Parameter Optimization and Learning for 3D Object Reconstruction from Line Drawings

DU, Hao

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

The Chinese University of Hong Kong
July 2010

Abstract

The visual system of human beings can easily interpret 2D line drawings and perceive their 3D models. In order to emulate this visual ability by a computer, many algorithms have been proposed in the literature. The optimization-based algorithms are the most successful and popular methods in this field. These algorithms derive a 3D configuration from its 2D line drawing by minimizing an objective function, whose input variables are the missing depths of the vertices in the line drawing. The objective function is formulated as a weighted sum of several image regularities, each of which describes a rule that the optimal 3D configuration should satisfy as much as possible. In this way, the algorithms seek a reconstruction that best satisfies the regularities.

Although the state of the art optimization-based 3D reconstruction algorithms can reconstruct a wide range of 3D objects, there is a serous problem with these algorithms, in which a large number of free parameters are set without optimization, causing distorted 3D objects often. In this thesis, we propose two approaches to solve the parameter setting and tuning problems. Firstly, the values of different image regularities span in very different ranges and their varying patterns during the optimization procedure are not well correlated, thus the traditional fixed regularity weights in the objective function would cause the problem that only the large value regularities dominate the optimization direction. We propose an adaptive parameter-setting strategy to handle this different regularity range problem. This adaptive strategy can be understood as a proper normalization method for the image regularities at each search step. It can also be seen as making the final objective function as a weighted sum of improvement ratios on the image regularities but not a weighted sum of absolute regularity values anymore. The experimental results show that the adaptive parameter-setting strategy brings dramatic improvement on the 3D reconstruction results.

Secondly, the weights of the image regularities in the objective function are traditionally set with heuristics and trials. However, when the number of regularities becomes large it is very difficult to set the weights appropriately. Improper weights may lead to less plausible or even unacceptable reconstruction results. We propose a parameter-learning framework to learn the best regularity weights. In the learning framework, a large 3D object database is constructed to provide the ground truth 3D objects for the training and testing datasets. A reconstruction error measure is defined to evaluate different weights' fitness. We employ an evolutionary algorithm to search for the best regularity weights in the large search space. The experimental results show that the proposed parameter-learning framework can effectively find better regularity weights which produce significantly better reconstruction results than the previous manually-set weights do.

摘要

人類視覺系統可以容易地理解二維線畫圖並感知它們所表示的三維模型。為了使計算機模擬這個智能能力，文獻中已提出了許多不同的算法。其中基於優化的三維物體重建算法在此領域中取得了很大成功且普及率最高。這種算法通過最小化一個目標方程來從二維線畫圖中得到它的三維結構，其目標方程的自變量即是線畫圖中各節點所缺失的深度坐標。此目標方程的表達形式是一個由許多圖像規則構成的加權和，其中每個圖像規則規定一個最優三維結構所應符合的條件。通過這種方式，此類算法尋找那個可以最好地滿足所有圖像規則的三維重建結果。

雖然目前最好的基於優化的三維重建算法可以重建範圍很廣的三維物體，這類算法仍有一個嚴重的參數問題。它的大量自由參數的設置方法沒有被優化，這通常會導致算法重建出歪斜扭曲的三維物體。在本論文中我們提出兩個方法來分別解決此類算法的參數設定和調試問題。首先，不同圖像規則的取值範圍差別很大，而且他們的值在優化過程中的變化趨勢也沒有很好的相關性，由此傳統的在目標方程中使用固定的圖像規則權重的方式會導致在優化過程中那些取值大的圖像規則主導了優化進行的方向。我們提出一種自適應的參數設定策略來處理這個圖像規則取值範圍不同的問題。這種自適應的策略可以被理解成在每一步優化中對圖像規則進行合適的歸一化；它也可以被看成是把目標方程的形式改變成為圖像規則取值的提高比率的加權和，而不再是圖像規則的絕對取值構成的加權和。實驗結果表明這種自適應的參數設定策略大幅度提高了基於優化的三維重建算法的重建結果質量。

第二，傳統上調試目標方程中圖像規則的權重都是用啟發式的方式或者多次試驗的方法。然而，當所使用的圖像規則的數目變得很大的時候，用這種方法很難調試出合適的圖像規則權重組合。不適當的權重會導致低質的甚至是不能接受的三維重建結果。我們提出一個參數學習的框架來學習出最優的圖像規則權重組合。在此學習框架中，一個三維物體數據庫被創建出來從而為訓練數據集和測試數據集提供標準真實的三維物體。一種三維重建結果的誤差測算方法也被定義出以評價不同權重組合的合適程度。我們使用遺傳算法在這個很大的權重搜索空間中尋找最優的圖像規則權重組合。實驗結果表明我們提出的參數學習框架可以有效地尋找到更好的圖像規則權重組合，這些通過學習得到的權重組合與之前手工設定的權重組合相比可以產生出明顯更理想的三維重建結果。

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. Xiaoou Tang and my co-supervisor Prof. Jianzhuang Liu, who have been guiding and supporting me all along the way during the past two years and have provided me with such excellent environment and precious chances to carry out frontier research in the computer vision field. It was my great honor to work under such prestigious researchers and nice teachers. They have taught me the way of thinking of research ideas, the detailed approaches to research problems, and the macro research trends. They have also demonstrated me the way to live a successful and happy life. These all will always be invaluable fortunes in my life.

I am also grateful to all the members of our Multimedia Lab, Boqing Gong, Chunjing Xu, Deli Zhao, Mo Chen, Ming Liu, Kaiming He, Kui Jia, Ke Liu, Tianfan Xue, Wei Zhang (Da), Wei Zhang (Xiao), Xiaotian He, Yingze Wang, Yueming Wang, Yichen Yang, Zhimin Cao, Zhenguo Li, etc. I have learned a lot from all these great friends. Their perspectives and insights to different research problems and life have greatly inspired me. I am proud and lucky to spend the past happy two years in such a harmonious family with these wonderful friends.

Finally, I would like to thank my parents for their constant support, caring and love, and my uncle who gave me especial help when I met difficulty in the past two years, and all my family members because they are my source of strength and power.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 3D Reconstruction from 2D Line Drawings and its Applications

3D object reconstruction from 2D representations is a classic and important research topic in computer vision community. It has many important applications including architectural design, 3D game, animation, virtual reality, etc. The 3D reconstruction problem is difficult because the depth information is lost when a 3D object is projected onto a 2D image plane. However, human beings can easily interpret the configuration of a 3D object shown on a 2D image. To enable the computers to acquire this intelligent ability, many algorithms have been proposed for different application scenarios, such as 3D reconstruction from video sequences, from single view or multiple view images, from engineering drawings of multiple views, etc.

In this thesis, we focus on the problem of 3D object reconstruction from a single 2D line drawing. A 2D line drawing is defined as the parallel or nearly-parallel projection of a 3D object in a generic view where all the edges and vertices of the object are visible, and the line drawing can be represented by a single edge-vertex graph. Fig. 1.1 shows an example of 3D object reconstruction from a single 2D line drawing. Fig. 1.1 (a) is the input line

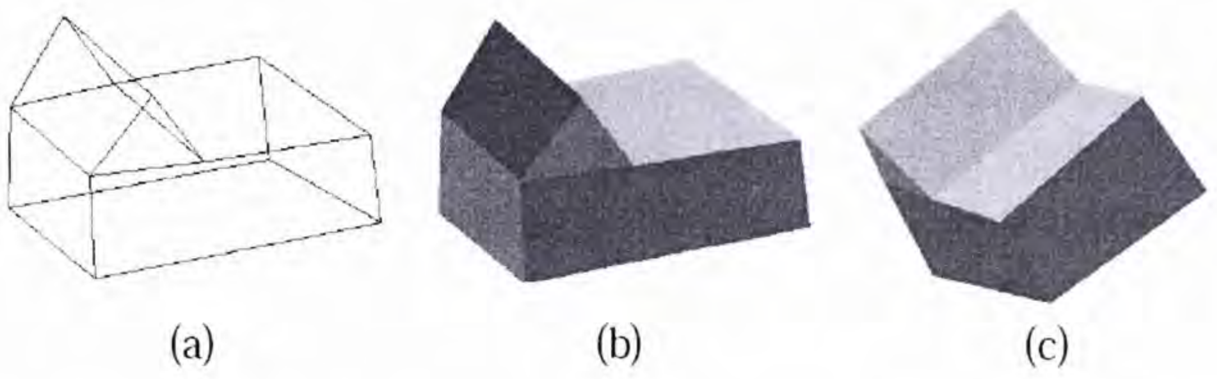

Figure 1.1: 3D object reconstruction from a single 2D line drawing

drawing. Fig. 1.1 (b) and (c) are the reconstructed 3D object viewed from two different viewing angles. It is straightforward and easy to represent a 3D wireframe object in a 2D line drawing. Thus, the automatic reconstruction of accurate 3D objects from their 2D line drawings would be desired and helpful for various 3D reconstruction applications such as architectural design, landscape plan, 3D modeling in games, animations, films and etc. In general, these applications can be regarded as the work done by a computer-aided design (CAD) system. Current sophisticated CAD tools still cannot automatically convert a line drawing into a 3D object, which prevent the users, especially the conceptual designers, from freely and conveniently expressing their design ideas.

Many researchers have noticed the importance of 3D reconstruction problem from 2D line drawings and a number of publications have been devoted into this research both in computer vision and in CAD and graphics literature [7], [24], [27], [29], [30], [32], [31], [28], [36], [37], [41], [44], [45], [47], [49], [2], [3], [10], [11], [54], [55], [51]. The specific application scenarios of these works include: (1) flexible sketching user interface in CAD systems for conceptual designers who tend to prefer using pencil and paper rather than mouse and keyboard [3], [27], [30], [42]; (2) providing rich databases for object recognition systems and reverse engineering algorithms [2], [3], [11], [50]; (3)

automatic conversion of industrial wireframe models to solid models [2], [19]; (4) interactive generation of 3D models from real images [16], [28], [51], [46]; (5) friendly user interface in 3D object retrieval systems [5], [39] [61]. The line drawings used in these applications may be acquired from the sketches on paper, drawing on a digitizer tablet, drawing on the screen with a mouse or existing industrial wireframe models. To extract the edge-vertex graph representation of a line drawing from a scanned image, some procedures may be required such as binarization of the image, thinning of the lines and analysis of the connectivity.

1.2 Algorithmic Development of 3D Reconstruction from 2D Line Drawings

Since the seventies of last century, researchers have been exploring the approaches of interpreting 2D line drawings as 3D objects. Significant improvements of the 3D reconstruction algorithms have been seen in recent years with the development of optimization-based 3D reconstruction algorithms which can reconstruct more complex and larger-scale 3D objects from a single 2D line drawing. In this thesis we focus on enhancing the performance of the state of the art optimization-based 3D reconstruction algorithms by optimizing its parameter setting and tuning strategies. Our work is important because the optimization-based algorithms have a large number of free parameters and it is difficult (almost impossible) to set these parameters optimally just with experience or heuristics. Our aim is to make the 3D reconstruction results not only topologically correct but also aesthetically reasonable. In order to provide a comprehensive picture of related works, in the following three subsections we will present the algorithmic developments of 3D reconstruction from 2D line drawings.

1.2.1 Line Labeling and Realization Problem

A large amount of early works in computer vision are about line labeling and 3D reconstruction based on a labeled drawing [9], [12], [13], [14], [15], [18], [20], [21], [35], [47], [49], [48], [57], [53]. Line labeling targets at the problem of finding a set of consistent labels from a line drawing and to provide a qualitative description of the scene by classifying the parts of a line drawing as the projection of concave, convex or contour edges. Although line labeling explores 3D information, 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. [20] and Clowes [9] independently first described a scheme for labeling line drawings in 1971. They targeted at the case where all faces are planar, that is, a "polyhedral world", so there were only four possible labels $\{+, -, \vee, \wedge\}$ in their works. It was 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 1975, Waltz [57] presented a filtering algorithm which has a 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 compatible labeling of its neighbor junctions. 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 a tree searching with backtracking. Most of the line labeling algorithms are designed for polyhedra without hidden edges. Recently, Cooper published a series of works which extended the line labeling research to handling wireframes objects with hidden lines visible as well as curved objects [13], [14], [15]. However, a limitation of line labeling is that multiple consistent labeling solutions for one line drawing are possible [43].

Another area of the 3D reconstruction 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 [47], [49], [48], [52], [41]. 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 line drawings which are either extracted from an image or drawn by a person, it is usually impossible to find a 3D object whose projection is exactly the same as the line drawing. Small deviations of some vertices from the precise 2D projection may cause the corresponding 3D face to be nonplanar. Thus there is no theoretically exact solution to the 3D reconstruction problem. However, human beings usually can easily understand what an imperfect line drawing represents. This problem of the realization approaches is called superstrictness. Therefore, the limitations of these methods are that such a formulation is superstrict and thus not robust; realizability can be efficiently checked only when a legal labeling is available.

1.2.2 3D Reconstruction from Multiple Line Drawings

Works of 3D reconstruction from multiple line drawings try to reconstruct a 3D CAD model from its multiple (three, in general) orthographic projections [1], [60], [22], [33], [8], [17], [23], [25], [34]. More information can be used from three orthographic views than from a single projected view for the reconstruction task, so this work is easier compared with 3D reconstruction from a single line drawing. Traditionally, engineering objects are represented by three orthographic views: front, top and side views. Liu et al. [33] used matrices to represent conic faces for the reconstruction of different objects such as planar, cylindrical and conical faces. He also gave the proof that minimum number of views required to represent conics are three. The approach in [8] 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. [17] introduced a novel technique of reconstruction from x-sectional views. Handling of sectional views was also discussed by Wesley [60] but their approach is limited to full sectional views only. Technique of Dimri [17] took into account full sectional, half sectional, offset sectional. As these approaches deviate much from our main theme in this thesis, we will not give more detailed reviews on this stream of research.

1.2.3 3D Reconstruction from a Single Line Drawing

The state of the art algorithm of 3D reconstruction from a single line drawing is the optimization-based algorithm which was firstly proposed by Marill [36] in 1991. Since then, many improvements on this algorithm have been proposed [27], [7], [6], [28], [58]. Our work in this thesis is trying to enhance the performance of the optimization-based algorithms from the parameter setting and tuning perspective, therefore, in the following we will give a detailed introduction of the framework of the optimization-based 3D reconstruction algorithms.

In general, the optimization-based algorithms reconstruct a 3D object from a single 2D line drawing in two steps. The first step is face identification and the second step is 3D geometry reconstruction. We will use the planar object reconstruction as an example to illustrate the two steps of the algorithms. For curved object reconstruction the general framework is the same.

Face Identification From A Line Drawing

Face identification from a line drawing is the first step of 3D reconstruction. An 3D object consists of several faces. If the face configuration of an object

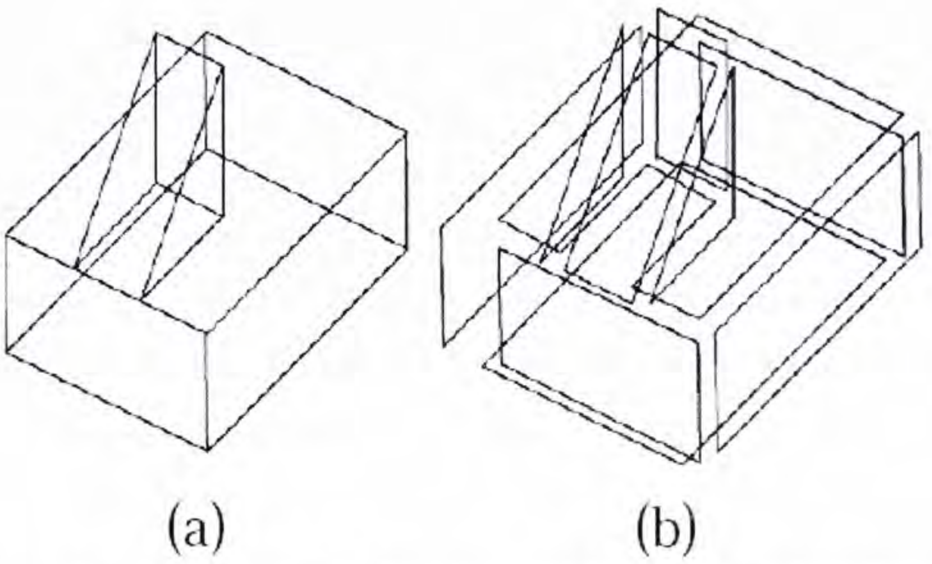

Figure 1.2: (a) Input line drawing. (b) Faces of the line drawing

is known before the reconstruction of its 3D geometry, the complexity of the reconstruction will be reduced significantly. Fig. 1.2 shows the faces of a line drawing.

In general, a face is a closed cycle in the line drawing's edge-vertex graph. There are many cycles in a line drawing but only a small subset of them represent the 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 [30], [3], [2], [24], [45], [32], [29], [31], [26].

A distinct decomposition method for extracting face topologies from wire-frame models was proposed by Agarwal and Waggenspack [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 [30]. 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 [27] and [29] 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 [27] 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 [29] 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. [32] and [31] 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 [27], [29].

In this thesis we use planar manifold objects in training and testing datasets. Among these face identification techniques, the work in [30] is most suitable for us, because the previous approaches could not handle the visible hidden lines very well. 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. [30] proposed the 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. We implemented this method to identify faces for our 3D reconstruction system.

3D Geometry Reconstruction

To fully reconstruct the 3D object after face identification, the next step of the optimization-based algorithms is to formulate the problem as an optimization problem and to minimize an objective energy function.

Marill [36] firstly presented the optimization-based approach which was 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 [4] proposed to minimize the standard deviation of the segment magnitudes (MSDSM) in the recovered planar object, and Shoji et al. [44] 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. Later, some researchers extended the criterions following this idea and incorporated more heuristic regularities in the reconstruction process. Leclerc and Fischler et al. [24] 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 [40] and [56] concentrated on the reconstruction of symmetric polyhedra by developing a regularity of model symmetry. Lipson and Shpitalni [27] 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. When the reconstruction process begins, the given 2D edge-vertex graph is analyzed and image regularities are identified. A 3D configuration can be represented by a vector Z containing the z coordinates of the vertices. The objective 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 weighting vector W. The final objective function to be optimized takes the form

$$F(Z) = W^T \alpha(Z), \tag{1.1}$$

where α is the vector containing all the constraints including $\alpha_{planarity}$, $\alpha_{parallel}$, $\alpha_{vertical}$, $\alpha_{isosometry}$, $\alpha_{cornerskewedorthogonality}$, etc. Here we note that the global balancing coefficient vector W is actually a free parameter. Its dimension is the number of different image regularities. When a large number of image regularities is used, it is very difficult to set W properly just with

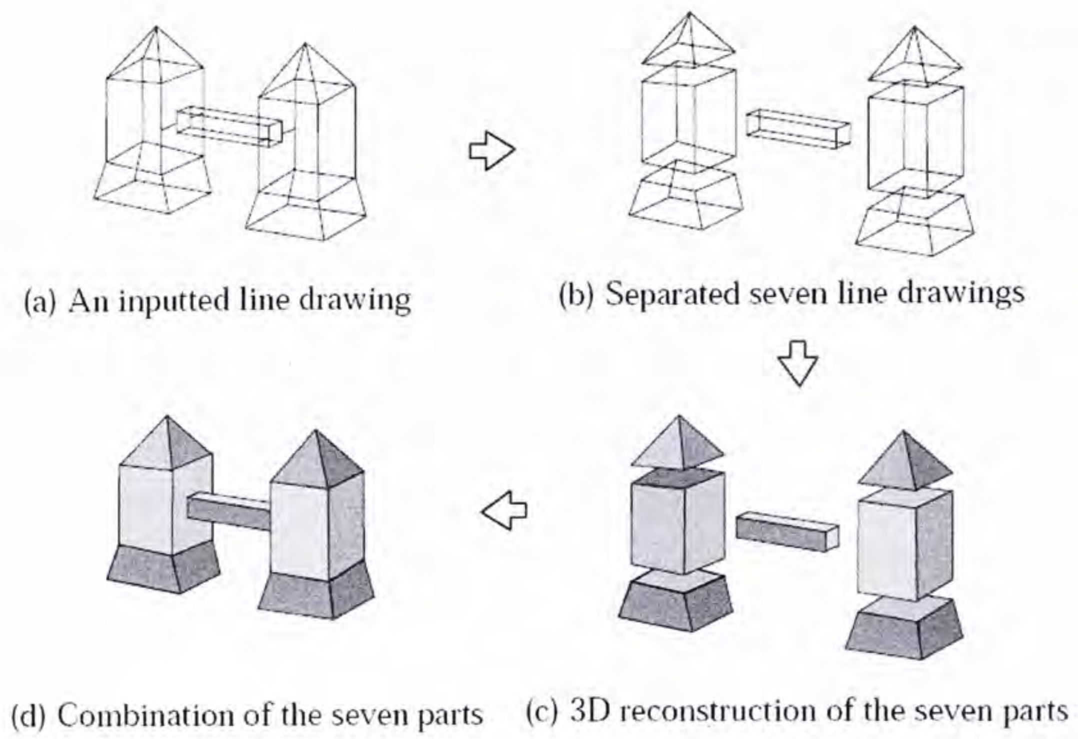
(a) An inputted line drawing
(b) Separated seven line drawings
(d) Combination of the seven parts
(c) 3D reconstruction of the seven parts

Figure 1.3: Divide-and-conquer strategy

experience and heuristics. The work in this thesis will tackle this problem.

To reconstruct more complex objects, Chen et al. [7] used a divide-and-conquer strategy as shown in Fig. 1.3. 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. And for curved object reconstruction, Wang [58] recently proposed an approach in which the problem is firstly converted into the planar object reconstruction problem and solved by the method in [7], and then the reconstruction result is converted back into the original curved form.

Through the above introduction of the algorithms' developments, we can see that even though the state of the art methods become more and more complicated so that they can deal with more complex or curved objects, they are still holding the core idea of the optimization-based algorithms which is to formulate the reconstruction process as an optimization problem and to minimize an objective energy function as expressed in (1.1). Therefore, all the methods must decide the value of the free parameter W and whether or not the resulted energy function in (1.1) is appropriate will significantly affect the final reconstruction result. This intrigues the research topic of this thesis.

1.3 Research Problems and Our Contributions

The optimization-based algorithms reconstruct 3D objects from a line drawing by minimizing an objective energy function as shown in (1.1). This function can be seen as a weighted sum of a set of image regularity terms. As it is not analytically tractable, the step-wise hill-climbing algorithms are usually used to find the optimization result. Each image regularity term in this function reflects how well the 3D configuration is consistent with a certain aspect of human being's perception of the 2D line drawing. For example, the face planarity term shows the degree of flatness of all the faces identified in the face identification step. Including this term in the objective function is appropriate because human beings tend to interpret all the identified faces as flat faces. By incorporating many such regularities into the final objective function and looking for the best 3D configuration which minimizes it, the algorithms can produce the desired reconstruction result which is consistent with human being's perception as much as possible. And because the algorithm is not seeking exact algebraic solutions but doing an optimization procedure, it has the advantage that the errors or inaccuracies in a hand-written line drawing can be tolerated, which is in accordance with human being's ability.

However, there are still problems with this optimization formulation. Firstly, the regularity weights W are fixed in the objective function during the optimization procedure, while the different image regularities' values can vary dramatically during the process with different value ranges. For example, during the optimization procedure a face cycle in 3D can produce a face planarity term with a magnitude of 10000 but an un-parallel line pair can only produce a parallel-line term with a magnitude of 100. At the final steps of optimization, both of these two terms will produce values of nearly zero. And the varying patterns among different regularities are not well correlated. Thus, it is quite possible that at certain steps of hill-climbing only a subset of the image regularities, which are producing larger values, dominates the optimization direction. Therefore, with fixed regularity weights the algorithm cannot fully make use of all the image regularities and the hill-climbing procedure becomes more prone to go to the local optimal points. The second problem with the optimization-based algorithms is its large number of free parameters. Since Lipson and Shpitalni [27], researchers have developed more than dozen image regularities to mimic human being's interpretation of line drawings. In the objective function, each dimension of the free parameter W is a weight assigned to a specific image regularity. Thus, the number of free parameters in the objective function equals to the number of image regularities. Assume that we want to use ten image regularities to do the reconstruction and the weight of each regularity is an integer within $[1, 100]$, then the number of feasible weight assignments is $100^{10} = 1 \times 10^{20}$. This huge number means that it is almost impossible to heuristically find the optimal weight assignment when we want to use many image regularities. Because of this parameter problem, in practice the 3D reconstruction systems usually just use a small number (three to four) of the image regularities and the parameters (regularity weights) are arbitrarily set with trials and heuristics.

In this thesis, we propose approaches to solve the above two problems with the optimization-based 3D reconstruction algorithms. We firstly present an adaptive parameter-setting strategy to deal with the problem of different image regularity magnitudes. In the proposed adaptive parameter-setting strategy, the weights of the image regularities are no longer fixed during the optimization procedure but varying according to the values of the corresponding image regularity terms. Secondly, we build a 3D object database and with the database we develop a parameter-learning framework to learn the optimal weight assignments for a large number of image regularities. Our experimental results show that both the new adaptive parameter-setting strategy and the parameter-learning framework can effectively improve the performance of the state of the art optimization-based 3D reconstruction algorithms.

Chapter 2

Adaptive Parameter Setting

The free parameters of the optimization-based 3D reconstruction algorithms are the weights of the image regularities. Traditionally, the regularity weights are fixed real values. However, as different image regularities' values span in very different ranges and their varying patterns during the optimization procedure are not well correlated, the fixed-weights strategy can bring the problem that only the large value regularities dominate the optimization direction in certain steps of the hill-climbing optimization procedure. In this chapter, we will present an adaptive strategy to set the image regularity weights so that all the image regularities can be fully utilized in the hill-climbing optimization procedure and thus the problem mentioned above is overcome. Before introduce the strategy, we firstly introduce twelve image regularities which are used in 3D reconstruction from 2D line drawings.

2.1 Regularities in Optimization-Based 3D Reconstruction

In [27], Lipson and Shpitalni introduced a large number of image regularities into the optimization-based 3D reconstruction algorithms. Our parameter setting and tuning work in this thesis is mainly based on these image regularities. In this section, most of the following regularities and explanations come from this paper. Image regularities are special geometrical relationships between individual entities or within groups of entities in a line drawing. Its heuristic rule is that the image regularities do not appear in the line drawing accidentally, but rather correspond to some real geometrical regularities existing in the 3D object. An example of a typical regularity is parallelism. The heuristic rule for the parallelism regularity is that if two lines are parallel in the sketch plane, they probably represent parallel lines in the 3D object. This heuristic rule has a sound statistical basis: two un-parallel lines in space will appear parallel only when viewed from a very limited scope of viewpoints. Parallel lines in space, however, will appear parallel when viewed from any viewpoint. Hence, if two lines are parallel in the sketch plane, it is more likely that they represent true spatially parallel lines. Most of the following image regularities are based on similar grounds. The notion of image regularities is so deeply rooted in the human visual system that an image failing to comply with them often perplexes the viewer. An excellent example is found in M. C. Escher's puzzling drawings, where some of the scenes contain parallel lines that do not correspond to parallel lines in the three dimensional scene.

Because there are inevitable errors in a hand-written line drawing, the image regularities should be able to handle small variations. Take the parallelism of two lines as an example. A cutting threshold of the angle between two lines can be used to detect the parallelism, however this method is too strict and not consistent at the values around the threshold. To avoid this threshold problem, a continuous compliance factor $\mu(a)$ is defined for the image regularities. Here for parallelism, the a is the angular difference between the two lines in question. The compliance factor μ ranges from 1.0 for exact parallelism ($a = 0$) and descends to zero like a standard normal distribution curve with the deviation being the threshold value,as a approaches to $90°$. For each pair of lines in 3D, their line-parallelism term's value will be determined by both their layout in the 2D line drawing and their 3D positions. Intuitively, it could be said that "the more the lines are parallel in the 2D sketch plane, the more they are required to be parallel in 3D space." Thus, by avoiding a clear cut threshold, a small change in the angle of a line would not cause a step difference in interpreting parallelism. The general formulation of the function $\mu(x)$is given by:

$$\mu_{a,b}(x) = e^{-((x-a)/b)^2}, \tag{2.1}$$

where x represents the value to be checked, a is a nominal value (e.g., $0°$ for parallelism and $90°$ for perpendicularity), and b is a reasonable deviation (e.g., $7°$ for parallelism).

Here $\mu(x)$ has been defined so that it evaluates to 1.0 when $x = a$ exactly, and degenerates to 0.0 like a standard distribution curve with $\sigma = b$ as x retreats from a. For practical purposes, equation (2.2) has been modified to eliminate values close to zero that may otherwise be weak regularities. That is:

$$\mu_{a,b}(x) = max[0, 1.1 \cdot e^{-((x-a)/b)^2} - 0.1], \tag{2.2}$$

The inclusion of the consistency principle is a significant contribution to the robustness of the interpretation of imperfect 2D line drawings.

Notations

The following notation is used in the image regularity expressions:

α represents the criteria value. These criteria are summed up and used as a minimization target.

w represents the weight inside an image regularity term which is assigned to certain entities. Its value is determined by the entities' congruence level with the associated regularity in the 2D line drawing.

v, v' are vectors that respectively represent a vertex belonging to a 3D object and its 2D projection in the sketch plane.

l, l' are unit vectors that respectively represent a direction in 3D and in the sketch plane.

The following is a list of image regularities. Each regularity includes an observable geometrical relationship in 2D and the associated configuration presumed in 3D. The mathematical terms used to evaluate the regularities are given.

2.1.1 Face Planarity

As we are considering objects with planar faces only, each face cycle identified in the 2D line drawing should reflect a planar face in 3D. The evaluation of this condition is performed in two stages: first, the best-fit surface for the face cycle in 3D is found; then, the deviation of each vertex from that surface is computed, squared, and summed.

The best-fit plane is assumed to be in the form:

$$ax + by + cz + d = 0, \tag{2.3}$$

The plane coefficients a, b, c are computed by solving the linear system (2.4) using the given list of points $(x_i; y_i; z_i i = 1..n)$ lying on the plane, and arbitrarily assuming $d = 1$.

$$\sum \begin{pmatrix} x_i^2 & x_i y_i & x_i z_i \\ x_i y_i & y_i^2 & y_i z_i \\ x_i z_i & y_i z_i & z_i^2 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \sum \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}, \tag{2.4}$$

The coefficients are then normalized by having $\sqrt{a^2 + b^2 + c^2} = 1$ with d scaled appropriately and the deviation of a point from the plane taken as the absolute value $|ax_i + by_i + cz_i + d|$.

2.1.2 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

$$\begin{aligned} \alpha_{parallel} &= w_{1,2}[\cos^{-1}(\hat{l_1} \cdot \hat{l_2})]^2 \\ w_{1,2} &= \mu_{0^\circ,7^\circ}(\cos^{-1}(\hat{l'_1} \cdot \hat{l'_2})), \end{aligned} \tag{2.5}$$

where $\hat{l_1}$ and $\hat{l_2}$ are the unit direction vectors of the first and second lines, respectively. The weight $w_{1,2}$ given to a specific pair of lines for this regularity depends on how the two lines are parallel in the sketch plane.

2.1.3 Line Verticality

A line that is vertical in the sketch plane (parallel to the y-axis of the drawing page) is "vertical in space," i.e., its two endpoints have similar depth z-coordinates. The term used to evaluate the verticality of a line is

$$\begin{aligned} \alpha_{vertical} &= w_l[\cos^{-1}(\hat{l_y})]^2 \\ w_l &= \mu_{0^\circ,7^\circ}(\cos^{-1}(\hat{l'_y})), \end{aligned} \tag{2.6}$$

where $\hat{l_y}$ is the vertical component of the line's direction vector and $\hat{l'_y}$ is its vertical component in the sketch plane.

2.1.4 Isometry

Lengths of entities in the 3D model are uniformly proportional to their lengths in the sketch plane. This term's usefulness can be seen from the example that it can make sure that an common isometric cube's projection in 2D is not interpreted as a special long cube's projection viewed from a specific angle. This term accounts for non-uniformity corresponds to the standard deviation of scales as follows:

$$\begin{aligned} \alpha_{isometry} &= n \cdot \sigma^2(r_i = 1...N_e) \\ r_i &= \frac{length(entity_i)}{length'(entity_i)}, \end{aligned} \tag{2.7}$$

where n is the number of entities, r_i is the ratio between the current length of entity i in 3D and its length in the sketch plane, and α is the standard deviation of the series of r_i.

2.1.5 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, the test is based on the fact that the projection of an orthogonal corner spans at least 90°. A junction of three lines has eight variants, created by flipping the direction of each line and considering the eight resulting permutations. Fig. 2.1(a) shows some three-line junctions which may appear to form orthogonal corners. Fig. 2.1(b)'s junctions are not. Fig. 2.1(c) shows all the eight possible variants of a three-line junction. Every variant is tested in this regularity. For each variant, three lines exist $l'_{i=1...3}$, forming three pairs between themselves, $l'_{i=1,2}, l'_{i=2,3}, l'_{i=3,1}$. Each line is described by a 2D unit direction vector l'_i in the sketch plane, pointing from the junction outwards. If a junction variant spans less than 90° (i.e., is not a projection of an orthogonal corner), all of the three dot-products of its direction-vector pairs will be positive. If a three-line junction is a projection of an orthogonal corner, all of its eight variants must span at least 90°. Thus, if any one of the eight variants appears to span less than 90° , (shows such an "all-positive" condition), the tested junction is unlikely to represent an orthogonal corner. Consequently, the term used to evaluate the corner orthogonality condition is

$$\begin{aligned} \alpha_{corner} &= w_{corner} \sum_{Pair=1}^{3} [\sin^{-1}(\hat{l_1} \cdot \hat{l_2})]^2 \\ w_{corner} &= \begin{cases} 1 & if \beta \leqq 0 \\ \mu_{0,0.1} & if \beta \geqq 0 \end{cases}, \ \beta = \max_{8variants} [\min_{3pairs} (\hat{l'_a} \cdot \hat{l'_b})], \end{aligned} \tag{2.8}$$

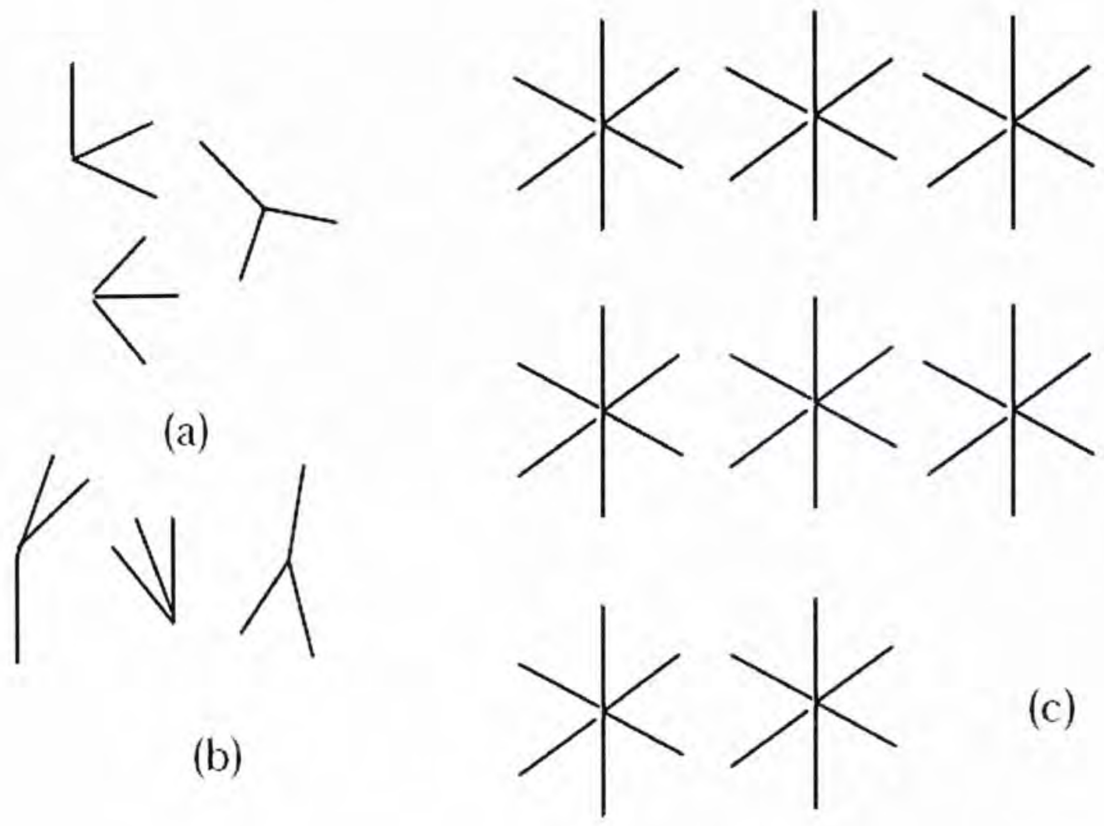

Figure 2.1: Corner orthogonality

2.1.6 Skewed Facial Orthogonality

A face contour that shows skewed orthogonality is probably orthogonal in space. If entities on the contour of a planar face join only at right angles, then the contour can be said to be orthogonal. If this contour is viewed from an arbitrary viewpoint, it will exhibit skewed orthogonality, as is illustrated in Fig. 2.2 (a). Faces or entity chains showing skewed orthogonality are easily detected by alternating their boundary lines between two main directions which correspond to the main axis directions of the original shape (see Fig. 2.2 (a)). The statistical behavior of the alternating values produced by multiplying the scalar-product and the cross-product of adjacent 2D lines in the sketch plane is used for detection. Consistent behavior is likely to represent skewed orthogonality. The amount by which the face is considered to have skewed orthogonality is represented by the value of the weighting coefficient

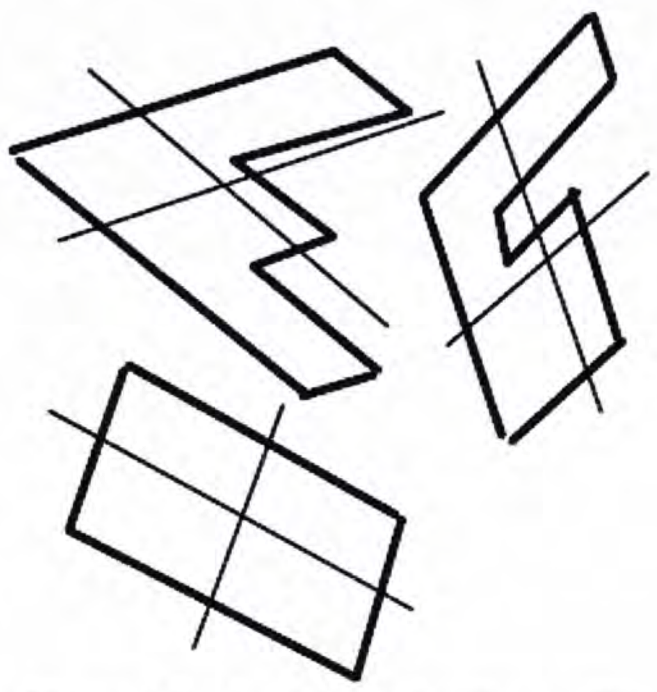
Figure 2.2: Skewed facial orthogonality

$w_{skewedorthogonality}$. The terms to evaluate the above for a face are

$$\alpha_{skewed\ orthogonality} = w_{skewed\ orthogonality} \sum_{i=1}^{n} [\sin^{-1}(\hat{l_i} \cdot \widehat{l_{i+1}})]^2$$
$$w_{skewed\ orthogonality} = \mu_{0,0.2}(\sigma(\beta_{i=1 \ldots n})),\ \beta = (-1)^i \cdot [\hat{l'_i} \cdot \widehat{l'_{i+1}}] \cdot [\hat{l_i} \times \widehat{l_{i+1}}], \tag{2.9}$$

where n represents the number of lines along the face contour.

2.1.7 Skewed Facial Symmetry

A face showing skewed symmetry in 2D denotes a truly symmetrical face in 3D. Algorithms for the detection of skewed symmetry have been the subject of extensive research. A simplification used here is that if skewed symmetry exists in a polygonal shape, its axis intersects the contour at two points, each either a vertex or midpoint of an entity. Assuming also that the number of entities on both sides of the symmetry axis in a truly symmetrical shape is equal, the number of possible symmetry-axis candidates is reduced significantly to n, where n is the number of vertices in the shape. Each possible candidate symmetry-axis passes through the vertices v_k and $v_{k+n/2}$, where $k = 1/2, 2/2, 3/2, ..., n/2$ and,

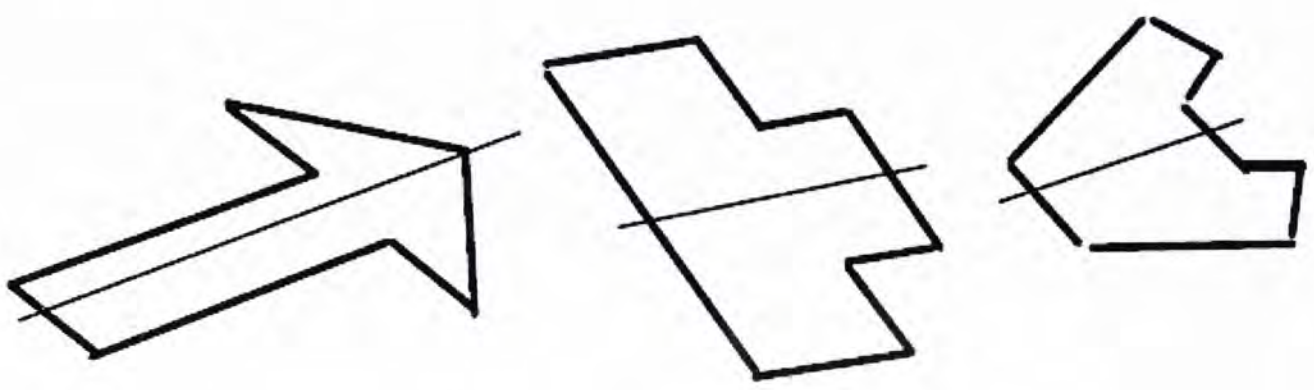

Figure 2.3: Skewed facial symmetry

for example, $v_{2\frac{1}{2}} = (v_2 + v_3)/2$.

The relationship between the vertices of the shape and the candidate symmetry-axis determines whether the axis can serve as a skewed symmetry axis. This relationship is represented, per axis k, by the weighting coefficient w_k. The maximal w_k determines the selected symmetry-axis if the face possesses the skewed symmetry characteristic at all.

$$\begin{gathered} w_{skewed\ symmetry} = \max_{k=1/2,2/2,3/2,\dots,n/2}[w_k] \\ w_k = \mu_{0,0.2}[\sigma_{k=1/2,2/2,3/2,\dots,n/2}(skew_i) + \sigma_{k=1/2,2/2,3/2,\dots,n/2}(sym_i)] \\ skew_i = [\tfrac{v'_k - v'_{k+n/2}}{||v'_k - v'_{k+n/2}||} \cdot \tfrac{v'_{k+i} - v'_{k-i}}{||v'_{k+i} - v'_{k-i}||}] \cdot [\tfrac{v'_k - v'_{k+n/2}}{||v'_k - v'_{k+n/2}||} \times \tfrac{v'_{k+i} - v'_{k-i}}{||v'_{k+i} - v'_{k-i}||}] \\ sym_i = \tfrac{dist\ of\ v_{k+i}\ from\ axis}{dist\ of\ v_{k-i}\ from\ axis} - 1, \end{gathered} \tag{2.10}$$

Note that the vertices v'_k are in the 2D sketch plane. Two conditions are required for skewed symmetry to occur. First, corresponding points must be evenly distanced from the symmetry axis, a condition denoted by the *sym* term above; second, lines stretched between corresponding points must form a consistent angle with the symmetry axis, a condition denoted by the *skew* term above. If skewed symmetry has been detected, the optimization term will be

$$\alpha_{skewed\ symmetry} = w_{skewed\ symmetry} \sum_{i=1}^{n/2} [\arcsin([\tfrac{v_{k+1} - v_{k-1}}{||v_{k+1} - v_{k-1}||}] \cdot [\tfrac{v_k - v_{k+n/2}}{||v_k - v_{k+n/2}||}])]^2, \tag{2.11}$$

where k denotes the axis that has been selected.

2.1.8 Line Orthogonality

All line pairs in a junction except those that are collinear are perpendicular in 3D. This statement does not represent a pure regularity in the sense that it does not depend entirely on the appearance of the entity in the image plane apart from the exception clause. For a junction, the regularity serves mainly to initially "inflate" the flat projection into 3D. The term used for this computation is:

$$\begin{aligned} \alpha_{line\ orthogonality} &= \sum_{i=1}^{n} w_i[\arcsin(\hat{l_1} \cdot \hat{l_2})]^2, \\ w_i &= \mu_{0^\circ,7^\circ}(\arccos(\hat{l_1'} \cdot \hat{l_2'})) \end{aligned} \tag{2.12}$$

where n is the number of non-collinear pairs of lines meeting at the junction.

2.1.9 Minimum Standard Deviation of Angles

All angles between all pairs of lines meeting at junctions must be similar (MSDA = Minimum Standard Deviation of Angles). The term used for this computation for the entire body is

$$\alpha_{MSDA} = n \cdot \sigma^2(\arccos(\hat{l_1} \cdot \hat{l_2})) \tag{2.13}$$

where $\hat{l_1}$ and $\hat{l_2}$ represent the unit direction vectors of all possible line pairs meeting at vertices of the object.

2.1.10 Face Perpendicularity

All adjacent faces must be perpendicular. Again, this criterion serves to initially "inflate" the flat projection to a convex shape in 3D space from which optimization is more easily achieved. The term used here is

$$\alpha_{face\ perpendicularity} = \sum_{i=1}^{n}[\arcsin(\hat{n_1} \cdot \hat{n_2}))]^2 \tag{2.14}$$

where $\hat{n_1}$ and $\hat{n_2}$ denote all possible combinations of normals of adjacent faces, and n is the number of such combinations.

2.1.11 Line Collinearity

Lines collinear in the sketch plane are collinear in space. The term used to denote this heuristic is

$$\begin{gathered}\alpha_{line\ collinearity} = \sum_{i=1}^{n} w_i \cdot \max_{j=1,\ldots,4} [\frac{det|\hat{v_j}, \hat{v_{j+1}}, \hat{v_{j+2}}|}{\max(||\hat{v_j} - \hat{v_{j+1}}||, ||\hat{v_{j+1}} - \hat{v_{j+2}}||, ||\hat{v_{j+2}} - \hat{v_j}||)}]^2 \\ w_i = \mu_{0^\circ,7^\circ}(\arccos(\hat{l'_1} \cdot \hat{l'_2}))\end{gathered} \tag{2.15}$$

where n is the number of such collinear pairs and $v_j = 1...4$ are the four 3D end vertices of the two lines.

2.1.12 Whole Symmetry

Based on the spirit of the law of symmetry from Gestalt psychology, the whole symmetry term was firstly proposed by Cao et al. [?]. It considers a symmetry measure S for a closed planar shape. It is defined as

$$S = \frac{A}{P^2}, \tag{2.16}$$

where A and P are the area and perimeter of the figure, respectively.

It holds that $S \leq \frac{1}{4\pi}$ for any closed planar figure. A circle is the most symmetrical planar figure with $S = \frac{1}{4\pi}$. For a polygon with m vertices, its symmetry measure $S \leq 4m\tan(\frac{\pi}{m})$. The maximum is achieved if and only if the polygon is the most symmetrical with m equal-length sides. These facts indicate that (2.16) is a rather reasonable measure of symmetry.

A polyhedron consists of more than three faces, each being a polygon.Weconsider the recovered object as the integration of all its planar faces in 3D space. Thus, the whole symmetry measure of a polyhedron with n faces is defined as

$$WS = \sum_{i=1}^{n} \frac{A_i}{P_i^2}, \tag{2.17}$$

where A_i and P_i, $1 \leq i \leq n$, are the area and perimeter of face i, respectively. We expect that given a line drawing, maximizing WS combined with other two criteria would provide us with the most plausible recovered 3D object. The intuition behind it is that if we force the faces of the reconstructed object to be as symmetrical as possible, then the flat 2D line drawing will be inflated into a 3D object.

It should be mentioned that the faces of the 3D object may not be strictly planar. In this case, the area of a face is denoted by the sum of the areas of the triangles obtained by the triangulation of the face.

2.2 Adaptive Parameter Setting in the Objective Function

The twelve image regularities introduced in previous section can be categorized into two groups. The first group's regularity values are determined by a given 3D configuration only. This group includes Face Planarity, Minimum Standard Deviation of Angles (MSDA), Line Orthogonality, Face Perpendicularity, and Whole Symmetry. The other seven regularities belong to the second group, whose regularity values are not only determined by a given 3D configuration, but also by the 2D line drawing's appearance.

No matter for which group of regularities, after initially analyzing the 2D line drawing at the beginning of the 3D reconstruction process, all the regularities' values can be determined when a certain 3D configuration is given in the optimization procedure. The goal of the optimization-based algorithms is to find the optimal 3D configuration which minimizes the value of the weighted sum of the image regularity terms. Because the 2D line drawing is a parallel projection of a 3D wireframe object, the x and y coordinates of the object's vertexes in 3D must be the same as the x and y coordinates of the vertexes in

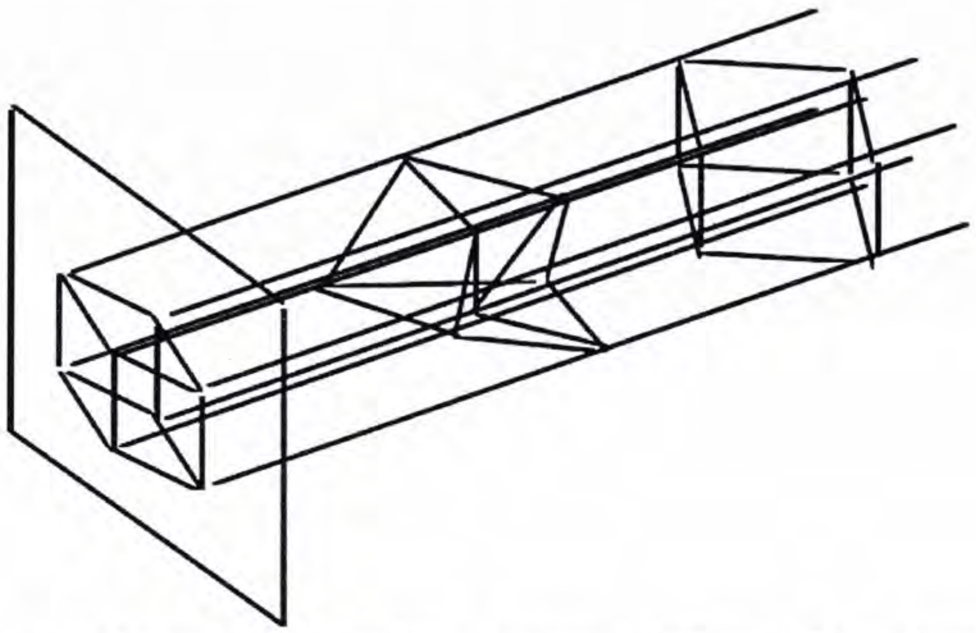

Figure 2.4: Different z coordinates represent different 3D configurations

the 2D line drawing. Thus, only the different assignments of z coordinates for the object's vertexes can represent different 3D configurations. In other words, for each vector $Z = [z_1, z_2, ..., z_n]$, where n is the number of vertexes in the 2D line drawing, there is a corresponding 3D configuration which can be evaluated by the image regularities. Fig. 2.4 illustrates this property. Based on this formulation, we can write the objective function of the optimization-based 3D reconstruction algorithms as

$$F(Z) = W^T \cdot \alpha(Z) = \sum_{i=1}^{12}[w_i \cdot \alpha_i(Z)], \tag{2.18}$$

where $i = 1, ..., 12$ represents one of the twelve image regularities, w_i is the weight of the ith regularity and α_i is the overall value of the ith regularity for the whole 3D configuration. Note that w_i's here are the regularities' overall weights in the final objective function, but not the weights of certain associated entities inside a regularity as in Section 2.1 any more. α_i's value is computed by firstly calculating the ith regularity value for each associated entity (or entities) in the 3D configuration and then summing the individual values up. Take the Line Parallelism regularity as an example. For each possible pair of lines (v_j, v_k) in the 3D configuration, their Line Parallelism regularity value

$\alpha_{parallel,j,k}$ can be calculated as in Eq. (2.5). Then $\alpha_{parallel}$ is calculated as $\alpha_{parallel} = \sum_{j,k} \alpha_{parallel,j,k}$.

2.2.1 Hill-Climbing Optimization Technique

The optimization-based algorithms use the Hill-climbing search technique to look for the optimal 3D configuration (Z^*) which minimizes the objective function $F(Z)$. Although the Hill-climbing search can not guarantee to find the globally optimal point(s), it is effective in most cases of the 3D reconstruction from 2D line drawings because its procedure can be well explained by a heuristic process of inflating the 2D line drawing into the 3D object.

In this inflation process, the input line drawing is thought of as a flat object with all zero z coordinates lying in the xy plane at first. The hill-climbing search starts with this initial condition, which is $Z_0^* = [z_1, z_2, ..., z_n] = [0, 0, ..., 0]$, where z_i is the ith vertex's z coordinate in the 3D space. At the tth step of the hill-climbing process, the objective function's value $F(Z_t^*)$ at the current Z vector Z_t^* is firstly calculated. Then, all the children of Z_t^* are evaluated by the objective function $F(Z)$. A child of Z_t^* is a vector which is one step-size away from Z_t^*. For example, let s be the step-size and the current vector is $Z_t^* = [z_{1,t}, z_{2,t}, ..., z_{n,t}]$, then all the children of Z_t^* are the following $2n$ Z vectors:

$$\begin{array}{c} [z_{1,t}+s, z_{2,t}, ..., z_{n,t}] \\ [z_{1,t}-s, z_{2,t}, ..., z_{n,t}] \\ [z_{1,t}, z_{2,t}+s, ..., z_{n,t}] \\ [z_{1,t}, z_{2,t}-s, ..., z_{n,t}] \\ \cdot \\ \cdot \\ \cdot \\ [z_{1,t}, z_{2,t}, ..., z_{n,t}+s] \\ [z_{1,t}, z_{2,t}, ..., z_{n,t}-s] \end{array} \tag{2.19}$$

For each of these $2n$ children, $F(Z)$ is calculated and the child with the minimum objective function value is selected as Z^*_{t+1}, which is the next step's new current Z vector. If $F(Z^*_{t+1}) < F(Z^*_t)$, then the search process goes on; otherwise, the hill-climbing search process terminates and outputs the current Z vector Z^*_t.

In the real implementation, in order to coarsely inflate the line drawing at first and then refine the inflation result so that the algorithm is faster, there are three rounds of hill-climbing searches with the different step-sizes s_1, s_2, s_3, where $s_1 > s_2 > s_3$. Each round starts with the resulted 3D configuration of its preceding round.

2.2.2 Adaptive Weight Setting and its Explanations

From the definitions of the different image regularities, we can see that the α_i terms in the objective function (2.18) may have very different value ranges. For example, at an intermediate step of a simple 3D cube's reconstruction the Face Planarity term's value $\alpha_{faceplanarity}$ can be more than 27000 while the Line Parallelism term's value $\alpha_{lineparallelism}$ is less than 4. And at the final step of the hill-climbing reconstruction, both $\alpha_{faceplanarity}$ and $\alpha_{lineparallelism}$ are less than one. To reconstruct the object shown in Fig. 1.2, the ranges of the twelve regularity terms are as follows: $\alpha_{face\ planarity} \in [0, 142929.2]$, $\alpha_{MSDA} \in [3.2, 20.9]$, $\alpha_{line\ parallelism} \in [0, 34.4]$, $\alpha_{isometry} \in [0, 10.2]$, $\alpha_{corner\ orthogonality} \in [2.1, 12.1]$, $\alpha_{skewed\ facial\ orthogonality} \in [1.4, 17.5]$, $\alpha_{skewed\ facial\ symmetry} \in [3.1, 24.0]$, $\alpha_{line\ orthogonality} \in [3.4, 20.9]$, $\alpha_{face\ perpendicularity} \in [1.8, 91.3]$, $\alpha_{whole\ symmetry} \in [119.4, 301.1]$, $\alpha_{line\ collinearity} \in [0.4, 115.6]$, $\alpha_{line\ verticality} \in [0, 2.8]$. The varying patterns of this two regularity terms' values during the optimization procedure are also not well correlated. However, in the traditional definition of the objective function (2.18), the regularity weights w_i are all fixed real values. In this way, it is quite possible that during the hill-climbing process only the regularities with large magnitudes dominate the optimization direction. This is a serious problem with the state of the art optimization-based algorithms. In the following we propose a new adaptive weight-setting strategy to solve this regularity range problem.

To solve the regularity range problem, a straightforward idea is to normalize the regularity terms so that their values become being in a same range. For example, dividing the Face Planarity term by 100000 and the Line Parallelism term by 10 can normalize their values into the same range $[0, 1]$ approximately. However, this simple normalization strategy will not work. Its main problem is that the image regularity values can not linearly reflect the 3D configuration's compliance with the regularities' high-level semantics. And the varying patterns of the regularity values during the optimization procedure are also not well correlated. Moreover, at the final steps of the hill-climbing procedure all the regularity terms would be nearly zero and with similar magnitudes. At this time the above simple normalization would make the large range regularities' values too small.

Instead of using the simple constant normalization, we propose to update the value of the regularity weights w_i after each step of the hill-climbing procedure, so that the weights can appropriately adapt to the current situations of the corresponding image regularities. The updating rule of the weights after the tth step is:

$$\begin{aligned} w_{i,t+1} &= w_i/\alpha_{i,t}, i = 1, 2, ..., 12 \\ W_{t+1} &= [w_{1,t+1}, w_{2,t+1}, ..., w_{12,t+1}], \end{aligned} \tag{2.20}$$

where $i = 1, 2, ..., 12$ represents one of the twelve image regularities, w_i's are fixed overall weights of the image regularities which reflect the relative importance of the regularities, $\alpha_{i,t}$ is the ith regularity's overall value at the new current Z vector Z^*_{t+1}, that is $\alpha_{i,t} = \alpha_i(Z^*_{t+1})$, W_{t+1} is the new weight vector and will be used in the $t+1$th step to calculate the objective function's value $F(Z)$.

Substitute the updating rule (2.20) into the objective function (2.18), we can get

$$F_{t+1}(Z) = W_{t+1}^T \cdot \alpha(Z) = \sum_{i=1}^{12}[\frac{w_i}{\alpha_i(Z_{t+1}^*)} \cdot \alpha_i(Z)], \tag{2.21}$$

where $F_{t+1}(Z)$ is the objective function used at the $t+1$th step of the hill-climbing procedure to evaluate all the children of the current Z vector Z_{t+1}^*.

In the practical implementation of the updating rule, when some of the regularity terms $\alpha_i(Z_{t+1}^*)$ become very small, their corresponding weights are not updated any more because at this time these terms are nearly optimal and dividing them by a very small value becomes meaningless.

The proposed adaptive weight-setting strategy can be understood in two ways. In the first way, it can be similarly explained with the idea of the straightforward constant normalization. The new step-wise objective function (2.21) can be interpreted as that all the $\alpha_i(Z)$ in the $t+1$th step are normalized by $\alpha_i(Z_{t+1}^*)$. As all the children vector Z's are just one step-size away from Z_{t+1}^*, $\alpha_i(Z)$'s magnitude would not change very much from $\alpha_i(Z_{t+1}^*)$'s. Thus, in this interpretation the weights' updating rule is just normalizing the regularity values into similar magnitudes, which are around unit 1, with their previous values.

Another way of understanding the adaptive weight-setting strategy is to see the ratio term $\frac{\alpha_i(Z)}{\alpha_i(Z_{t+1}^*)}$ as a whole which reflects how much improvement on the ith regularity can be made by using the child vector Z instead of using the current vector Z_{t+1}^*. In this way, the whole objective function can be seen as a weighted sum of the improvement ratios on the individual regularities using Z instead of Z_{t+1}^*. This explanation tells us that using the adaptive strategy not the absolute regularity values determines the optimization direction any more, but the improvement ratios of the regularity values determines. This characteristic of the adaptive weight-setting strategy solves the regularity value range problem very well. It is intuitively reasonable and also suitable for the hill-climbing 3D inflation process. The experimental results in Chapter 4 demonstrate that the proposed adaptive weight-setting strategy can significantly improve the performance of the optimization-based 3D reconstruction algorithms. Comparing with the traditional fixed-weight strategy, the proposed adaptive strategy makes the quality of the reconstructed 3D objects much better

In this chapter the adaptive parameter-setting strategy is proposed to solve the regularity value range problem. However, how to decide the values of w_i in (2.21), which reflect the relative importance of the image regularities, is still an unsolved problem. In the next chapter, we will present a parameter-learning framework to solve this problem.

Chapter 3

Parameter Learning

The weights w_i of the image regularities in the objective function (2.21) are free parameters which traditionally are set with heuristics or trials. When the number of regularities becomes large, it is almost impossible to manually set the parameters appropriately. Assume ten image regularities are used to do the reconstruction and the weight of each regularity is an integer within $[1, 100]$, then the total number of feasible parameter assignments would be $100^{10} = 1 \times 10^{20}$, which is an untacklable huge number. Because of this, most of the practical 3D reconstruction systems can only use a small number (three to four) of the image regularities. Thus the large number of parameters problem in the optimization-based 3D reconstruction algorithms has hindered their way to fully utilize all the image regularities' powers. In this chapter, we propose a parameter-learning framework to solve this problem. The proposed framework will be able to learn the appropriate weights for a large number of image regularities.

3.1 Construction of A Large 3D Object Database

In order to learn the appropriate image regularity weights, a sufficiently large database of 3D objects need be built firstly. These 3D objects will be used as the ground-truth objects in the parameter-learning procedure.

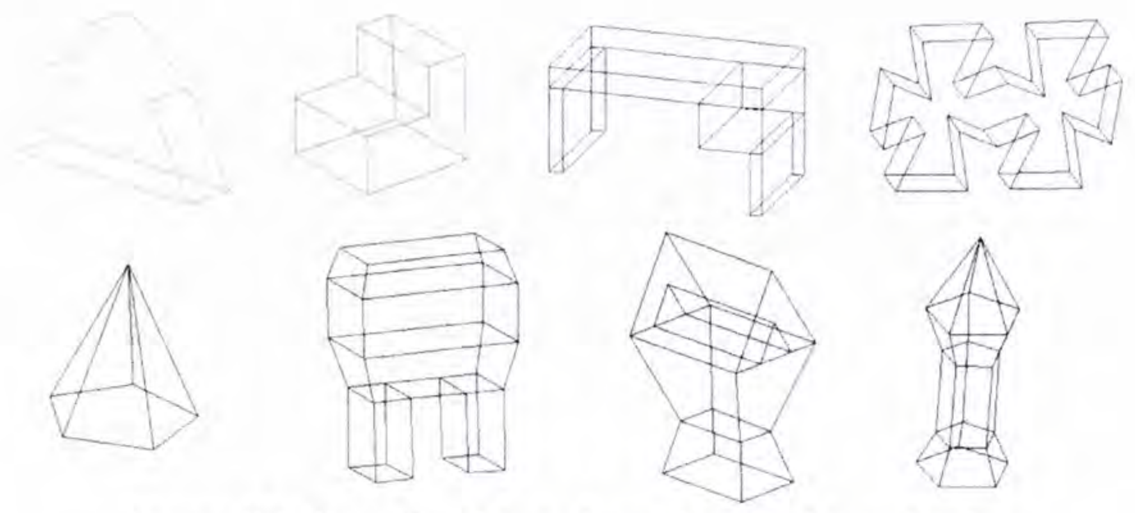

Figure 3.1: Samples objects from the 3D object database

To build the database, we designed more than 70 different 3D objects and created them with a CAD software. All the 3D objects are only with planar faces. This is sufficient because current curved object reconstruction algorithms usually first convert the curved objects into planar objects and then do the 3D reconstruction. These objects hold a large variety of 3D shapes which cover the common 3D structures used in practical applications. Simple cubes, pyramids and complex objects like lamp, desk and house are all included in this database. Fig. 3.1 shows some sample objects from the database.

All the 3D objects are represented by a wireframe model, in which all of the object's vertexes are indexed and their x, y, z coordinates are known, and all the edges are indexed by their two ending vertexes. This 3D wireframe model is consistent with the 2D line drawing's graph representation, which makes it convenient to project the 3D objects into 2D line drawings. We build a tool with OpenGL to view the 3D wireframe models and project them in to 2D line drawings.

3.2 Training Dataset Generation

To learn the regularity weights, we not only need the ground-truth 3D objects, but we also need their corresponding 2D line drawings from which the 3D objects are expected to be reconstructed. For each 3D object, it has a large number (infinite number theoretically) of possible 2D parallel projections with different projection angles. Which 2D line drawing of a 3D object should be used in the training dataset? This question can be answered only after we investigate the criterion of a good 3D reconstruction result from a 2D line drawing.

The 3D reconstruction from a single 2D line drawing is an ill-posed problem, which means that there are many possible correct 3D configurations for a single 2D line drawing. Fig. 3.2 demonstrates this problem. Give a 2D line drawing, the aim of the optimization-based 3D reconstruction algorithms is to reconstruct the 3D configuration which is in accordance with human being's interpretation of the line drawing. Take the line drawing shown in Fig. 3.2 as an example. When a person sees the line drawing, he would almost definitely interpret this object as a 3D isometric cube with its eight edges having the same lengths. However, this line drawing can actually represent a very long cube with a special projection angle. Based on the reconstruction criteria that the reconstructed 3D object should be the same as what the human beings perceive, the correct reconstruction of the 2D line drawing shown in 3.2 should be the isometric cube. This reconstruction criteria is appropriate not only because it is consistent with human beings' perception, but also because it is using the maximum likelihood idea in its underlying principle. This understanding can be seen from the cube example too. If the line drawing in Fig. 3.2 is the projection of a long cube, then only a very limited scope of projection angles can produce similar line drawings. However, if it is an isometric cube, there are quite a lot of projection angles which can produce similar line drawings as the one in Fig. 3.2. Thus, we can see that the expected best reconstruction result of the 3D reconstruction algorithms is actually the 3D configuration which has the highest probability to produce the given 2D line drawing after a casual parallel projection. In another way of interpretation,

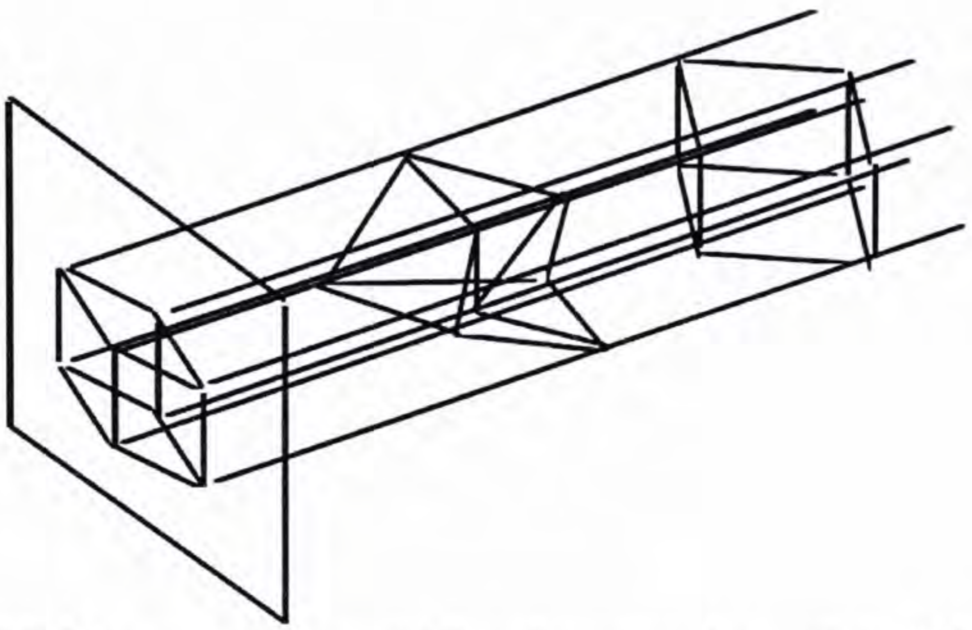
Figure 3.2: Many possible correct 3D configurations for a cube's 2D line drawing

we can also see the reconstruction result as a stable 3D configuration which after a small rotational perturbance the appearance of the 2D projection does not change very much. There is a reference paper [59] which provides theoretical discussions of this problem. This understanding of the reconstruction criterion can help us decide which 2D projections of a 3D object to be used in the training dataset.

In order to select the appropriate 2D projections of the 3D objects for the training dataset, we firstly project each 3D object with a large number of different projections angles. As a rotation in 3D can be represented by a combination of separate rotations which are along the x-axis, y-axis and the z-axis respectively, we enumerate all the possible projection angles of a 3D object by setting the x, y, z rotations from $0°$ to $180°$ respectively, with a step-size $10°$. Thus there are totally $18^3 = 5832$ candidate line drawings for each 3D object. The next step is to select the best line drawing from the candidates. To make sure that we can include the most appropriate 2D projection in the training dataset, we select 10 out of the 5832 line drawings into the training dataset for each 3D object. The method of selecting these line drawings is as follows:

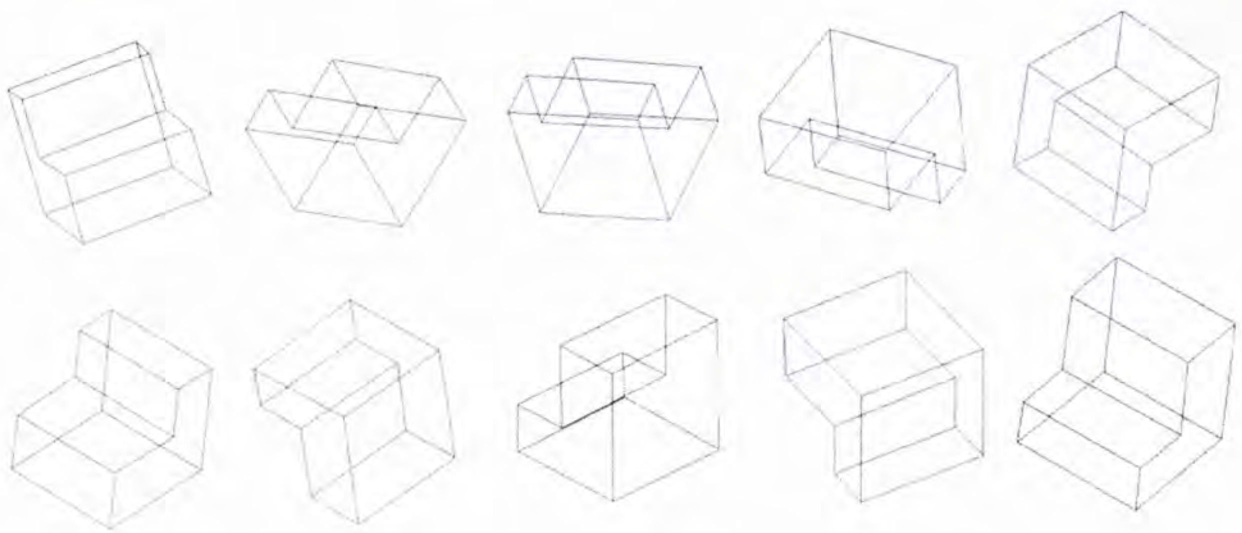

Figure 3.3: Training samples of a bed-shaped object

We firstly use the state of the art 3D reconstruction algorithm to reconstruct all the 5832 line drawings and then compare the reconstructed results with the ground-truth object using the error measure defined in Algorithm 2. The best ten line drawings with minimum reconstruction errors for each object are selected into the training dataset. After this automatic selection procedure, we have also manually checked all the selected line drawings to make sure that the most appropriate line drawings are included in the training samples for each 3D object. The ten training samples of a bed-shaped object are shown in Fig. 3.3.

3.3 Parameter Learning Framework

After having prepared all the training data, the whole parameter learning framework can be introduced in this section. The basic idea of learning the parameters is to try all different parameter combinations and use them to reconstruct the line drawings in the training dataset, then compare the reconstructed results with the ground truth objects and choose the best parameters which produce the most similar reconstruction results as the ground truth objects. This idea of parameter learning is straightforward. However, as mentioned before there are a huge number of different parameter combinations which makes it impossible to try all different parameters, thus the exhaustive search method can not be used. We will introduce the Evolutionary Algorithms to solve this explosive number of combinations problem. And we will also introduce a quantitative way of measuring how similar a reconstructed object is with the ground truth object. After these two parts, the complete learning framework will be introduced at the end of this section.

3.3.1 Evolutionary Algorithms

An evolutionary algorithm is a generic population-based optimization algorithm. It creates an environment which mimics the biological evolution process and the natural selection process in nature so that it expects the optimal solution to the original problem can be found just as only the best species of animals survives in nature. It is a probabilistic algorithm which iterates until the result converges or meets a stopping criteria. At iteration t, an evolutionary algorithm maintains a population $P_t = \{x_{1,t}, x_{2,t}, ..., x_{n,t}\}$. Each individual of the population represents a potential solution to the problem at hand and it is implemented by some data structure S which is usually a fixed-length string. Each individual solution $x_{i,t}$ is evaluated by a fitness function which measures how well it solves the original problem. Then, the new population of the $t + 1$th iteration is formed by selecting the more fit individuals. After the selection step, some members of the new population undergo some transformations by means of genetic operators to form new solutions. The transformation can be unary or with higher order. An unary transformation creates new individuals by changing a single individual in some way. Higher order transformations create new individuals based on information of several individuals. These two kinds of transformations correspond to the gene mutation and the chromosome crossover in the biological evolution. After some number of iterations (generations, in genetic programming terminology), the program converges and the best individual in the final population is expected to be the optimal solution. The process of an evolutionary algorithm is illustrated in Algorithm. 1. Although the evolutionary algorithms can not guarantee to find the global optimal solution, they have shown very good performance in many general optimization applications because they are not easily to be trapped into a suboptimal region in the solution space. We use the evolutionary algorithm in our parameter-learning framework to learn the regularity weights. A comprehensive introduction of evolutionary algorithms can be found in [38].

```
Require: $t \leftarrow 0$, initialize $P_t$, evaluate $P_t$
  while (not termination-condition) do
    $t \leftarrow t + 1$
    select $P_t$ from $P_{t-1}$
    alter $P_t$
    evaluate $P_t$
  end while
```

Algorithm 1: Evolutionary algorithm's process

3.3.2 Reconstruction Error Calculation

In order to measure the similarity between the reconstruction result and the corresponding ground truth object, we need design an error function which calculates the difference between the reconstruction result and the ground truth object. As each line drawing is a parallel projection of a ground truth object from a certain projection angle, the difference between a reconstructed object and the ground truth object is only in the z coordinates of all the vertexes. Their x, y-coordinates are the same. Therefore, to appropriately measure the difference we can firstly align the two objects along the z direction and then calculate a distance measure among the corresponding vertexes.

It is not easy to define a good alignment of two different 3D objects, even though their x, y coordinates are the same. Here, we adopt a strategy which unifies the alignment step and the distance calculation step. Our method is to exhaustively search the best alignment, which produces the smallest distance measure, in a promising z-alignment search range. We use the L1-norm to measure the distances between corresponding vertexes. In this way, both the best alignment and the difference measure between the two objects are derived simultaneously. There is one more thing to take care about which is the Necker cube illusion. Because of the Necker cube illusion, there are two possible 3D interpretations for each 2D line drawing. Both of the two interpretations are correct and actually they correspond to the same 3D topology. The only difference between these two interpretations is that their corresponding z-coordinates are with opposite signs. Thus, when we try to calculate the difference between a reconstruction result and its ground truth object, we would test both the reconstruction result and its Necker cube illusion's counterpart which has all the z-coordinates' signs reversed and pick the smaller difference of the two as the correct measure of the reconstruction error.

Let n be the number of vertexes in the line drawing. Then we use the two Z vectors $Z_{recontructed}[i], i = 1, .., n$ and $Z_{ground\ truth}[i], i = 1, .., n$ to represent the reconstruction result and the ground truth 3D object respectively. It is obvious that if we add/subtract a constant value to/from all the dimensional elements in a Z vector, the corresponding 3D object's appearance is not changed but the object is just translated with a certain distance along the z direction. Thus, before searching the best alignment and determining the reconstruction error, we firstly roughly align the two objects by repositioning them so that $Z_{recontructed}[1] == Z_{ground\ truth}[1] == 0$. The whole procedure of calculating the reconstruction error for one line drawing is summarized in Algorithm. 2

```
Require: roughly align the two objects so that
  $Z_{reconstruted}[1] == Z_{groundtruth}[1] == 0$, $MinError = MaxValue$
  for $displacement = -range$ to $range$ do
    $Error_1 \leftarrow 0$
    $Error_2 \leftarrow 0$
    for $i = 1$ to $n$ do
      $Error_1 + = |Z_{groundtruth}[i] - (Z_{reconstruted}[i] + displacement)|$
      $Error_2 + = |Z_{groundtruth}[i] - (-Z_{reconstruted}[i] + displacement)|$
      {$Error_2$ is handling the Necker cube illusion counterpart}
    end for
    if $Error_1 < MinError$ then
      $MinError \leftarrow Error_1$
    end if
    if $Error_2 < MinError$ then
      $MinError \leftarrow Error_2$
    end if
  end for
  return $MinError$
```

Algorithm 2: Reconstruction error calculation for one line drawing

3.3.3 Parameter Learning Algorithm

With the training dataset, the Evolutionary Algorithm and the reconstruction error calculation method, we can present the whole parameter-learning framework now. Our aim is to find the best parameter assignment with which the total reconstruction error on the whole training dataset is minimized. After adopting the adaptive weights-setting strategy introduced in Chapter 2, the free parameters of the optimization-based 3D reconstruction algorithms are just the fixed weights w_i of the image regularities in the step-wise objective function Eq. (2.21). In our implementations we totally use twelve image regularities as introduced in Chapter 2. However, in the following descriptions we would generally use r to represent the number of image regularities.

The optimization-based 3D reconstruction algorithms search for the best Z vector which produces minimum objective function values, thus for the regularity weights in the objective function, it is not important what their actual values are but only their relative ratios are important. Therefore, we can arbitrarily fix one regularity's weight and just tune the others'. The parameter-learning problem's degree of freedom is actually $r - 1$. So in general we can think of the parameter-learning problem as looking for a best value for a $r-1$-dimensional vector O. As the search space is extremely large, we can not exhaustively test each possible value of O. Thus we use an Evolutionary Algorithm to search for the best value.

The fitness function of the Evolutionary Algorithm in our parameter-learning framework utilizes the reconstruction error calculation method introduced in Algorithm 2. It gives higher fitness scores to the vector values which produce less reconstruction errors for the whole training dataset. At each iteration of the evolutionary algorithm, the evolving population is just a set of potential vector values of O. The fitness function thus can give higher chance of survival to the individuals with better values. As introduced in Section 3.2, in the training dataset there are 10 line drawings for each 3D object. To calculate the overall reconstruction error on the whole dataset, the fitness function only picks the best reconstruction result among the 10 line drawings' results for each object. The pseudo code of the fitness function is shown in Algorithm. 3

The population's reproduction step in our proposed parameter-learning framework has two types of variation operators. The first operator is crossover, which interchange the values at some dimensions of two input individual vectors. The crossover operator is a binary operator which takes two input vectors and outputs two new vectors. There is a parameter $pCross$ in the Evolutionary Algorithm which controls the probability that a give couple of individual vectors is applied with the crossover operator. The second variation operator is mutation, which changes the numerical values at some dimensions of an input vector. The mutation operator is a unary operator. There is also a parameter $pMut$ which controls the probability that a given individual vector is applied with the mutation operator. And there is another parameter $pMutPerBit$

```
Require: Input weights vector $W$ (includes $O$ and the fixed weight), $n$ is the number of 3D objects in the database, $TotalError = 0$
  for $i = 1$ to $n$ do
    $GroupMinError = MaxValue$
    {there are 10 line drawings for each 3D object}
    for $j = 1$ to 10 do
      Reconstruction from $LineDrawing[i][j]$ with $W$
      Compute the $TmpReconError$ with Algorithm. 2
      if $TmpReconError < GroupMinError$ then
        $GroupMinError \leftarrow TmpReconError$
      end if
    end for
    $TotalError+ = GroupMinError$
  end for
  return $-TotalError$
  {Return $Minus\ TotalError$ because we need fitness measure, not error measure}
```

Algorithm 3: Fitness Function

in the mutation operator which represents the probability that the value at a given dimension of an individual vector is changed. In the learning framework we set a value range for each dimension of the objective vector O. Thus when a certain dimension is undergoing a mutation, its new value is randomly chosen from this range.

After the fitness evaluation of all the new individuals produced by the reproduction step, the selection step of the algorithm just selects the best individual vectors with highest fitness scores to form the new generation of the population and keeps the population size unchanged. The algorithm will continue the same process of reproduction and selection and produce new generations until no improvement can be made on the population or a certain number of iterations has been done. The final result of the Evolutionary Algorithm in the parameter-learning framework is the best individual vector in the last generation. This vector, together with the fixed weight of an arbitrarily selected regularity, is the final learning result of the parameter-learning framework. We will show in the experiments in Chapter 4 that using the learned weights can produce much better 3D reconstruction results than using the old manually-set weights. The whole procedure of the parameter-learning framework is show in Algorithm.4

```
Require: $t \leftarrow 0$, randomly initialize $P_t$ with $pop\_size$ feasible $O$ vectors
  while (not termination-condition) do
    do crossover and mutation on $P_t$, produce an intermediate population $P'_t$
    Evaluate $P'_t$ with Algorithm.3
    select the best $pop\_size$ vectors from $P'_t$ to form $P_{t+1}$
    $t \leftarrow t + 1$
  end while
  return the best $O$ vector from $P_t$ and the fixed weight
  {the best vector in the final population together with the fixed weight is
  the whole set of learned weights}
```

Algorithm 4: Parameter-learning framework

Chapter 4

Experimental Results

In this chapter we show the experimental results of our proposed algorithms. There are two sections which show the performance of the adaptive parameter-setting strategy and the parameter-learning framework respectively.

4.1 Adaptive Parameter Setting

In this section, we compare the reconstruction performance between using the traditional fixed regularity weights and using the proposed adaptive parameter-setting strategy. There are two parts of the experiments in this section.

4.1.1 Use Manually-Set Weights

In the first part, we use some manually-set regularity weights which had been showing reasonably good performance in practice. The manually set weights' values are shown in Table 4.2. We firstly fix the weights as old approaches and reconstruct some objects from the database. Then, we use the adaptive weights-setting strategy to reconstruct the objects again. The graphical results are shown in Fig.4.1

Fig.4.1(a) column shows the three input 2D line drawings. Fig.4.1(b) are the reconstruction results with the proposed adaptive weights-setting strategy. Fig.4.1(c) are the reconstruction results with the traditional fixed regularity

Image Regularities	MSDA	Face Planarity	Line Parallelism
Manually-Set Weights w_i	80	100	100
Image Regularities	Isometry	Corner Orthogonality	Skewed Facial Orthogonality
Manually-Set Weights w_i	60	70	70
Image Regularities	Skewed Facial Symmetry	Line Orthogonality	Face Perpendicularity
Manually-Set Weights w_i	70	70	50
Image Regularities	Whole Symmetry	Line Collinearity	Line Verticality
Manually-Set Weights w_i	40	50	50

Table 4.1: Manually-set image regularity weights

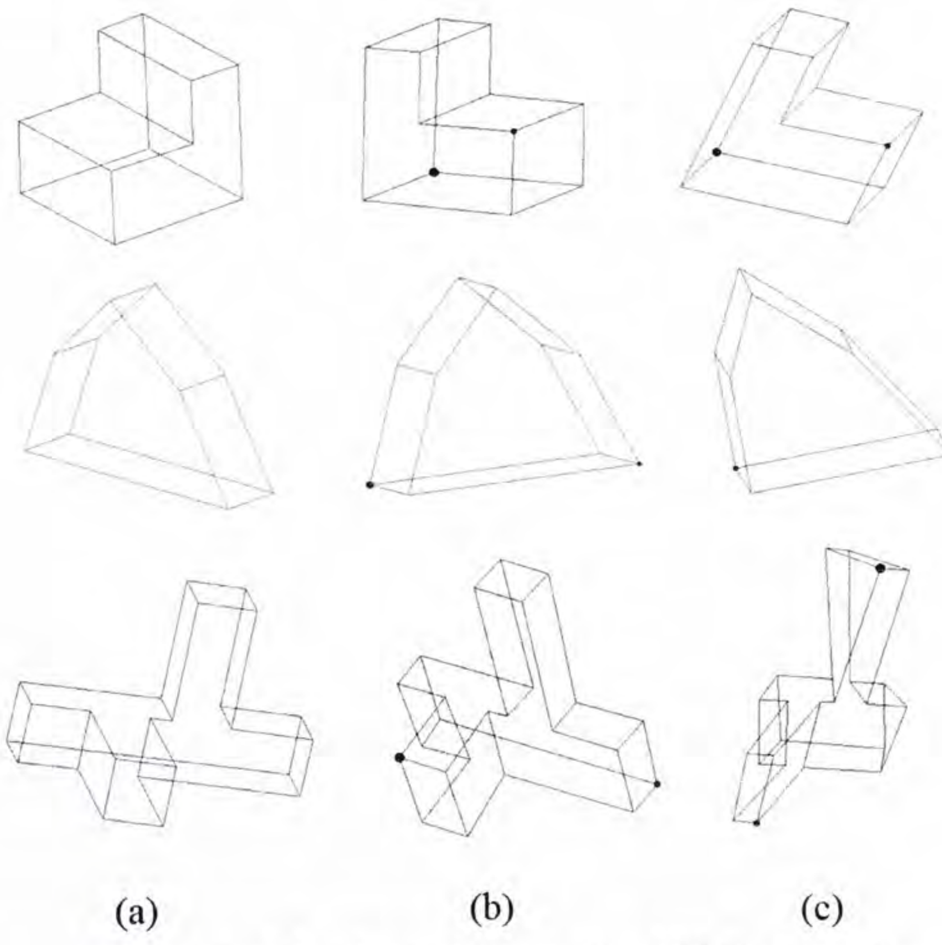

Figure 4.1: Adaptive weights-setting VS. fixed weights

weights. All the reconstruction results are shown in a different view angle from the input 2D line drawing so that we can better see the reconstructed 3D configuration. Comparing Fig.4.1(b) and Fig.4.1(c) we can see that using the adaptive parameter-setting strategy can make the reconstructed objects much more congruent with human being's interpretation. The reconstructed objects with the adaptive parameter-setting strategy are well rectified and have regular and tidy appearance. In the third row of Fig. 4.1, with the fixed weights the reconstruction result is even totally wrong while the result with the adaptive weight-setting is very good. Thus it is obvious that the adaptive parameter setting strategy can better utilize all the image regularities so that the reconstructed results are more congruent with human being's perception.

We also apply the fixed weights method and the adaptive weights-setting method on the whole training dataset with the manually-set weights. Table 4.2 shows the performance evaluation results. The total error on the dataset is calculated in the way as the fitness function (Algorithm.3) except that the return value is now the TotalError itself. From the results, we can see that using the adaptive weights-setting strategy has significantly reduced the overall construction error on the database. The error using the adaptive weights-setting strategy is only $4432/15662 \simeq 28.3\%$ of the error using the fixed weights. We also compared the reconstruction errors on each object with the two different strategies and tried to find some failure cases. However, we found that the adaptive weights-setting strategy consistently outperforms the fixed weights strategy on all the objects in the database. Therefore, we believe that the proposed adaptive weights-setting strategy is better than the old fixed weights strategy.

	On the Whole Training Dataset
Total Error of Adaptive Weights-Setting	4432
Total Error of Fixed Weights	15662

Table 4.2: Compare adaptive weights-setting strategy with fixed weights on the whole training dataset using manually-set weights

	Learned Best Weights	Training Error
Adaptive Weights	80 81 37 54 36 69 95 68 81 72 27 88	3669
Fixed Weights	80 22 61 70 36 67 84 90 46 5 27 55	7506

Table 4.3: Learning weights with two different weights-setting strategies

4.1.2 Learn the Best Weights with Different Strategies

In the second part of the experiments, in order to fairly compare the two weights-setting strategies, we try to learn the best weights with the two different weights-setting strategies using our proposed weights-learning framework, and then compare the learned weights' performance on the whole training dataset. The parameter values of the learning algorithm is as follows: MSDA's weight is fixed at 80, the population size is 20, the value range of each weight is $[0, 100]$, $pCross = 0.6$, $pMut = 0.8$ and $pMutPerBit = 0.01$.

The learning results are shown in Table 4.3, where all the numbers have been rounded to integers. The learned weights in the second column are arranged in the following order: w_{MSDA}, $w_{FacePlanarity}$, $w_{LineParallelism}$, $w_{Isometry}$, $w_{CornerOrthogonality}$, $w_{SkewedFacialOrthogonality}$, $w_{SkewedFacialSymmetry}$, $w_{LineOrthogonality}$, $w_{FacePerpendicularity}$, $w_{WholeSymmetry}$, $w_{LineCollinearity}$, $w_{LineVerticality}$. From the Training Error column, we see that learning with the adaptive weights-setting strategy produces less than half of the reconstruction errors on the whole training dataset than learning with the fixed weights strategy does.

Based on the above results shown in subsection 4.1.1 and subsection 4.1.2, we can draw the conclusion that using the proposed adaptive weights-setting strategy can significantly improve the quality of the 3D reconstruction results from 2D line drawings.

4.2 Evolutionary-Algorithm-Based Parameter Learning

This section shows the experimental results of the parameter learning framework. In the experiments, we firstly randomly split the whole dataset into two groups with equal sizes. One group is used for training, and the other is used for testing. We then compare the reconstruction performances on the testing dataset with the learned weights and with the old manually-set weights. As the training set and the testing set are disjoint, this comparison can fairly demonstrate whether using our proposed parameter learning framework can effectively find better regularity weights.

We did the experiments several times with different training and testing sets and different learning parameters. The results are summarized in Table 4.4. In this Table, Learning 1's learning parameters are: MSDA's weight is fixed at 80, the population size is 20, the value range of each weight is $[0, 100]$, $pCross = 0.6$, $pMut = 0.8$ and $pMutPerBit = 0.01$. Learning 2's learning parameters are: Face Planarity's weight is fixed at 80, the population size is 20, the value range of each weight is $[0, 100]$, $pCross = 0.6$, $pMut = 0.9$ and $pMutPerBit = 0.01$. In Learning 3, we do not fix any regularity weight but let the evolutionary algorithm to learn all the twelve weights. The other parameters are: the population size is 20, the value range of each weight is $[0, 100]$, $pCross = 0.6$, $pMut = 0.9$ and $pMutPerBit = 0.01$. We find that in Learning 3 the converging speed becomes much lower but the learned results do not show very much degradation. The manually-set weights in Table 4.4 are the same as used in Section 4.1.1. And the learned best weights are also arranged in the order as in Section 4.1.1. All the learning algorithms in these

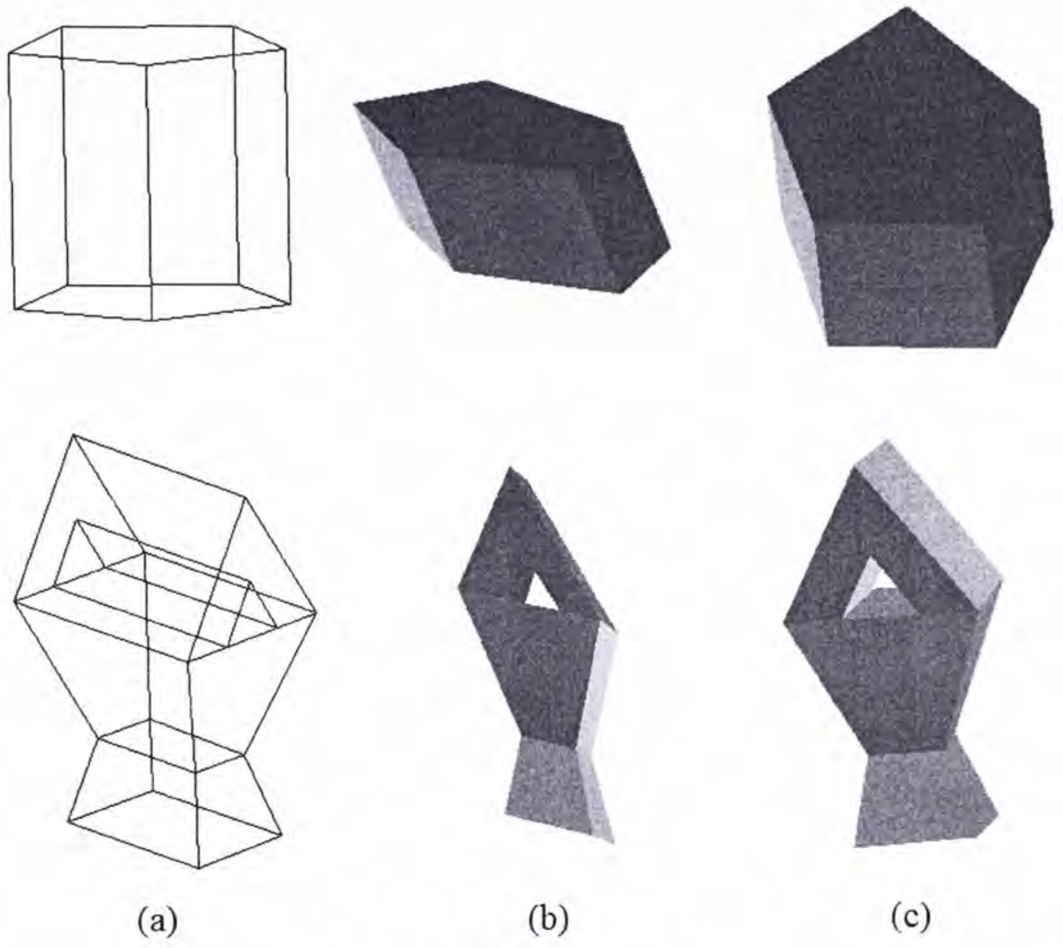

Figure 4.2: (a) Input line drawing. (b) Reconstruction result with manually-set weights. (c) Reconstruction result with learned weights.

experiments use the adaptive parameter-setting strategy introduced in Chapter 2.

In Table 4.4, we see that the learned weights can significantly reduce the reconstruction errors on the three testing datasets comparing with the manually-set ones. Learning 3 does not fix any regularity weight, so its average reconstruction precision improvement is the smallest. For Learning 1 and Learning 2, the average reconstruction precision improvement is 25.2%. Some graphical comparisons are shown in Fig. 4.2. These results demonstrate that the proposed parameter learning framework can effectively learn better regularity weights for the 3D reconstruction from 2D line drawings.

Fig. 4.3 shows two failure cases in which the reconstruction results with the learned weights are worse than the reconstruction results with the manually-set

	Learned Best Weights	Training Error	Testing Error	Precision Improvement
Manually-Set Weights	-	-	1901	-
Learning 1	80 83 99 58 47 69 92 68 65 22 27 93	2036	1621	17.3%
Learning 2	83 80 54 70 38 58 84 74 86 57 16 19	2136	1471	29.2%
Learning 3	7 90 10 38 37 99 36 79 41 36 47 55	1869	1531	24.2%

(a) Training&Testing Dataset 1

	Learned Best Weights	Training Error	Testing Error	Precision Improvement
Manually-Set Weights	-	-	2185	-
Learning 1	80 81 96 56 46 70 89 69 66 72 30 42	2063	1658	31.8%
Learning 2	83 80 54 70 38 58 84 74 86 57 16 19	2004	1571	39.1%
Learning 3	100 80 82 37 61 98 70 79 98 35 24 53	1966	1909	14.5%

(b) Training&Testing Dataset 2

	Learned Best Weights	Training Error	Testing Error	Precision Improvement
Manually-Set Weights	-	-	2263	-
Learning 1	80 81 37 54 36 69 95 68 81 72 27 88	1774	1882	20.2%
Learning 2	83 80 54 70 38 58 84 74 86 57 16 19	1665	1993	13.5%
Learning 3	7 79 9 38 37 70 36 81 41 45 46 56	1784	2120	6.7%

(c) Training&Testing Dataset 3

Table 4.4: Weights-learning results and comparisons

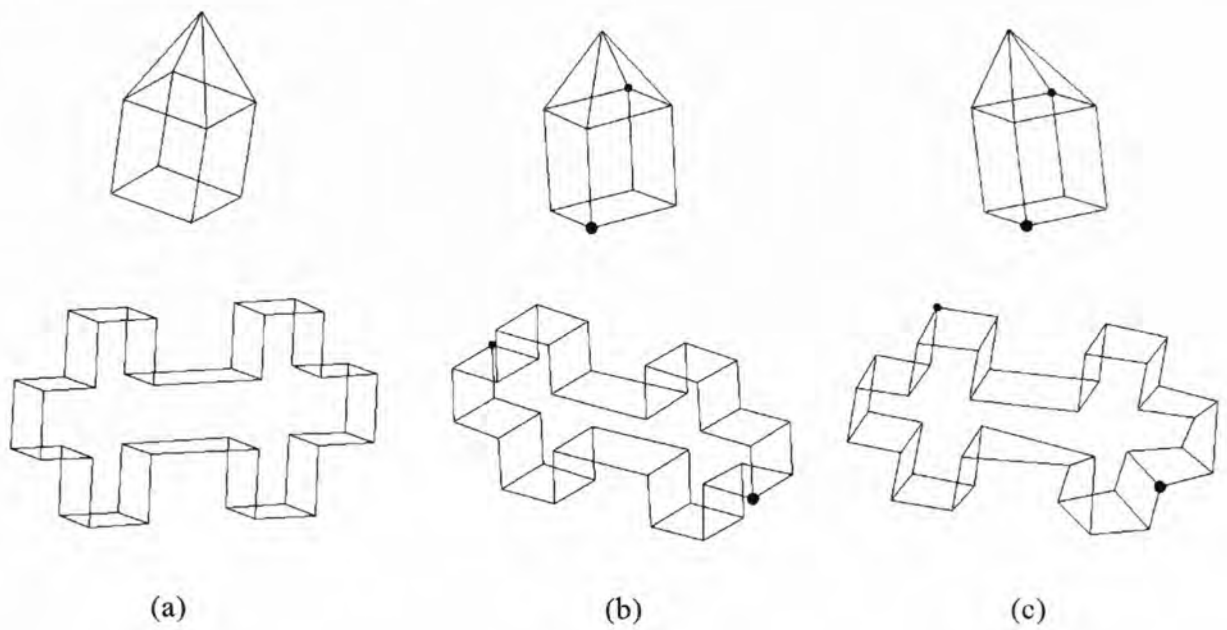

Figure 4.3: Failure cases with the learned weights. (a) Input line drawing. (b) Reconstruction result with manually-set weights. (c) Reconstruction result with learned weights.

weights. We use the learned weights of Learning 2 on dataset 2. The first failure case is due to the low weight of Isometry regularity in the learned weights. And in the second case, because the input line drawing has a large number of pairs of parallel lines, the Line Parallelism regularity is very important for this object. But the weight of the line parallelism regularity is relative low in the learned weights, thus its result is failed. Although there could be failure cases for the learned weights, its overall performance on the whole dataset is still much better than the manually-set weights' as shown before.

Chapter 5

Conclusions and Future Work

In this thesis we propose an adaptive parameter-setting strategy and an evolutionary-algorithm-based parameter-learning framework to improve the performance of the optimization-based 3D object reconstruction from 2D line drawings. The adaptive parameter-setting strategy solves the problem that the image regularities' values vary largely during the hill-climbing optimization procedure. It can be understood as a proper normalization method for the image regularities at each search step. And it can also be understood as making the final objective function in the optimization-based algorithms as a weighted sum of improvement ratios on the image regularities but not a weighted sum of absolute regularity values anymore. The experimental results show that the adaptive parameter-setting strategy brings dramatic improvement on the 3D reconstruction results from 2D line drawings.

The evolutionary-algorithm-based parameter-learning framework searches for the best weights assignment for the image regularities in the final objective function. We build a large 3D object database to provide the ground truth objects in the training and testing datasets. The evolutionary algorithm is used to search for the best solution in a very large search space. The experimental results show that the proposed parameter-learning framework can effectively find better weights assignments which produce significantly better reconstruction results than the old manually-set weights do.

The proposed approaches in this thesis have effectively improved the performance of the optimization-based 3D reconstruction from 2D line drawings algorithms from the perspective of parameter setting and tuning. In the future work, other aspects of the optimization-based 3D reconstruction algorithm can be exploited to improve its performance, such as the optimization strategy, result refinement methods and etc.

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] 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.

[5] L. Cao, J. Liu, and X. Tang. 3D object retrieval using 2D line drawing and graph based relevance feedback. *Proc. ACM Int'l Conf. Multimedia*, pages 105–108, 2006.

[6] L. Cao, J. Liu, and X. Tang. What the back of the object looks like: 3D reconstruction from line drawings without hidden lines. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(3):507–517, 2008.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

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

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

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

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

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

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

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

[33] 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.

[34] 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.

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

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

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

[38] Z. Michalewicz. *Genetic algorithms+ data structures*. Springer, 1996.

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

[40] 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.

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

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

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

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

[49] 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.

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

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

[52] 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.

[53] 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.

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

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

[56] 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.

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

[58] Y. Wang, Y. Chen, J. Liu, and X. Tang. 3D reconstruction of curved objects from single 2D line drawings. 2009.

[59] D. Weinshall and M. Werman. On view likelihood and stability. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):97–108, 1997.

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

[61] 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.