)^2, \tag{3.16}$$

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

$$w_{ijk}^{CC} = \frac{1}{1 + exp\ [-\sigma_2(\beta_{ijk} - 8\pi/9)]}, \tag{3.17}$$

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 $\sigma_2 = 100$.

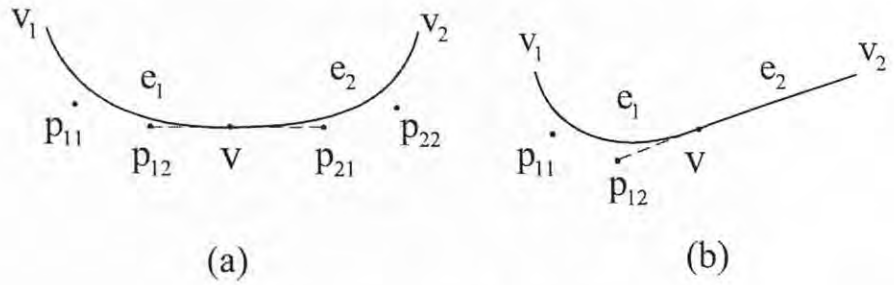


Figure 3.6: Illustration of curve concurvity.

3.2.4 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) [42], face planarity (α_2) [29] (effective on the planar faces but not the generalized faces), line parallelism (α_3) [33], and corner orthogonality (α_4) [33]. The three new regularities (denoted as α_5, α_6, and α_7) proposed in Section 3.2.3 and the regularity isometry (α_8) [33] 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 $\mathbf{C}(t)$ be described in the parameterized form

$$\mathbf{C}(t) = (x(t), y(t), z(t)). \tag{3.18}$$

Under parallel projection, the 2D image of $\mathbf{C}(t)$ is simply

$$\bar{\mathbf{C}}(t) = (x(t), y(t)). \tag{3.19}$$

Now we need to recover $z(t)$ from $\bar{\mathbf{C}}(t)$. In general, given $\bar{\mathbf{C}}(t)$ 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.

Obviously, a plane can be defined with a point and a normal vector,

$$n_x(x(t) - x_0) + n_y(y(t) - y_0)) + n_z(z(t) - z_0) = 0 \tag{3.20}$$

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 which the 3D curve $\mathbf{C}(t) = (x(t), y(t), z(t))^T$ lies on, we can recover the depth of the curve by

$$z(t) = z_0 - (n_x(x(t) - x_0) + n_y(y(t) - y_0))/n_z. \tag{3.21}$$

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.

The manipulation of the curve in the form of (3.18) is not convenient because we do not have explicit expressions for $x(t)$ and $y(t)$. We use Bezier curves to represent general curves in this thesis.

Bezier curves are very popular in the representation of curves. Bezier curves have good mathematical properties which enable them to be manipulated and analyzed easily. They are also convenient to be drawn with a few control points [5].

A Bezier curve of degree n is specified by a sequence of $n + 1$ points $\mathbf{P}_i$, $0 \leq i \leq n$, which are called control points. Its equation is

$$\mathbf{C}(t) = \sum_{i=0}^{n} \mathbf{P}_i B_{i,n}(t), \tag{3.22}$$

where $B_{i,n}(t)$ is the Bernstein polynomial function

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}. \tag{3.23}$$

From (3.22), we can see that the $n + 1$ control points $\mathbf{P}_i$ define the curve completely. In our reconstruction tool, the user is able to move the control points freely to change the shape of a curve. If a line drawing is obtained by scanning a sketch image, we can find the control points by fitting a Bezier curve to an inputted curve using the technique in [46]. When the control points are collinear, the Bezier curve degenerates to a straight line. Thus, it can be used to represent straight lines too.

Let the control points of a curve $\bar{\mathbf{C}}(t)$ in the 2D projection plane be $\bar{\mathbf{P}}_\mathbf{i} = (x_i, y_i)$, meaning that

$$\bar{\mathbf{C}}(t) = \sum_{i=0}^{n} \bar{\mathbf{P}}_i B_{i,n}(t).$$

Then we can find its corresponding Bezier curve $\mathbf{C}(t)$ in 3D space:

$$\mathbf{C}(t) = \sum_{i=0}^{n} \mathbf{P}_i B_{i,n}(t),$$

by deriving $\mathbf{P}_i = (x_i, y_i, z_i)$ with $z_i = z_0 - (n_x(x_i - x_0) + n_y(y_i - y_0))/n_z$ in (3.21).

Here, we need to determine the unit normal vector $\mathbf{n} = (n_x, n_y, n_z)^T$ of the plane each 3D curve lies on. In this thesis, 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:

$$F(z_1, z_2, \cdots, z_N, \mathbf{n}_1, \mathbf{n}_2, \cdots, \mathbf{n}_M) = \sum_{i=1}^{8} \lambda_i \alpha_i. \tag{3.24}$$

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 [29] to minimize F. Note that in (3.24), 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.

3.2.5 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 surface patches

Bezier and Coons patches [19] are suitable for patches with four boundaries (patches with three boundaries are a special case). A Bezier surface patch $\mathbf{S}(u, v)$ is defined by a $(n+1) \times (m+1)$ array of control points $\mathbf{P}_{i,j}$, as shown in Fig. 3.7. The parametric form of $\mathbf{S}(u, v)$ is

$$\mathbf{S}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{P}_{i,j} B_{i,n}(u) B_{j,m}(v). \tag{3.25}$$

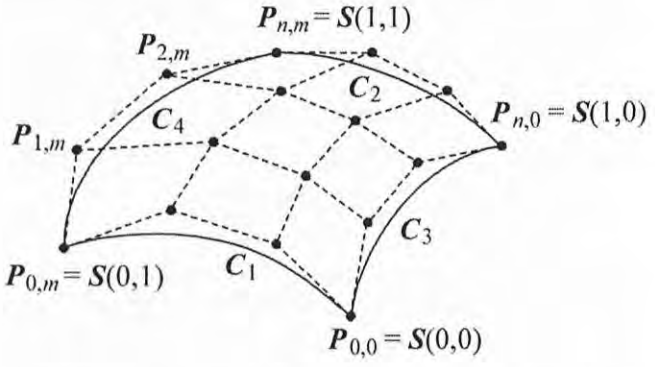


Figure 3.7: Bezier surface patch $\mathbf{S}(u, v)$ and its control points.

Now we derive the equations of the boundaries of the patch $\mathbf{S}(u, v)$. A Bezier patch comprises two families of parameterized curves $\mathbf{C}(u)$ and $\mathbf{C}(v)$. Without loss of generality, the parameters u, v are chosen to lie in the interval $[0, 1]$. Thus the boundaries of $\mathbf{S}(u, v)$ are four curves $\mathbf{C}_1(v)$, $\mathbf{C}_2(v)$, $\mathbf{C}_3(u)$, and $\mathbf{C}_4(u)$, with respect to $u = 0, 1$ and $v = 0, 1$. Considering $\mathbf{C}_1(v)$ where $u = 0$, we have

$$\begin{aligned}\mathbf{C}_1(v) = \mathbf{S}(0, v) &= \sum_{j=0}^{m} B_{j,m}(v) \sum_{i=0}^{n} \mathbf{P}_{i,j} B_{i,n}(0) \\ &= \sum_{j=0}^{m} \mathbf{P}_{0,j} B_{j,m}(v), \end{aligned} \tag{3.26}$$

since $B_{0,n}(0) = 1$ and $B_{i,n}(0) = 0$ for all $i \neq 0$. Similarly, we have the other three boundaries

$$\mathbf{C}_2(v) = \mathbf{S}(1, v) = \sum_{j=0}^{m} \mathbf{P}_{n,j} B_{j,m}(v), \tag{3.27}$$

$$\mathbf{C}_3(u) = \mathbf{S}(u, 0) = \sum_{i=0}^{n} \mathbf{P}_{i,0} B_{i,n}(u), \tag{3.28}$$

$$\mathbf{C}_4(u) = \mathbf{S}(u, 1) = \sum_{i=0}^{n} \mathbf{P}_{i,m} B_{i,n}(u). \tag{3.29}$$

Obviously, $\mathbf{C}_{1-4}$ are Bezier curves defined by their control points, which are also part of the $(n + 1) \times (m + 1)$ control points of $\mathbf{S}(u, v)$ in (3.25).

Given the patch boundaries represented by the 3D Bezier curves $\mathbf{C}_{1-4}$, our final task is now to generate the Bezier patch. This is equivalent to finding all the other control points $\mathbf{P}_{i,j}$ of $\mathbf{S}(u, v)$ based on the known control points, $\mathbf{P}_{0,j}$, $\mathbf{P}_{n,j}$, $\mathbf{P}_{i,0}$, and $\mathbf{P}_{i,m}$, of $\mathbf{C}_{1-4}$ where $0 \leq i \leq n, 0 \leq j \leq m$.

Consider three new surfaces obtained from the boundaries:

$$\mathbf{S}_1(u,v) = (1-u)\mathbf{S}(0,v) + u\mathbf{S}(1,v), \tag{3.30a}$$
$$\mathbf{S}_2(u,v) = (1-v)\mathbf{S}(u,0) + v\mathbf{S}(u,1), \tag{3.30b}$$
$$\begin{aligned}\mathbf{S}_3(u,v) = {} & (1-u)(1-v)\mathbf{S}(0,0) + (1-u)v\mathbf{S}(0,1) \\ & + u(1-v)\mathbf{S}(1,0) + uv\mathbf{S}(1,1).\end{aligned} \tag{3.30c}$$

Here, (3.30a) is a surface passing through curves $\mathbf{S}(0,v)$ and $\mathbf{S}(1,v)$, (3.30b) is a surface passing through curves $\mathbf{S}(u,0)$ and $\mathbf{S}(u,1)$, and (3.30c) is a surface passing through the four corners, $\mathbf{S}(0,0)$, $\mathbf{S}(0,1)$, $\mathbf{S}(1,0)$, and $\mathbf{S}(1,1)$, of the patch $\mathbf{S}(u,v)$. If we consider the surface defined by the sum $\mathbf{S}_1(u,v)+\mathbf{S}_2(u,v)$, we will find that each corner is counted twice. Hence, if we subtract $\mathbf{S}_3(u,v)$ from the sum, we will recover a surface that passes through the four boundaries with each corner counted once:

$$\begin{aligned}\mathbf{S}^*(u,v) &= \mathbf{S}_1(u,v) + \mathbf{S}_2(u,v) - \mathbf{S}_3(u,v) \\ &= (1-u)\mathbf{S}(0,v) + u\mathbf{S}(1,v) + (1-v)\mathbf{S}(u,0) + v\mathbf{S}(u,1) \\ &\quad - (1-u)(1-v)\mathbf{S}(0,0) - (1-u)v\mathbf{S}(0,1) \\ &\quad - u(1-v)\mathbf{S}(1,0) - uv\mathbf{S}(1,1).\end{aligned} \tag{3.31}$$

$\mathbf{S}^*(u,v)$ is called the Coons surface patch [19], which 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 $\mathbf{C}_{1-4}$ where $0 \le i \le n, 0 \le j \le m$. The interpolation equation is written as

$$\begin{aligned}\mathbf{P}_{i,j} = {} & (1-\frac{i}{n})\mathbf{P}_{0,j} + \frac{i}{n}\mathbf{P}_{n,j} + (1-\frac{j}{m})\mathbf{P}_{i,0} + \frac{j}{m}\mathbf{P}_{i,m} \\ & - (1-\frac{i}{n})(1-\frac{j}{m})\mathbf{P}_{0,0} - (1-\frac{i}{n})\frac{j}{m}\mathbf{P}_{0,m} \\ & - \frac{i}{n}(1-\frac{j}{m})\mathbf{P}_{n,0} - \frac{i}{n}\frac{j}{m}\mathbf{P}_{n,m}.\end{aligned} \tag{3.32}$$

The following proposition points out that the control points chosen in such a way generate a Bezier patch equivalent to the Coons patch.

Proposition 1. *The Bezier patch* $\mathbf{S}(u,v)$ *defined by the* $(n+1)\times(m+1)$ *control points* $\mathbf{P}_{i,j}$ *in* (3.32) *is the same as the Coons patch in* (3.31).

Proof. Substituting $\mathbf{P}_{i,j}$ in (3.32) into $\mathbf{S}(u,v)$ in (3.25) yields

$$\begin{aligned}
\mathbf{S}(u,v) &= \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,n}(u)B_{j,m}(v)\mathbf{P}_{i,j} \\
&= \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,n}(u)B_{j,m}(v)\big[(1-\frac{i}{n})\mathbf{P}_{0,j} + \frac{i}{n}\mathbf{P}_{n,j}\big] \\
&\quad + \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,n}(u)B_{j,m}(v)\big[(1-\frac{j}{m})\mathbf{P}_{i,0} + \frac{j}{m}\mathbf{P}_{i,m}\big] \\
&\quad - \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,n}(u)B_{j,m}(v)\big[(1-\frac{i}{n})(1-\frac{j}{m})\mathbf{P}_{0,0} \\
&\quad + (1-\frac{i}{n})\frac{j}{m}\mathbf{P}_{0,m} + \frac{i}{n}(1-\frac{j}{m})\mathbf{P}_{n,0} + \frac{i}{n}\frac{j}{m}\mathbf{P}_{n,m}\big].
\end{aligned} \tag{3.33}$$

Further we simplify some terms in (3.33).

$$\begin{aligned}
\sum_{i=0}^{n} B_{i,n}(u)\frac{i}{n} &= \sum_{i=0}^{n}\binom{n}{i}u^i(1-u)^{n-i}\frac{i}{n} \\
&= 0 + \sum_{i=1}^{n}\binom{n}{i}u^i(1-u)^{n-i}\frac{i}{n}.
\end{aligned}$$

Replacing i with $k = i-1$,

$$\begin{aligned}
\sum_{i=0}^{n} B_{i,n}(u)\frac{i}{n} &= \sum_{k=0}^{n-1}\binom{n}{k+1}u^{k+1}(1-u)^{n-1-k}\frac{k+1}{n} \\
&= u\sum_{k=0}^{n-1}\binom{n-1}{k}u^k(1-u)^{n-1-k} \\
&= u[u+(1-u)]^{n-1} = u.
\end{aligned}$$

Similarly, we can derive

$$\sum_{i=0}^{n} B_{i,n}(u)(1-\frac{i}{n}) = 1-u,$$

$$\sum_{j=0}^{m} B_{j,m}(v)\frac{j}{m} = v, \quad \sum_{j=0}^{m} B_{j,m}(v)(1-\frac{j}{m}) = 1-v.$$

Hence, (3.33) becomes

$$\begin{aligned}\mathbf{S}(u,v) &= \sum_{j=0}^{m}[(1-u)\mathbf{P}_{0,j} + u\mathbf{P}_{n,j}]B_{j,m}(v) + \sum_{i=0}^{n}[(1-v)\mathbf{P}_{i,0} + v\mathbf{P}_{i,m}]B_{i,n}(u) \\ &- (1-u)(1-v)\mathbf{P}_{0,0} - (1-u)v\mathbf{P}_{0,m} - u(1-v)\mathbf{P}_{n,0} - uv\mathbf{P}_{n,m} \\ &= (1-u)\mathbf{S}(0,v) + u\mathbf{S}(1,v) + (1-v)\mathbf{S}(u,0) + v\mathbf{S}(u,1) \\ &- (1-u)(1-v)\mathbf{S}(0,0) - (1-u)v\mathbf{S}(0,1) - u(1-v)\mathbf{S}(1,0) - uv\mathbf{S}(1,1),\end{aligned}$$

which is equal to $\mathbf{S}^*(u,v)$ in (3.31), and thus completes the proof. □

Although topologically-rectangular surface patches are most commonly used, there exist triangular faces in line drawings. For a three-boundary patch, we treat one corner as a special Bezier "curve" where all the control points are the same and coincident at that corner. In such a way, a three-boundary face can also be recovered with the same reconstruction scheme.

Triangle mesh

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:

$$\begin{aligned}Q(\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_K) &= \lambda \sum_i \sum_{j \in \mathcal{N}(i)} \|\mathbf{u}_i - \mathbf{u}_j\|^2 \\ &+ \gamma \sum_{i \notin \mathcal{S}} \sum_{j \in \mathcal{N}(i) \setminus \mathcal{S}} \|\mathbf{c}_i - \mathbf{c}_j\|^2 + \sum_{i \in \mathcal{S}} \|\mathbf{u}_i - \mathbf{u}'_i\|^2, \end{aligned} \tag{3.34}$$

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 (3.34), 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 [58]

$$\mathbf{c}_i = \mathbf{u}_i - \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{u}_j, i \notin \mathcal{S}, \tag{3.35}$$

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 derived by the process shown below.

First notice that the objective value $Q(\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_K)$ can be decomposed into the sum of function Q over each single coordinate, for that $\mathbf{u}_i$ represents a 3D mesh points (x_i, y_i, z_i). Therefore, we could optimize Q over each coordinate of $(\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_K)$ and then combine them together. Now we focus on the following optimization problem:

$$\begin{aligned} Q(u_1, u_2, \cdots, u_K) &= \lambda \sum_i \sum_{j \in \mathcal{N}(i)} |u_i - u_j|^2 \\ &+ \gamma \sum_{i \notin \mathcal{S}} \sum_{j \in \mathcal{N}(i) \setminus \mathcal{S}} |c_i - c_j|^2 + \sum_{i \in \mathcal{S}} |u_i - u_i'|^2, \end{aligned} \tag{3.36}$$

where $u_i \in \mathbb{R}, i = 1, 2, \cdots, K$ and

$$c_i = u_i - \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} u_j, \quad i \notin \mathcal{S}.$$

In the graph setting, we assume a connect graph $G = (V, E)$ with node set $V = \{1, \cdots, K\}$ and edge set $E = \{(i, j) | j \in \mathcal{N}(i)\}$. The Laplacian matrix $\mathbf{\Delta}$ is defined as below:

$$\mathbf{\Delta}_{ij} \triangleq \begin{cases} \deg(i) & \text{if} \quad i = j \\ -1 & \text{if} \quad i \neq j \quad \text{and} \quad j \in \mathcal{N}(i) \\ 0 & \text{otherwise} \end{cases} \tag{3.37}$$

Let the column vector $\mathbf{u} = \{u_1, \cdots, u_K\}^T$, the first term of (3.36) can be written as $\lambda \mathbf{u}^T \mathbf{\Delta} \mathbf{u}$. For any node subset $\mathcal{A} \subseteq V$, let $\mathbf{\Delta}_{\mathcal{A}}$ be the $|\mathcal{A}| \times |\mathcal{A}|$ Laplacian matrix of the subgraph induced by $\mathcal{A}$. And let $\mathbf{M}$ be the transformation matrix from $\mathbf{u}$ to $\mathbf{c} = (c_1, \cdots, c_{|\bar{\mathcal{S}}|})^T$: $\mathbf{c} = \mathbf{M}\mathbf{u}$ and $\bar{\mathcal{S}}$ be the complement set of $\mathcal{S}$. The second term of (3.36) can be written as: $\gamma \mathbf{c}^T \mathbf{\Delta}_{\bar{\mathcal{S}}} \mathbf{c} = \gamma \mathbf{u}^T \mathbf{M}^T \mathbf{\Delta}_{\bar{\mathcal{S}}} \mathbf{M} \mathbf{u}$. For any node subset $\mathcal{A} \subseteq V$, define the diagonal matrix $\mathbf{I}_{\mathcal{A}}$, where $\mathbf{I}_{\mathcal{A}}(i, i) = 1$ if $i \in \mathcal{A}$ and 0 otherwise. Let $\mathbf{u}' = \{u_1', \cdots, u_K'\}^T$, the third term of (3.36) then can be written as $(\mathbf{u} - \mathbf{u}')^T \mathbf{I}_{\mathcal{S}} (\mathbf{u} - \mathbf{u}')$. In sum, the objective function can be expressed as

$$\lambda \mathbf{u}^T \mathbf{\Delta} \mathbf{u} + \gamma \mathbf{u}^T \mathbf{M}^T \mathbf{\Delta}_{\bar{\mathcal{S}}} \mathbf{M} \mathbf{u} + (\mathbf{u} - \mathbf{u}')^T \mathbf{I}_{\mathcal{S}} (\mathbf{u} - \mathbf{u}'). \tag{3.38}$$

Taking the derivative of (3.38) with respect to $\mathbf{u}$ and setting it to zero, we obtain the closed form solution of $\mathbf{u}$:

$$\mathbf{u} = (\lambda \mathbf{\Delta} + \gamma \mathbf{M}^T \mathbf{\Delta}_{\bar{\mathcal{S}}} \mathbf{M} + \mathbf{I}_{\mathcal{S}})^{-1} \mathbf{I}_{\mathcal{S}} \mathbf{u}'. \tag{3.39}$$

3.2.6 The Complete 3D Reconstruction Algorithm

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

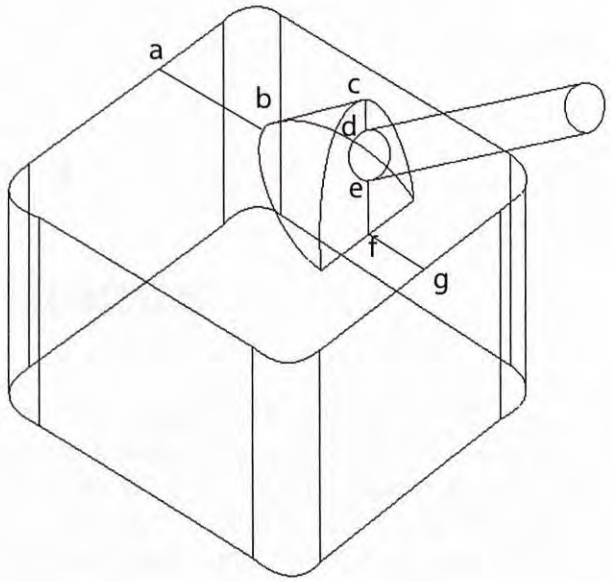


Figure 3.8: Illustration of adding artifical line on line drawing

When a line drawing represents a complex object, as shown in Fig. 3.8, we firstly add the artificial lines ab, cd, ef, and fg on the original line drawing. Then we use the divide-and-conquer technique in [10] 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.

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.

Chapter 4

Experiments

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 λ_{1-8} are chosen to be 100, 1, 80, 20, 80, 80, 80, and 15, respectively. The parameters λ and γ in (3.34) 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. They are not sensitive. For example, λ_{1-8} can be chosen in the ranges of $[90, 110]$, $[1, 5]$, $[70, 90]$, $[15, 25]$, $[70, 90]$, $[70, 90]$, $[70, 90]$, $[10, 20]$, respectively, and the results are similar.

Fig. 4.1 $\sim$ 4.4 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. Here, the original line drawing of teapot in Fig. 4.4 do not include the curves on ampullae. Before using the algorithm 2 for reconstruction, we apply method described in Section 3.2.2 to transform undevelopable surface into piecewise surfaces by adding two curves. Then the line drawing of teapot has been changed as shown in Fig. 4.4 and then reconstructed. Like the teapot, the lamp in Fig. 4.3 uses the similar method.

From Fig. 4.1 $\sim$ 4.4, we can see that the results accord with our visual perception very well. The input line drawings are not required to be very accurate, which can be seen from the figures. Note that some results may not look so perfect, such as the teapot. The ampullae is not as round as the real teapot as well as the position of pothole and hold are not perfectly right. 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. 4.1 $\sim$ 4.4, it ranges from 8.0 seconds to 127 seconds. Among the four steps given in Section 3.2.6, Steps 3 and 4 take almost all the computational time.

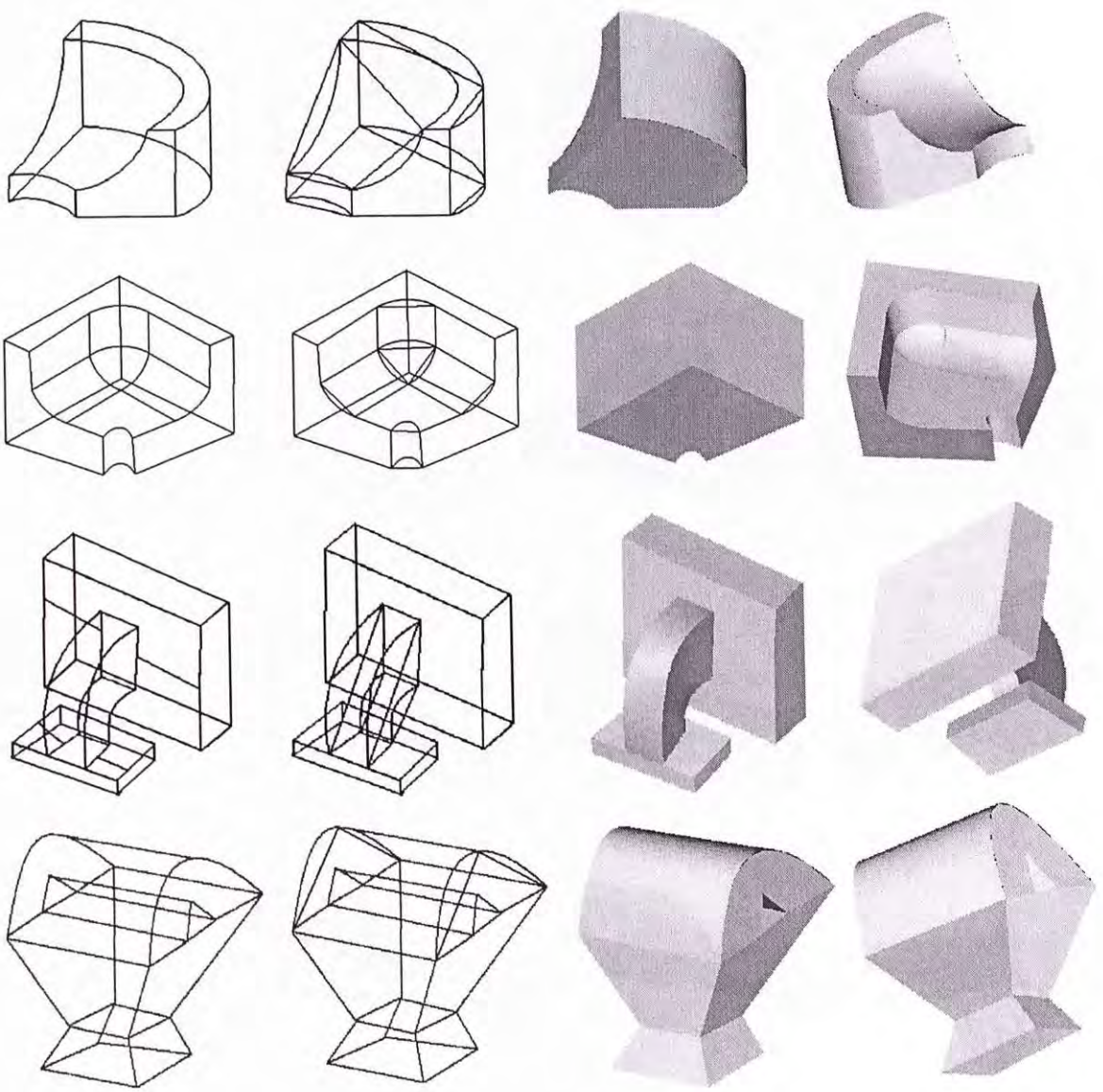


Figure 4.1: A set of line drawings, their transformed line drawings, and their reconstructed results, each shown from two viewpoints.

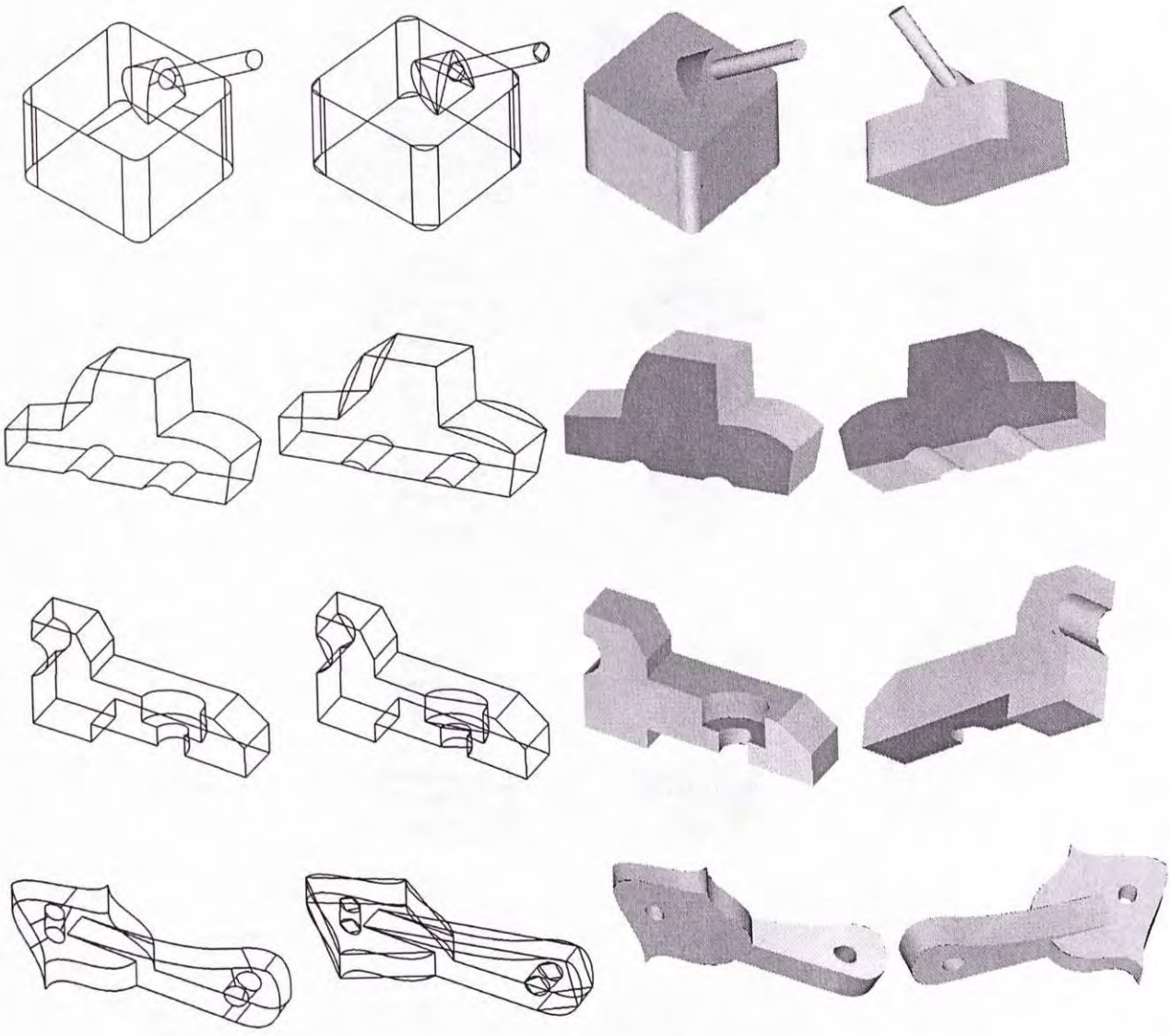


Figure 4.2: A set of line drawings, their transformed line drawings, and their reconstructed results, each shown from two viewpoints.

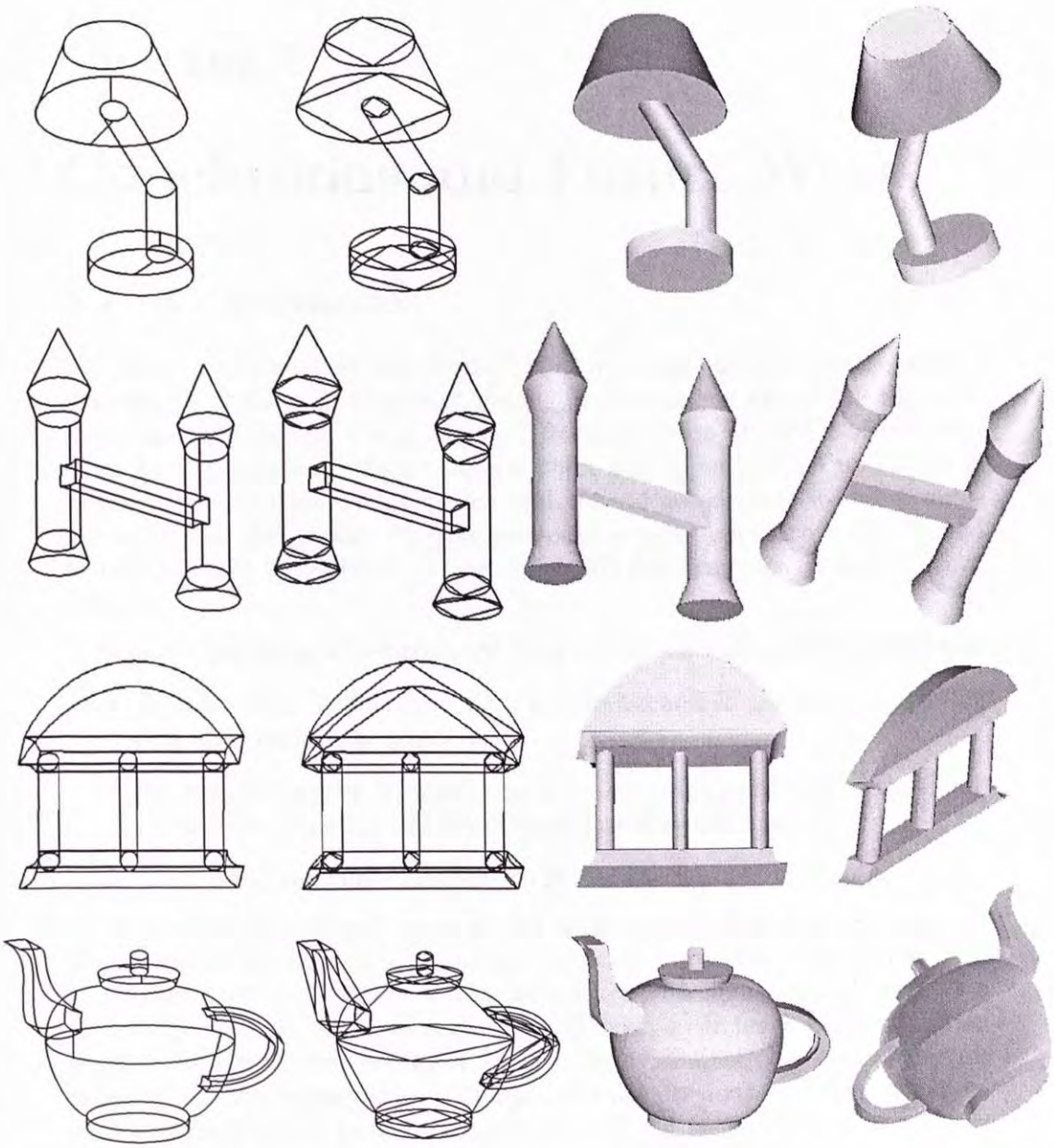


Figure 4.4: A set of line drawings, their transformed line drawings, and their reconstructed results, each shown from two viewpoints.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

3D object reconstruction from single 2D line drawings has been a long-standing challenging problem in computer vision. Most methods existing in the literature can only handle planar objects. Reconstruction of curved objects is a much harder problem. As we known, there still is no method published to tackle curved objects with complex and general geometrical structure automatically. In this thesis, we have proposed a novel approach to 3D curved manifold object reconstruction from single 2D line drawings. It includes four steps:

- Discriminating between curved faces and planar faces in the line drawing,
- Transforming the line drawing of a curved manifold into the line drawing of a (generalized) polyhedron,
- Reconstructing the 3D wireframe of the curved object based on the transformed line drawing and the original line drawing, and
- 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 results are quite satisfactory and in accordance with human visual perception of the objects. Our approach is also very efficient; it can finish a reconstruction within about two minutes after a line drawing is available, for each of the line drawings in the experiments. Compared with previous methods of curved object reconstruction from line drawings, ours can reconstruct more complex curved objects automatically.

5.2 Future work

Although the good results shown in our thesis testify the novel approach efficiently for 3D curved object reconstruction from single line drawings, there still

exist some cases where our method fails and needs to be improved. Moreover, many interesting and exciting directions in this area lie ahead. We describe these below in the hope that one or some of them may give inspiration to the future work.

- Our algorithm 1 in Section 3 cannot guarantee a global maximum because it is based on some randomized mechanisms. We find it is empirically efficient in searching the labeling combination and the problem of local maximum can usually be fixed by running the algorithm with multiple starts. Whether we can find a global algorithm for this problem is an interesting topic.

- Among all of the works on 3D reconstruction from line drawings, we first propose an complete new method to reconstruct 3D curved objects. However, we concentrate on the common objects containing 3D planar curves and our method may not be applicable to generalized complex 3D curved objects including space curves. In the future, tackling more free-formed curved objects is a significant work after its current stage. Moreover, we note that some results may not look so perfect in the experiments, such as the teapot. Work on finding algorithms of fine-tuning the results is also valuable in the future.

- In this thesis, we propose three new regularities (**Curve parallelism**, **Generalized face perpendicularity**, **Curve concurvity**) in Section 3.2.3, we believe that there exists more useful constrains on curved object reconstruction particularly. Thus, the future work can include developing more efficient regularities and using them in the optimization process. Moreover, nearly all the researchers on 3D reconstruction of line drawings explore only geometric and topological rules of line drawings, which are based on human's vision perception. Until now, no researchers try to incorporate machine learning schemes to handle this problem. Since the process that human's ability to understand things is a learning process which uses past experiences, attempting to reconstruct a 3D object from its 2D line drawing using a learning-based method is also attractive.

Bibliography

[1] S. Ablameyko, V. Bereishik, A. Gorelik, and S. Medvedev. 3D object reconstruction from engineering drawing projections. *Computing and 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] P. Bezier. Style, mathematics and nc. *Computer-Aided Design*, 22(9):524–526, 1990.

[6] E. Brown and P. Wang. 3D object recovery from 2D images: A new approach. *Proc. SPIE'96, Robotics and Computer Vision*, 2904:138–145, 1996.

[7] 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.

[8] 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*, 30(3):507–517, 2008.

[9] 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.

[10] Y. Chen, J. Liu, and X. Tang. A divide-and-conquer approach to 3D object reconstruction from line drawings. *IEEE Proc. Int'l Conf. Computer Vision*, 2007.

[11] A. Cicek and M. Gulesin. Reconstruction of 3D models from 2D orthographic views using solid extrusion and revolution. *Journal of Materials Processing Technology*, 152(3):291–298, 2004.

[12] M. Clowes. On seeing things. *Artificial Intelligence*, 2:79–116, 1971.

[13] P. Company, M. Contero, J. Conesa, and A. Piquer. An optimisation-based reconstruction engine for 3D modeling by sketching. *Computers and Graphics*, 28:955–979, 2004.

[14] P. Company, A. Piquer, M. Contero, and F. Naya. A survey on geometrical reconstruction as a core technology to sketch-based modeling. *Computer and Graphics*, 29(6):892–904, 2005.

[15] M. Cooper. The interpretations of line drawings with contrast failure and shadows. *Int'l Journal of Computer Vision*, 43(2):75–97, 2001.

[16] M. Cooper. Wireframe projections: Physicl realisability of curved objects and unambiguous reconstruction of simple polyhedra. *Int'l Journal of Computer Vision*, 64(1):69–88, 2005.

[17] M. Cooper. Constraints between distant lines in the labelling of line drawings of polyhedral scenes. *Int'l Journal of Computer Vision*, 73(2):195–212, 2007.

[18] M. Cooper. A rich discrete labeling scheme for line drawings of curved objects. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(4):741–745, 2008.

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

[20] 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.

[21] J. Dimri and B. Curumoorthy. Handling sectional views in volume-based approach to automatically construct 3D solid from 2D views. *Computer Aided Design*, 37:485–495, 2005.

[22] R. Goldman. *Pyramid Algorithms: A Dynamic programming approach to Curves and Surfaces for Geometric Modeling.* USA: Morgan Kaufmann publishers of Elsevier Science Press Inc., 2003.

[23] R. Haralick and L. Shapira. The consistent labeling problem: Part 1. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1(2):173–184, 1979.

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

[25] D. Huffman. Impossible objects as nonsense sentences. *Machine Intelligence*, 6:295–323, 1971.

[26] T. Kanade. Recovery of the three-dimensional shape of an object from a single-view. *Artificial Intelligence*, 17:409–460, 1981.

[27] M. Kuo. Reconstruction of quadric surface solid from three-view engineering drawings. *Computer-Aided Design*, 30(7):517–527, 1998.

[28] N. Langrana, Y. Chen, and A. Das. Feature identification from vectorized mechanical drawings. *Computer Vision and Image Understanding*, 68(2):127–145, 1997.

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

[30] 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.

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

[32] 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.

[33] 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.

[34] 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*, 30(2):315–327, 2008.

[35] 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*, 23(10):1106–1119, 2001.

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

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

[38] J. Liu and X.Tang. Efficient search of faces from complex line drawings. *IEEE. Proc. Computer Vision and Pattern Recognition*, 2:791–796, 2004.

[39] S. Liu, S. Hu, J. Sun, and C. Tai. *A Matrix-Based Approach to Reconstruction of 3D Objects from Three Orthographic Views.* IEEE Computer Society, Washington, DC, USA, 2000.

[40] D. Lysak. Interpretation of engineering drawings of polyhedral and non-polyhedral objects from orthographic projections. *PhD Thesis, Dept. of Electrical & Computer Engineering, The Pennsylvania State University*, 1991.

[41] J. Malik. Interpreting line drawings of curved objects. *Int'l Journal of Computer Vision*, 1(1):73–103, 1987.

[42] T. Marill. Emulating the human interpretation of line-drawings as three-dimensional objects. *Int'l Journal of Computer Vision*, 6(2):147–161, 1991.

[43] T. Marill. Why do we see three-dimensional objects? *A.I.Memo*, 1992.

[44] M. Masry, D. Kang, and H. Lipson. A freehand sketching interface for progressive construction of 3d objects. *Computer and Graphics*, 29(4):563–575, 2005.

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

[46] T. Pastva. Bezier curve fitting. *Master's Thesis, Naval Postgraduate School*, 1998.

[47] 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.

[48] L. Ros and F. Thomas. Overcoming superstrictness in line drawing interpretation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(4):456–466, 2002.

[49] A. Shesh and B. Chen. Smartpaper: An interactive and user friendly sketching system. *Proc. Eurograph*, 2004.

[50] H. Shimodaira. A shape-from-shading method of polyhedral objects using prior information. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(4):612–624, 2006.

[51] K. Shoji, K. Kato, and F. Toyama. 3D interpretation of single line drawings based on entropy minimization principle. *IEEE. Proc. Computer Vision and Pattern Recognition*, 2:90–95, 2001.

[52] 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*, 18:1000–1012, 1996.

[53] P. Sturm and S. Maybank. A method for interactive 3D reconstruction of piecewise planar objects from single images. *British Machine Vision Conference*, pages 265–274, 1999.

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

[55] K. Sugihara. An algebraic approach to shape-from-image problem. *Artificial Intelligence*, 23:59–95, 1984.

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

[57] T. Syeda-Mahmood. Indexing of technical line drawing databases. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(8):737–751, 1999.

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

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

[60] P. Varley and R. Martin. A system for constructing boundary representation solid models from a two-dimensional sketch. *IEEE Proc. Geometric Modeling and Processing*, pages 13–30, 2000.

[61] P. Varley and R. Martin. The junction catalogue for labelling line drawings of polyhedra with tetrahedral vertices. *Int.J. of Shape Modeling*, 7(1):23–44, 2001.

[62] P. Varley and R. Martin. Estimating depth from line drawings. *Proc. 7th ACM Symposium on Solid Modeling and Application*, pages 180–191, 2002.

[63] P. Varley, R. Martin, and H. Suzuki. Frontal geometry from sketches of engineering objects: Is line labelling necessary? *Computer Aided Design*, 37:1285–1307.

[64] P. Varley, Y. Takahashi, J. Mitani, and H. Suzuki. A two-stage approach for interpreting line drawings of curved objects. *EuroGraphics Workshop on Sketch-Based Input*, pages 105–108, 2004.

[65] 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.

[66] D. Waltz. Understanding line drawings of scenes with shadows. pages 19–91, 1975.

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

[68] M. Yu, I. Atmosukarto, W. K. Leow, Z. Huang, and R. Xu. 3D model retrieval with morphing-based geometric and topological feature maps. *IEEE. Proc. Computer Vision and Pattern Recognition*, 2:656–661, 2003.