Founded in 1905

ADAPTIVE SLICING OF CLOUD DATA FOR REVERSE ENGINEERING AND DIRECT RAPID PROTOTYPING MODEL CONSTRUCTION

BY

WU YIFENG

(B.Eng., M.Eng.)

DEPARTMENT OF MECHANICAL ENGINEERING

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

NATIONAL UNIVERSITY OF SINGPAORE

2003

ACKNOWLEDGEMENT

I would like to express my sincere respect and gratitude to my research supervisors, A/Prof. Loh Han Tong, A/Prof. Wong Yoke San and A/Prof. Zhang Yunfeng for their invaluable guidance, advice and discussion in the entire duration of the project. They shared their knowledge, provided inspiration and were ready to help whenever I needed their advice. I am very fortunate to have such kind, knowledgeable and passionate supervisors in my academic life.

I would also like to give my sincere appreciation to A/Prof. Fuh Ying Hsi, for his kind assistance in my project. Special thanks are given to Dr. Tang Yaxin and Mr. Ning Yu, for their kind helps to complete the laser fabrication of the case study in this thesis.

My thanks also go to Ms. Tan Hwee Lynn Cyrene and professional officer Mr. Neo Ken Soon for their help on laser scanning.

I would also like to thank my colleagues and friends Ms. Zhang Wei, Dr. Liu Kui, Mr. Fan Liqing, Mr. Wang Zhigang, Ms Li Lingling, Ms. Wang Binfang and Mr. Rishi Jian for their encouragement and friendship.

Finally, I would express my sincere appreciation to my family for their constant support and deep love.

SUMMARY

Reverse engineering (RE) is the process of creating a CAD model and manufacturing a part using an existing part or a prototype, which can be utilized to produce a copy of an object, extract the design concept of an existing model, or re-engineer an existing part. In RE process, the shape of the part can be rapidly captured by utilizing the optical non-contact measuring techniques, e.g., laser scanner. This normally produces a large cloud data set that is usually arbitrarily scattered. Rapid prototyping (RP) is a material-addition fabrication method in which the physical part is generated layer-by-layer. In order to produce a physical part model of complex geometric shape rapidly, RP has been widely used. Therefore, modelling point cloud for RP fabrication is an essential step to integrate RE and RP so that reconstruction of a part can occur rapidly.

In general, modelling point cloud for RP relies mostly on surface construction from cloud data and CAD model slicing using a commercial software. However, this process may inherently lead to three progressive shape errors among cloud data, CAD model, STL model and final RP model, which are difficult for the user to control. Moreover, surface construction is time-consuming and needs expert experience.

In this thesis, an intuitive method of point cloud segmentation by using the shape-error to control the layer thickness so that the built part will be within a specified tolerance is presented. The thickness of each layer in the generated model will

therefore be different. In this respect, we assume that the RP machines used for fabrication accept arbitrary thickness.

Two methods for adaptive slicing have been developed. One uses a correlation coefficient to determine the neighbourhood size of projected data points, so that a polygon can be constructed to approximate the profile of projected data points. It basically consists of the following steps:(1) the cloud data points are segmented into several layers along the RP building direction; (2) points within each layer are treated as co-planar and a polygon is constructed to best-fit the points; (3) the thickness of each layer is determined adaptively such that the surface error is kept to just within a given error bound.

The other method uses wavelets to construct a polygon, and the general steps are similar to the first one. However, the most important step, which is the polygon construction, is different. This method has two main steps: Firstly, the nearly maximum allowable thickness for each layer is determined with the control of the band-width of projected points. Secondly, for each layer, the profile curve is generated with a wavelets method. In detail, the boundary points between two regions in one layer are extracted and sorted by a tangent-vector based method, which uses a fixed neighbourhood size to quicken the sorting process. Wavelets are then applied to the curve construction from the sorted data points from coarser to finer level under the control of the shape tolerance, such that the constructed curve has nearly minimal number of points while the shape error is within specified tolerance.

Algorithms for the developed methods have been implemented using C/C++ on the OpenGL platform. Both methods can deal with complex surfaces with multiple loops. Simulation results and actual case studies demonstrate the efficacy of the algorithms.

TABLE OF CONTENTS

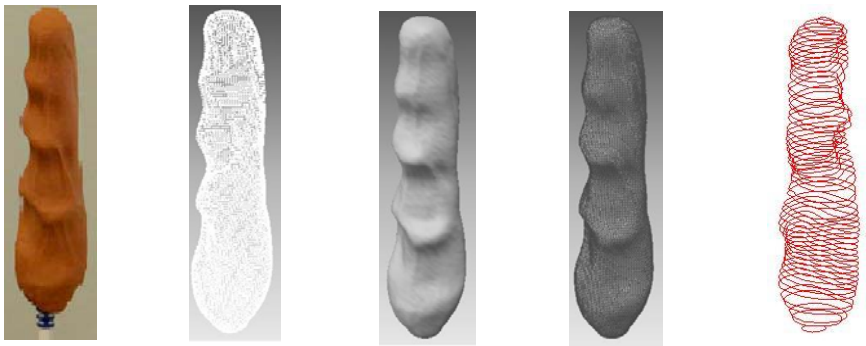
LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

1.1 Problem Statement

Reverse engineering (RE) refers to creating a CAD or digital model from an existing physical object, which can be utilized to produce a copy of an object, extract the design concept of an existing model, or re-engineer an existing parts (Varady et at. 1997). There are various properties of a 3D physical object that one may be interested in recovering by reverse engineering, such as its shape, colour and material properties. This thesis addresses the problem of recovering 3D shape, for computer-aided 3D modelling.



a b c d e

a. Unknown digital model of 3D object, **U**
b. Scanned data points, **X**
c. Constructed surface model, **S**
d. Constructed STL model, **T**
e. Layer-based 3D RP model, **R** (wire form)

Fig. 1.1: Example of direct RP model construction from cloud data

Rapid prototyping (RP) is an emerging technology to fabricate physical parts quickly by building them layer-by-layer. This thesis mainly addresses an RE problem of layered-based RP model construction directly from cloud data captured from a physical object.

As shown in Fig 1.1, the goal of direct RP model construction can be stated as follows: Given a set of sample points **X** assumed to lie on or near an unknown 3D object **U**, create a layer-based 3D model **R** approximating **U**. The constructed model **R** should have the same topology as **U**, and can be every where close to **U**, i.e. the shape error which is estimated by the largest distance between **X** and **R**, meets the requirement of shape tolerance ε**.**

In our application, the cloud data **X** is obtained by registering the scanned data from different view angles. We assume that **X** has no holes (or have been filled), and **X** lies on or near the unknown object, but **X** can be noisy and unstructured (not ordered).

Reconstruction methods typically first reconstruct a 3D model and then slice this 3D model to achieve a layer-based 3D model. In this thesis, an approach is presented that directly slices the sample points **X**, and constructs a layer-based 3D model.

Moreover, the layer-based 3D model construction problem is first examined in its general form that makes few assumptions about the sample **X** and the unknown 3D object **U**. The cloud points **X** may be noisy, and no structure or other information is assumed within them. The 3D object may have arbitrary topology, including sharp features such as the creases and corners. The constructed model **S** should have the same topology as **U**, and is close to **U** within a specified tolerance.

1.2 Reverse Engineering and Rapid Prototyping

We were led to consider the general layer-based 3D model construction problem stated above by the demands of reducing product development time, that is to integrate RE and RP technology for rapid product development.

1.2.1 Reverse engineering

In RE, a part model designed by the stylist, usually in the form of wood or clay mock-up, is firstly sampled and then the sampled data are transformed to a CAD representation for further fabrication. The shape of the stylist's model can be rapidly captured by utilizing optical non-contact measuring techniques, e.g., laser scanner.

There are several application areas of reverse engineering such as to produce a copy of a part when no original drawings or documentation are available, to re-engineer an existing part when analysis and modifications are required to construct a new improved product, and to generate a custom fit to human surfaces, for mating parts such as helmets, space suits or prostheses.

The process of reverse engineering can usually be subdivided into three stages (Varady et al., 1997, Li et al., 2002), i.e., data capturing (aimed at acquiring sample point coordinates), data segmentation (aimed at clustering points into groups, representing curves or surfaces of the same type) and CAD modelling and/or updating (aimed at constructing bounded surface regions from segmented data groups and combining the surface regions into complete geometric models). We will introduce these three steps of reverse engineering in the following sections.

1.2.1.1. Data capturing

Data capturing is a crucial step in RE, and there are many different methods for acquiring shape data. Table 1.1 shows the most commonly used methods in data acquisition and their main advantages and disadvantages. Essentially, each method uses some mechanism or phenomenon for interacting with the surface or volume of the object of interest. There are non-contact methods, where light, sound or magnetic fields are used, while in others the surface is touched by using mechanical probes at

the end of an arm, such as a CMM, which can usually produce accurate results up to 10um or better. Non-contact methods usually give a fast and high resolution result but with noise and dense data points, while contact methods give a more precise result but with a slow speed, especially for objects of complex shape.

Table 1.1 Data acquisition methods

Methods	Principle	Usages	Advantages	Disadvantages
Optical	Light triangulation	CAD	Fast ; High resolution	Affected by the surface quality of reflecting light; Dense
Acoustic	Speed of sound	Medical; Sea depth detection	Special cases	noisy
Magnetic	Hall effect	Medical Oil tube detection	Special cases	noisy
CMM	Contact	Car design CAD	Fast to regular shapes such as circle, cone, sphere etc. ; High precision; Easier for curve or surface reconstruction	Slow to complex shapes

The contacted measurement with a CMM is fast and highly repeatable when measured geometric elements are line, plane, cylinder, sphere and cone etc., because the process requires only a limited number of probing points and the probe radius compensation is quite straightforward. The machine can be programmed to follow paths along a surface and collect very accurate, nearly noise-free data. However, in the case of complex surfaces involving numerous measurement points, the measurement motion becomes difficult (Yan and Gu 1996). Also, contact methods are not good for soft materials.

Compared to the contact measurement techniques, non-contact means have a higher resolution of surface digitization and more rapid measurement speed, e.g., the VIVID 900 Laser scanner can capture over 300,000 points in 2.5 seconds or 75,000 points in 0.5 seconds with a fast mode scan. In addition, the corresponding dimensional accuracy is usually in the range from several hundredths to several tenths of a millimeter for RE applications. However, there are also some problems for the non-contact measurement methods (Varady et al., 1997, Lee and Woo 2000): The methods tend to pick up redundant points and the distributed density of points measured in the digitization steps often do not meet the surface geometric trend. Moreover, the bright and shiny objects are difficult to be measured by optical methods.

1.2.1.2. Data segmentation

Data segmentation is a process to divide the original sampling data point set into subsets, one for each natural surface, so that each subset contains just those points sampled from particular natural surfaces. There are generally three different methods for data segmentation: edge-based, face-based and feature based (Milroy et al., 1997, Yang and Lee 1999, Jun et al., 2001).

The edge-based method (Milroy et al., 1997) is one popular approach of a two-stage process, edge detection and linking. This works by trying to find boundaries in the point data representing edges between surfaces. In the edge detection process, the local surface curvature properties are used to identify boundary present in the measured data. Curvature is selected as the mathematical basis for edge detection. Hamann (1993) presented a method for curvature estimation from 3D meshes, and Kobbelt (2000) extracted curvature from a locally fitted quadratic polynomial approximant. If edges are being sought, an edge-linking process follows, in which

disjoint edge points are connected to form continuous edges. This technique thus infers the surface from the implicit segmentation provided by the edge curves. Yang and Lee (1999) extended the edge-based method by using parametric quadric surface approximation to identify the edge points.

In the face-based segmentation (Chen and Schmitt 1994, Peng and Loftus 1998, Jun et al., 2001), a group of points is connected into a distinctive region with similar geometrical properties, such as normal vectors or curvatures, and each region is then transformed into appropriate surfaces using a region-growing algorithm. The exact boundary edges can be derived by intersection or other computations from surfaces. Triangles are firstly generated from input scanned points and the cost values for each edge from these triangles are computed as reference values. The cost value of each polygonal mesh is defined by the area and the normal. The basic concept of detecting boundary meshes is to select all the meshes whose cost value is higher than a certain level. Then, a region-growing process was imposed to aggregate a polygonal mesh into subregion until the area of the subregion reaches the user-defined area criterion.

Feature-based segmentation method (Jun et al., 2001) extracts or reconstructs geometric features directly from scanned point set using intelligent algorithms such as an artificial neural network or genetic algorithm.

1.2.1.3. Surface modelling

Segmentation process outputs labeled points belonging to particular regions. For these regions, techniques are given for surface modelling by many researchers. Surface fitting is a technique of representing large amount of data into a concise form which is useful for later manipulations. A surface fitting problem can be posed as follows: let D be a domain in the (x, y) plane, and suppose F is a real valued function defined on D.

Suppose we know the values F(x_i, y_i) (i=1, 2,…,N) located in D. Find a function f defined on D which reasonably approximates F. In geometric modeling, surfaces are presented by either polyhedral or curved surface approximation. Polyhedral approximation is described in (Reqicha 1990, Eck and Hoppe 1996). Curve surface approximation methods may be divided into three types: algebraic, parametric, and dual. Algebraic surfaces are the ones where the surfaces are approximated using polynomial equation, and there are two approaches for algebraic surface fitting (Menq and Chen 1996). In general the algebraic surfaces have infinite domain while parametric surfaces are bounded. Dual representation of surfaces included both algebraic and parametric surfaces. Many surface fitting algorithms, such as quadratic surface fitting, B-spline surface fitting, rotational surface fitting and lofted surface fitting etc., have been reported (Ueng et al.,1998).

1.2.2 Rapid prototyping

To generate physical objects from CAD models directly, Rapid prototyping can produce the physical part by adding layer by layer. Rapid Prototyping (RP) is an emerging, non-traditional fabrication method and has been recognized as a valid tool to shorten the lead-time from design to manufacture effectively. A variety of RP technologies have emerged (Yan and Gu 1996). They include stereolithography (SLA), selective laser sintering (SLS), fused deposition manufacturing (FEM), laminated object manufacturing (LOM), and three-dimensional printing (3D printing). Among these technologies, the advantages and disadvantages were discussed by Chua and Leong (1996).

In RP, the STL file format (Jacobs 1992) has become the de facto standard. An STL file consists of a list of triangular facet data. Each facet is uniquely identified by a unit normal and three vertices.

1.3 Previous Work

In general, modelling point cloud for RP can be realised in three different approaches (Lee and Woo 2000). As shown in Fig. 1.2, in the first approach, a surface model is reconstructed from the point cloud and is closed up as a solid. Then, this solid can be sliced based on its geometry information or can be converted to a RP file format, such as STL, which will be sliced by commercial software. The second approach creates an STL-format file of a model directly from the point cloud (e.g., triangulation) (Chen et al., 1999, Lee et al., 2000, Sun et al., 2001), and this STL file can be fed into RP machine directly. RP machine can slice STL model. The third approach goes directly from point cloud to an RP slice file (layer-based model) (Liu 2001). This slice file need not be further sliced by RP machine.

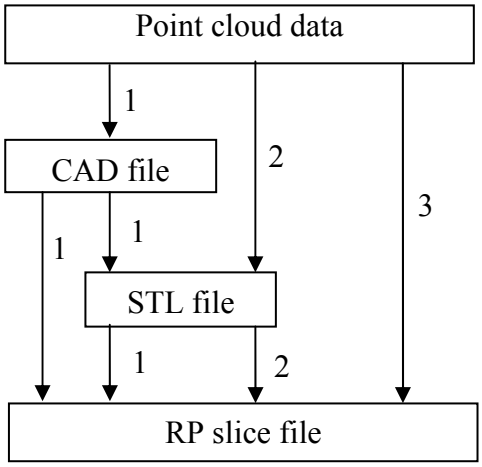


Fig. 1.2: Point cloud data modelling for RP fabrication

1.3.1 Surface model based slicing

Most of researchers focus on surface-model based slicing for RP manufacture (Lee and Woo 2000). The CAD model is firstly constructed with surface modelling method, and the surface model is closed up to form a solid model. Then this model is sliced to generate a RP model. Most commercial CAD systems have the function to generate the STL file, from CAD model directly, and this STL file is further sliced and transferred into an RP manufacturing file format suitable for SLA, LOM, FDM, SLS and so on.

1.3.1.1. Surface reconstruction

3D modelling is a process of segmentation and surface fitting. The surface reconstruction has received considerable attention in the past. The main issues are to deal with surfaces of arbitrary topology, to allow non-uniform sampling, and to produce models with provable guarantees, e.g., smooth manifolds that accurately approximate the actual surface (Boissonnat and Cazals 2002).

Early attempts by Boissonnat (1984) and Edelsbrunner (1994) were approaches aiming at constructing a geometric data structure such as Delaunay triangulations of the data points and extracting from this structure a set of facets that approximate the surface. Along this direction, Boissonnat and Cazals (2002) further devised the method of natural neighbour interpolation, which allows for dealing with non-uniform samples. The natural neighbour interpolation was used, which is computed from Voronoi diagram of the sample points. Further, it directly produces a surface without computing an intermediate polyhedral approximation, and the reconstructed surface is implicitly represented as the zero-set of a signed pseudo-distance function which interpolates the data points and their normals.

A different approach consists in using the input points to define a signed distance function and to compute its zero-set. The surface is therefore regarded as a level surface of an implicit function defined over the entire embedding space. Hoppe et al. (1992,1993) calculated a tangent plane at a sample point using nearby neighbour points and assigned a distance to the plane as the signed distance function. Polygonal vertices were then obtained by finding the sample points with the zero set of this function using the marching cube algorithm. The advantages of Hoppe's algorithm are as follows: (1) the algorithm is suitable for handling scattered points because no assumptions of inherent structures of the sampled data set are made; and (2) the algorithm is capable of automatically inferring the topological type of the surface, including the presence of the boundary curves. The major complaint against marching-cubes-based algorithm is that it is slow for dealing with cloud data.

Another kind of surface reconstruction method is based on segmentation and fitting (Hoffman and Jain, 1987).The cloud data is divided into a suitable patchwork of surface regions to which an appropriate single surface is fitted. Data segmentation, accomplished either manually or through software, defines the patch boundary curves and produces a patchwork of surface regions (Weir et al., 1996). Data modelling methods, such as those employing parametric (Varady et al., 1997) or quadric (Weir et al., 1996, Chivate and Jablokow 1993) functions are applied to fit appropriate surfaces to the data patches. Non-uniform rational B-spline (NURB) curves and surfaces are a current research topic due to their ability to accurately approximate most types of surface entity encountered in design and manufacturing applications (Piegl and Tiller 1997).

1.3.1.2. Slicing CAD model

Most commercial CAD systems have the function to generate RP-format files, such as the STL file, from CAD-format files directly. At present, the interface between CAD systems and RP machines is realized by the direct transferring of an STL file (Xu 1999), i.e, STL file slicing occurs in the RP system. Here, we review some methods for slicing CAD models or STL models. In general, there are two slicing approaches for determining the layer thickness, i.e., uniform slicing and adaptive slicing. Uniform slicing is the simplest approach in which a CAD model is sliced at equal intervals. If the layer thickness is sufficiently small, a smooth part model can be obtained. This may, however, result in many redundant layers and a long build time on the RP machine. On the other hand, if the layer thickness is too large, the build time is short, but one may end up with a part having a large shape error.

Kulkarni and Dutta (1996), and Xu (1999) presented an adaptive slicing algorithm to slice CAD models, namely it determines a variable layer thickness for an object represented in parametric form. This algorithm uses the normal curvature in the vertical direction to determine the maximum allowable layer thickness for the surface at the reference level with a pre-specified cusp height. The sliced data are fed into Stratasys 3D system for RP fabrication. Mani et al. (1999) gave a method for region-based adaptive slicing of CAD models. Whereas in traditional adaptive slicing the user can impose a single surface finish (cusp height) requirement for the whole object, in region-based adaptive slicing, user has the flexibility to impose different surface finish requirements on different surfaces of the model. The sliced data are fed into Stratasys 3D system with a FDM file format.

Tata et al. (1998) provided an efficient method for layer manufacturing from an STL model. This algorithm is based on three fundamental concepts: choice of criterion for accommodating complexities of surfaces, recognition of key characteristics and

features of the object, and development of a grouping methodology for facets used to represent the object. The output is 2D data slice data that can be machined by CNC machines, or by SLA RP machines.

Sabourin et al. (1997) presented a method to slice the STL model. It builds exterior regions of a part within regular thin layers, using adaptive layer thickness, so as to produce the required precise part surface. At the same time it also builds the interior regions of the part with thick and wide material application. The sliced data are fed into Stratasys 3D system with a FDM file format.

1.3.2 Direct STL-file generation from cloud data

This approach directly generates the STL file from cloud data, without model construction. Direct generation of STL file from the scanned data is favourable in that it can reduce the time and error in the modelling process.

Chen et al. (1999) presented a data reduction method for automatic STL file generation directly from CMM data points. First, all the measured points are jointed and triangulated based on the vertex-to-vertex rule of STL file format, and the normal for each triangle is calculated. Second, for each point, select the neighbouring triangles, which share the concerned point, and use their normals to determine whether the point can be removed or not. Thus the data points in a flat surface are reduced. Third, to further reduction, an error bound is specified, and then regions with similar normals are formed. After reduction of point set, an algorithm for re-triangulation is implemented, to cover the blank region. This algorithm can generate a STL file from CMM data directly, without a surface model construction. However, it is only applied to CMM data, with the structure information. It does not work for laser scanned data points, which are dense and have no structure information, such as sequences of the points.

Lee et al. (2002) presented a method for STL file generation from scanned data points based on segmentation and Delaunay triangulation. A triangular net is generated to maximize the smallest angle over all triangulations by adopting a bounding box and max-min angle criterion with the consideration of topological relationships among 3D points. After that, segmentation is performed based on local and global curvature between triangles and the segments are classified as plane, smooth and rough regions. Then, Delaunay triangulation is performed maintaining the segment boundary so that geometric characteristics still exist. However, this method can only deal with structured data points, i.e. whose sequences are known.

Sun et al. (2001) presented a unified, non-redundant triangular mesh method to model the cloud data. This algorithm consists of two steps. First, an initial data thinning is performed to reduce the copious data set size, employing 3D spatial filtering. Second, the triangulation commences using a set of heuristic rules, from a user defined seed point. Thus, a geometric model can be constructed from 3D digitized data. In fact, a STL file can be generated with this method.

1.3.3 Direct layer-based model construction

To construct a layer-based model from cloud data is still a new issue. Liu (2001) developed an automated segmentation approach for generating a layer-based model from point cloud. This is accomplished via three steps. Firstly, the cloud data is adaptively subdivided into a set of regions according to a given subdivision tolerance (the maximum distance between cloud data points and their respective projected plane), and the data in each region is compressed by keeping the feature points (**FP**s) within the user-defined shape tolerance using a digital image processing based reduction method. Secondly, based on the **FP**s of each region, an intermediate point-based curve

model is constructed, and RP layer contours are then directly extracted from the models. Finally, the RP layer contours are smoothed and subsequently closed to generate the final layer-based RP model. He has demonstrated that the developed system is able to generate a layer-based model from point cloud. However, the subdivision tolerance, which is used to control the layer thickness, does not have an explicit relationship with the shape error, thus making the actual shape-error difficult to control.

1.4 Research Objectives and Organization of the Thesis

As seen in the previous section, the surface model generated from the first approach has the advantage that it can be edited. However, the shape error of the final RP model (between the RP model and the cloud data) comes from three sources: (1) shape error between the cloud data and the surface model, (2) shape error between the surface model and the STL model, and (3) shape error between the STL model and the layer-based RP model. This will make the shape error of the RP model very difficult to control. The model generated from the second approach is effectively a STL model. The shape error of the final RP model comes from two sources: (1) shape error between the cloud data and the STL model, and (2) shape error between the STL model and the layer-based RP model. Still, the control of the final shape error is not straightforward. In the third approach, a layer-based model is directly generated from the cloud data, which is very close to the final RP model. Therefore, there is only one source of shape error. If this error can be controlled effectively, this approach will have a clear advantage over the other two modelling approaches in terms of shape error control on the RP model.

1.4.1 Overview of algorithm

In this thesis, we will present an intuitive approach of point cloud segmentation by using the shape-error to control the layer thickness so that each layer will yield the same shape error. The thickness of each layer in the generated model will therefore be different. In this respect, we assume that the RP machines used for fabrication are able to handle arbitrary thickness.

We develop two methods for adaptive slicing. One is adaptive neighbourhood search (ANS) based adaptive slicing, and it uses correlation coefficient to determine the neighbourhood size of projected data points, so that we can construct a polygon to approximate the profile of projected data points. It consists of the following steps:

(1) The cloud data are segmented into several layers along the RP building direction;

(2) Points within each layer are treated as planar data and a polygon is constructed to best-fit the points;

(3) The thickness of each layer is determined adaptively such that the surface error is kept within a given error bound.

The other one is wavelets-based adaptive slicing, and it uses wavelets to construct a polygon, and the general steps are similar to the first one. However, the most important step, polygon construction, is different. This method has two main steps: First, near maximum allowable thickness for each layer is determined with the control of band-width of projected points. This estimated band-width is controlled by user specified shape tolerance. Second, for each layer, the profile curve is generated with wavelets method. The boundary points between two regions in one layer are extracted and sorted by a tangent-vector based method, which uses a fixed neighbourhood size to quicken the sorting process. Wavelets are then applied to the

curve construction from sorted data points from coarser to finer level under the control of the shape error.

The wavelet-based method has better error control and is more robust than the first one, because fewer parameters are used. Moreover, the approach is fast, since fixed neighbourhood size is applied in the sorting process and fast wavelets decomposition and reconstruction are used to curve construction, and parallel algorithm can be used for curve construction in different layers.

In our research work, the algorithms can deal with the cloud data and model construction where,

(1) The cloud data is an unorganized, noisy sample of an unknown object

(2) This unknown object (surface) can have arbitrary topological type

(3) No other information, such as structure in the data or orientation information, is provided

(4) Constructed RP model has the same shape errors, as that of the real product, if we ignore the machine error of the RP machine.

1.4.2 Organisation of thesis

The thesis contains five chapters as follows:

Chapter 1 introduces the major processes of reverse engineering and rapid prototyping. The literature review is given. The research objectives are then outlined.

Chapter 2 presents the ANS-based slicing method, i.e., a shape error-controlled direct RP model construction algorithm directly from unorganized cloud data. This algorithm is based on adaptive neighbo urhood size determination based on correlation coefficients of planar points.

Chapter 3 describes the WAS method, i.e., a wavelets based direct RP model construction algorithm from cloud data. The multiresolution techniques are reviewed and then polygon construction from cloud data is presented.

Chapter 4 illustrates the algorithms with simulation results and real case studies. The advantages and disadvantages of these two algorithms are compared.

Chapter 5 summarizes the work and proposes suggestions for future work.

CHAPTER 2 ANS-BASED ADAPTIVE SLICING

In this chapter, an adaptive neighbourhood search based adaptive slicing (ANSAS) method is presented. This algorithm constructs the direct RP model under the control of shape error. It directly slices the point cloud along a direction used to generate a layer-based model, which can be applied directly for fabrication using rapid prototyping (RP) techniques. It employs an iterative approach to find the maximum allowable thickness for each layer. The main challenge is that the thickness of the layer must be carefully controlled so that every layer will yield the same shape error, which is within the given tolerance bound. A correlation coefficient is derived from the given shape error and employed in a neighbourhood search for the construction of the curve in each layer. This method seeks to generate a direct RP model with minimum number of layers based on a given shape error. Issues including multiple loop segmentation in layers, profile curve generation, and data filtering, are discussed.

2.1 The Proposed Adaptive Segmentation Approach

In our approach, the cloud data set is segmented into a number of layers by slicing the point cloud along a user-specified direction. The data points in each layer are projected onto an appropriate plane and then these projected data points will be used to reconstruct a polygon approximating the profile curve. Segmentation of point cloud is an important step in the process of direct RP model construction. In general, there are two slicing approaches for determining the layer thickness, i.e., uniform slicing and

adaptive slicing. Uniform slicing is the simplest approach in which point cloud is sliced at equal intervals. If the layer thickness is sufficiently small, a smooth part model can be obtained. This may, however, result in many redundant layers and a long build time on the RP machine. On the other hand, if the layer thickness is too large, the build time is short, but one may end up with a part having a large shape error. Adaptive slicing is one of the approaches to resolve this problem.

In cloud data modelling, a shape tolerance is usually given to indicate the maximum allowable deviation between the generated model and the cloud data points. Therefore, an intuitive method is to use the shape tolerance to control the layer thickness of point cloud during segmentation. Since the actual shape error can only be calculated after a layer is constructed, the segmentation process should be iterative in nature. This process can be illustrated using the example in Fig. 2.1. Given a slicing direction (also the RP building direction), the initial layer is obtained from one end with a sufficiently small thickness. The mid-plane in the initial layer is used as the projection plane and the points within the layer are projected onto this plane. The 2D points on the plane are then used to construct a closed polygonal curve. The distances between the points and the polygon are then calculated as the actual shape errors (σ_{actual}). If σ_{actual} is smaller than the given shape tolerance (σ_{given}), the layer thickness is adaptively increased until that σ_{actual} is very close to σ_{given}. This final thickness is the maximum allowable thickness for the first layer. The first layer is thus constructed by extruding the polygon along the slicing direction with the determined maximum allowable thickness. The subsequent layers are constructed in the same manner.

This layer-based model can be used directly for RP fabrication. The details of this approach are described in the following sections.

2.2 Planar Polygon Curve Construction within a Layer

Once a layer is obtained, the points within the layer are projected (along the slicing direction) onto the projection plane. The next step is to construct one or several closed polygon curves to accurately represent the shape defined by these points. If the maximum distance between any two neighbouring points in one closed curve is less than the distance between two loops, we can use a threshold to separate these closed curves. Since each polygon curve is closed, these polygon curves are constructed at a time and the different curves are split naturally. Here, we only discuss the single loop curve construction problems.

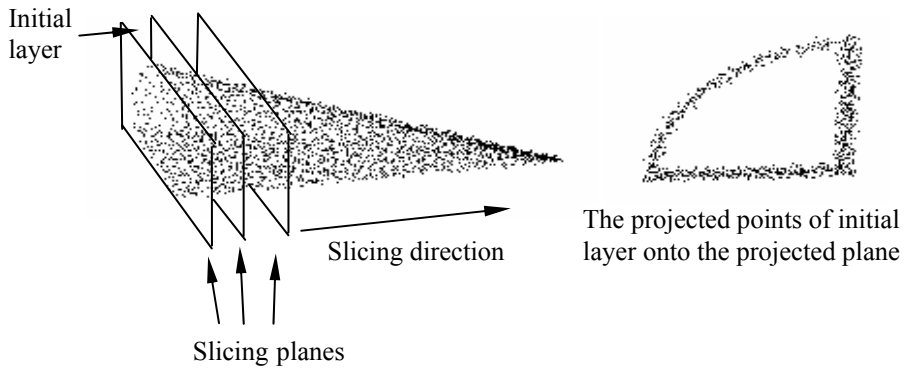


Fig. 2.1: Point cloud slicing and projecting

The curve construction is to approximate the unorganised point set by a curve. In our application, since the projected points have local linearity, we can use line segments to represent the local shape of points and thus form a polygon. To approximate the point set accurately, the polygonal curve must keep the feature points of original shape defined by the point set. Liu (2001) presents an algorithm to construct feature-based planar curve from the unorganised data points. In his algorithm, the data points are firstly sorted based on the estimated oriented vector to generate the initial curve. It begins with a fixed point and then this point and the centre of its

neighbourhood points determine the oriented vector of point, based on which the next point can be retrieved. Repeatedly, the feature points are determined. The data is compressed by removing redundant points other than the feature points. Finally, the curve is obtained by linking all the feature points using straight-line segments. However, in his algorithm, when determining the oriented vector of a point, he used a fixed radius of neighbourhood points. This could result in losing some feature points if the chosen radius is too large; and if the chosen radius is too small, the algorithm lacks of efficiency.

Lee (2000) used the concept of correlation in probability theory (Pitman 1992) to compute a regression curve. In our work, we employ the correlation concept to determine the radius of neighbourhood adaptively in the process of curve construction. Based on this idea, we present an efficient algorithm to reconstruct a polygonal curve from unorganized planar point set.

2.2.1 Correlation coefficient

Correlation refers to the degree of association between two or more quantities. In a two-dimensional plot, the correlation coefficient is used to measure the strength of the linear relationship between two variables on the two axes. Let X and Y be two variables, then the correlation coefficient of X and Y can be defined as:

$$\rho(X,Y)=\frac{Cov(X,Y)}{S(X)S(Y)} \tag{2.1}$$

Where, $Cov(X,Y)=E[(X-E(X))(Y-E(Y)]=E(XY)-E(X)E(Y)$ and $E(\zeta)$ denotes an expectation of a random variable ζ. $S(\zeta)$ represents a standard deviation of a random variable ζ. Let (X, Y) stands for a set of N data points $\{\mathbf{P}_i = (x_i, y_i), |i = 1, \ldots, N\}$, then Eq. (2.1) can be re-written as:

$$\rho(X,Y)=\left|\frac{\sum_{i=1}^{N}\left(x_i-\bar{x}\right)\left(y_i-\bar{y}\right)}{\sqrt{\sum_{i=1}^{N}\left(y_i-\bar{y}\right)^2}\sqrt{\sum_{i=1}^{N}\left(x_i-\bar{x}\right)^2}}\right| \tag{2.2}$$

Where $\bar{x}$ and $\bar{y}$ are the average values of $\{x_i\}$ and $\{y_i\}$, respectively. $\rho(X, Y)$ has a value between 0 and 1 representing the degree of linear dependence between X and Y. In our application, we use this idea to check the linearity of the points with a neighbourhood. Fig. 2.2 shows a point **P** with two neighbourhood radii, R_1 and R_2. The correlation coefficients are 0.921 and 0.632 for R_1 and R_2, respectively. Points within R_1 show a better linearity. This is obvious as the neighbourhood of **P** within R_2 include inflection points.

In the problem of planar curve construction, we need to find the maximal neighbourhood for each segment, in which a line segment can accurately fit the points. Using this idea of correlation coefficient, we can determine the neighbourhood radius adaptively.

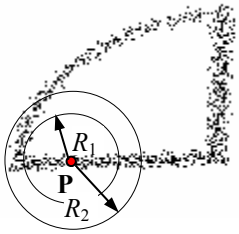


Fig. 2.2: Correlation coefficients of neighborhood points of point **P**

2.2.2 Initial point determination

The initial point (**IP**) is a reference point to start the construction of the first segment of the polygonal curve from the planar data points. As the points are unorganised and

error-filled, the **IP** selection is very important. Liu (2001) proposed a random search method in which a point (start point) is randomly selected from the data points and the points within a certain neighbourhood of this point are identified. The centre of these points is then calculated and the point closest to the centre is selected as the **IP**. The problem with this method is that if the randomly selected point is very close to an inflection point, the **IP**, subsequently identified by using a fixed neighbourhood radius, may be of too much deviation from the original shape. An example is shown in Fig. 2.3. If point **Q** is selected as the start point, the centre point **O** of the neighbourhood will be far away from the original shape and the closest point to **O** will also be the worst point to be the **IP**.

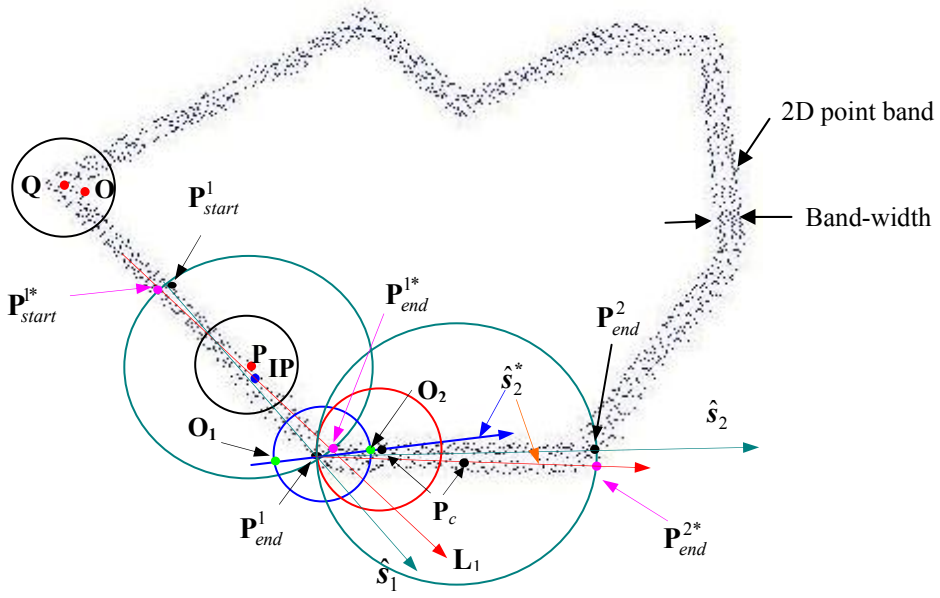


Fig. 2.3: **IP** determination and first, second segment construction

To resolve this problem, it is necessary to make sure that the points within the first neighbourhood have good linearity. In our approach, we first randomly select a start point, and then use a fixed radius to find its neighbourhood points. The correlation coefficient (ρ) of this neighbourhood is then calculated. If ρ is larger than a pre-set

bound, this neighbourhood is used to find the **IP**, i.e., the point that is nearest to centre of this neighbourhood. Otherwise, we will re-select a point and repeat this checking process. For the case in Fig. 2.3, point **O** will be dropped due to its poor linearity, while point **P** can be used as the start point to find the **IP**. The **IP** can then be used as a reference point for the first segment construction.

2.2.3 Constructing the first line segment ($\mathbf{S}^1$)

After the **IP** is identified, its neighbourhood (for the first line segment, $\mathbf{S}^1$) is obtained such that the ρ satisfies the user requirement. At the same time, it is necessary to make the neighbourhood radius R as large as possible so that the resulted polygon has the minimum number of line segments. Hence, R needs to be determined adaptively. In our approach, we start with a conservatively small value of R and search for the close-to-optimal neighbourhood radius based on the correlation coefficient. A small ρ means poor linearity and thus we need reduce the neighbourhood radius; a large one means good linearity and we can increase the neighbourhood radius. This iterative process is described as follows:

Algorithm *find_neighbourhood_**S**^{1}*

Given a planar data set $\boldsymbol{C}$, the **IP**, initial radius of neighbourhood R, increment of radius ΔR, and the predefined low-bound of correlation coefficient ρ_{low} and high-bound ρ_{high}

(1) Select all the points $\mathbf{P}_i$ from $\boldsymbol{C}$, such that $\|\mathbf{P}_i - \mathbf{IP}\| \le R, \mathbf{P}_i \in \boldsymbol{C}$, to form a data set $\boldsymbol{C}_1$.

(2) Compute the correlation coefficient ρ of data set $\boldsymbol{C}_1$ using Eq. (2.2).

(3) If $\rho > \rho_{\text{high}}$

$R = R + 2\Delta R$, go to step (1)

Else if $\rho < \rho_{\text{low}}$

$R = R - \Delta R$, go to step (1)

Else

Return R and points from $\boldsymbol{C}_1$, stop.

End IF

With these neighbourhood points having good linearity, we can construct a straight line segment that locally fits these points. Here, we use a least-square method to compute a regression line, which passes the **IP(**x_{IP}**,** y_{IP}**)** and best fits the points within the neighbourhood. Let $\boldsymbol{C}_1 = \{\mathbf{P}_i = (x_i, y_i) | \ i = 1, \ldots, N\}$ be the neighbourhood points, a straight line, $\mathbf{L}_1$: $y = a(x - x_{IP}) + y_{IP}$, can be computed by minimizing a quadratic function:

$$\varepsilon = \sum_{i=1}^{N} \left(a\,(x_i - x_{IP}) + y_{IP} - y_i \right)^2 \tag{2.3}$$

As shown in Fig. 2.3, line $\mathbf{L}_1$ has two intersection points, $\mathbf{P}_{start}^{1*}$ and $\mathbf{P}_{end}^{1*}$, with the neighbourhood circle (centred at **IP** with a radius R). In theory, $\mathbf{P}_{start}^{1*}$ and $\mathbf{P}_{end}^{1*}$ can be considered as the start and end points of the first segment. However, they may not be among the points within the neighbourhood. Thus, we select two points, which are the closest to $\mathbf{P}_{start}^{1*}$ and $\mathbf{P}_{end}^{1*}$ respectively, within the neighbourhood, as the start and end points, i.e., the closest to $\mathbf{P}_{start}^{1*}$ as $\mathbf{P}_{start}^{1}$ and the closest to $\mathbf{P}_{end}^{1*}$ as $\mathbf{P}_{end}^{1}$. $\mathbf{S}^1$ is therefore obtained. $\mathbf{P}_{start}^{1}$ $\mathbf{P}_{end}^{1}$ also defines the *unit oriented vector* of this neighbourhood ($\hat{\boldsymbol{s}}_1 = \mathbf{P}_{end}^{1} - \mathbf{P}_{start}^{1} \big) / \left\| \mathbf{P}_{end}^{1} - \mathbf{P}_{start}^{1} \right\|$). Using $\mathbf{P}_{start}^{1}$ $\mathbf{P}_{end}^{1}$ as the diameter, a new neighbourhood circle is obtained, we then delete all the other points within this circle. The remaining planar data set $\boldsymbol{C}$ is also updated.

In the aforementioned procedure for constructing the first line segment, the selection of the initial R plays an important role. This can be illustrated by the example

shown in Fig. 2.4 in which the cloud data represent two linear segments. Starting from the **IP**, if the initial R is too small, e.g., R_1, only a few neighborhood points are included for the first iteration, which gives a poor correlation coefficient. This will lead to the reduction of R and the iteration ends with an even smaller R (with 2-3 points inside). This is certainly not what we want. On the other hand, if the initial R is too large, e.g., R_2, we may have a satisfactory correlation coefficient at the first try, but this may lead to losing the fine corner feature. In our algorithm, we select the initial R such that there are 30-50 data points in this selected region. This generally produces satisfactory result. However, this number also depends on the scanning resolution. In our application, we use a laser scanner with a resolution of 0.001mm. Moreover, if there are fines features on the scanned part, it is assumed that a fine resolution should be used so that there are sufficient data points representing these fine features.

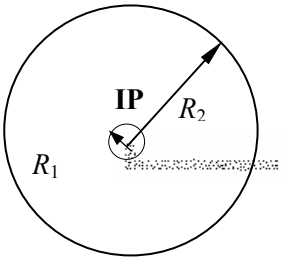


Fig. 2.4: Possible problems with the selection of the initial R

2.2.4 Constructing the remaining segments ($\mathbf{S}^i$)

The method for constructing the remaining segments is slightly different from that of the first segment. We begin with $\mathbf{P}_{end}^1$ as the start point for the second segment, i.e., $\mathbf{P}_{start}^2$. We then adaptively determine the neighbourhood for $\mathbf{S}^2$. Since the same algorithm is used for constructing the remaining segments, we denote the start point as $\mathbf{P}_{start}^i$ ($i \geq 2$). The algorithm to find the radius of the neighbourhood of $\mathbf{S}^i$ is described as follows:

Algorithm *find_neighbourhood_*$\boldsymbol{S}^i$

Given a planar data set $\boldsymbol{C}$, $\mathbf{P}^i_{start}$, initial radius of neighbourhood R, increment of radius ΔR, and the predefined low-bound ρ_{low} and high-bound ρ_{high}

(1) Construct a neighbourhood circle that is centred at $\mathbf{P}^i_{start}$ and has a radius of R. Select all the points $\mathbf{P}_k$ from $\boldsymbol{C}$, such that $\left\|\mathbf{P}_k - \mathbf{P}^i_{start}\right\| \le R$, to form a data set $\boldsymbol{C}_i$ (k = 1, 2, …, n). Compute ρ of data set $\boldsymbol{C}_i$.

If $\rho < \rho_{\text{low}}$

$R = R - \Delta R$, go to step (1)

Else

Use the least-square method (section 2.2.3) to compute a regression line that passes through $\mathbf{P}^i_{start}$. This line has two intersection points with the neighbourhood circle, $\mathbf{O}_1$ and $\mathbf{O}_2$. Let $\mathbf{P}_{ave} = \sum_{k=1}^{n} \mathbf{P}_k / n$. If $\left\|\mathbf{P}_{ave} - \mathbf{O}_1\right\| > \left\|\mathbf{P}_{ave} - \mathbf{O}_2\right\|$, $\hat{\boldsymbol{s}}^*_{\boldsymbol{i}} = (\mathbf{O}_2 - \mathbf{O}_1)/\left\|\mathbf{O}_2 - \mathbf{O}_1\right\|$; otherwise; $\hat{\boldsymbol{s}}^*_{\boldsymbol{i}} = (\mathbf{O}_1 - \mathbf{O}_2)/\left\|\mathbf{O}_2 - \mathbf{O}_1\right\|$.

End IF

(2) Construct a neighbourhood circle that is centred at $\mathbf{P}_c$ ($\mathbf{P}_c = \mathbf{P}^i_{start} + R\,\hat{\boldsymbol{s}}^*_{\boldsymbol{i}}$) and has a radius of R. Select all the points $\mathbf{P}_k$ from $\boldsymbol{C}$, such that $\left\|\mathbf{P}_k - \mathbf{P}_c\right\| \le R$, to form a data set $\boldsymbol{C}_i$. Compute ρ of data set $\boldsymbol{C}_i$.

(3) If $\rho > \rho_{\text{high}}$

$R = R + 2\Delta R$, go to step (4)

Else if $\rho < \rho_{\text{low}}$

$R = R - \Delta R$, go to step (4)

Else

Return $\mathbf{P}^i_{start}$ and $\mathbf{P}^{i*}_{end}$, and all the points from $\boldsymbol{C}_i$, stop.

End If

(4) Use the least-square method (section 2.2.3) to compute a regression line that passes through $\mathbf{P}^i_{start}$. This line has two intersection points with the neighbourhood circle, $\mathbf{P}^i_{start}$ and $\mathbf{P}^{i*}_{end}$. Set $\hat{\boldsymbol{s}}^*_{\boldsymbol{i}} = (\mathbf{P}^{i*}_{end} - \mathbf{P}^i_{start})/\left\|\mathbf{P}^{i*}_{end} - \mathbf{P}^i_{start}\right\|$. Go to step (2).

Since we do not have any *prior* knowledge about the neighbourhood of $\mathbf{S}^i$, i.e., the unit oriented vector $\hat{\boldsymbol{s}}_i$, we need to find a reasonable estimate to start the iterative process. This is achieved in step (1) of Algorithm *find_neighbourhood_*$\boldsymbol{S}^i$. We start by choosing a small *R* to create a neighbourhood circle (centred at $\mathbf{P}^i_{start}$ as shown in Fig. 2.3) such that the points within this circle have a good linearity. We then compute a regression line that passes through $\mathbf{P}^i_{start}$, which helps determine a good estimate of $\hat{\boldsymbol{s}}^*_i$. From step (2), we start with a neighbourhood circle (centred at $\mathbf{P}_c = \mathbf{P}^i_{start} + R\,\hat{\boldsymbol{s}}^*_i$) and adaptively find the maximal allowable neighbourhood radius. The example shown in Fig. 2.3 illustrates this process for the construction of $\mathbf{S}^2$. From the final neighbourhood circle of $\mathbf{S}^2$, $\mathbf{P}^{2*}_{end}$ is obtained. The closest point to $\mathbf{P}^{2*}_{end}$, within this neighbourhood, is then found and used as $\mathbf{P}^2_{end}$. The other point worth mentioning in the above procedure is that in each round, a regression procedure is executed. This may cause long computation time. A trade-off solution is to use the $\hat{\boldsymbol{s}}^*_i$ obtained from step (1) throughout the remaining iterative process so that the computation becomes more efficient.

The outputs from the above procedure are $\mathbf{P}^i_{start}$ and $\mathbf{P}^i_{end}$, and all the points from $\boldsymbol{C}_i$. Using $\mathbf{P}^i_{start}$ $\mathbf{P}^i_{end}$ as the diameter, a new neighbourhood circle is obtained, we then delete all the other points within this circle. The remaining planar data set $\boldsymbol{C}$ is also updated. The above algorithm is then applied to construct $\mathbf{S}^{i+1}$, until the remaining planar data set $\boldsymbol{C}$ is null.

2.3 Adaptive Layer Thickness Determination

Upon the completion of the construction of the polygon curves for the initial layer, the thickness will be adjusted by using the given shape tolerance (ε) as the control parameter. The shape error (σ) of the initial layer is obtained by calculating the distances from all the points in the projection plane to the polygon curve and selecting the maximum distance. If $\sigma < \varepsilon$, the thickness of the initial layer is increased; otherwise, the thickness of the initial layer is reduced. The points within the updated initial layer are then projected to the projection plane. Through the curve construction process described in section 2.2, a new polygon curve is obtained. The shape error is then re-calculated and compared with ε and subsequently a decision is made whether to increase or reduce the thickness of the initial layer. This iteration process is continued until the shape error of the initial layer is slightly less than ε. The construction of the first layer is then completed.

The construction of the subsequent layers is similar to that of the first layer, i.e., (1) creating an initial layer with a pre-set thickness, (2) projecting the data points within the initial layer to a 2D plane, (3) constructing a polygon curve from the data points in the 2D plane, (4) calculating the shape error of the initial layer, and (5) adaptively increasing or reducing the thickness of the initial layer until the shape error is just within ε, e.g., between 0.9ε and ε. In this way, a direct RP model is generated layer by layer adaptively.

For implementation of the aforementioned iterative procedure, a binary search algorithm is developed for finding the thickness of a given layer. This algorithm is described as follows:

Algorithm *find_thickness_layer*

Given an initial layer thickness h (a relative large value) and shape tolerance ε

(1) Set $h_{new} = h$
while (h_{new} < the total height of the data cloud)
{ If $\sigma(h_{new}) < \varepsilon$;
$h_{new} = h_{new} + h$;
Else
Return h_{new}, go to (2);
End if
}

(2) $H_{low} = 0$, $H_{high} = h_{new}$
while ($H_{low} < H_{high}$)
{
$H_{mid} = (H_{low} + H_{high})/2$;
If *band-width* (H_{mid}) > 2ε
$H_{high} = H_{mid}$;
Else if $\sigma(H_{mid}) > \varepsilon$
$H_{high} = H_{mid}$;
Else if $0.9\varepsilon < \sigma(H_{mid}) < \varepsilon$
Return H_{mid};
Else
$H_{low} = H_{mid}$;
End if
}

There are two steps in the above algorithm. In step (1), the search range of the layer thickness, h_{new}, is determined. In the second step (2), a binary search approach is employed. It can be seen that before $\sigma(H_{mid})$ is checked, *band-width* (H_{mid}) is checked first to decide whether to halve the search range. Since the calculation of $\sigma(H_{mid})$ involves 2D polygon construction, the process is computationally heavy. This

checking step is incorporated to avoid the unnecessary calculation of $\sigma(H_{mid})$, thus improving efficiency. The *band-width* (H) is defined as follows: within the 2D projection plane, the projected points of layer H resemble a band, and at any position of the band, there is a band-width (see Fig. 2.3). The polygon generated for representing the band will be within the boundary of the band. Therefore, if the maximum band-width of the band is larger than 2ε, the shape error of the generated polygon will, most likely, be larger than ε. In this case, the search range for the thickness of the layer is halved straightaway.

A simple method is developed to estimate the band-with. We firstly place the 2D points of layer h into $\mathbf{S}_h$. We then obtain another layer between h and $h+\Delta h$ and project its data points to the same plane. The Δh is no less than the minimum layer thickness for the RP machine. The 2D points within Δh are placed into $\mathbf{S}_{\Delta h}$. An example of these two sets of data points is shown in Fig. 2.5. For every point in $\mathbf{S}_h$, $\mathbf{P}_i$ (i = 1, 2, …, n), we then find the corresponding point ($\mathbf{Q}_i$) in $\mathbf{S}_{\Delta h}$, such that among all the points in $\mathbf{S}_{\Delta h}$, $\mathbf{P}_i\mathbf{Q}_i$ gives the minimum distance (L_i). Then, among all the L_i (i = 1, 2, …, n), the maximum one (L_{max}) is taken as the band-width of $\mathbf{S}_h$. Using this estimation method, we find that the points in $\mathbf{S}_h$ that give the maximum distances are along the interior boundary of $\mathbf{S}_h$. Although errors may occur in the estimation of band-width for some points along the interior boundary (e.g., at corners in the example in Fig. 2.5), since we are only interested in L_{max}, we find that L_{max} will never over-estimate the actual band-width. Since we set up 2ε as the low band of the band-width to determine whether to proceed to polygon construction, this estimation suit our algorithm very well.

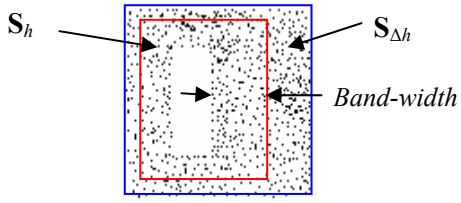


Fig. 2.5: Estimation of the band-width of the 2D data points

Using this method, our algorithm is able to handle cases where very thin layer is needed with relatively good efficiency. For example, when the surface of the part is near-parallel to the slicing plane, the initial relatively large h will result in diffuse points in the 2D plane, or points having a large band-width. With this band-width checking, the algorithm *find_thickness_layer* is able to reduce the search range quickly and proceed to find a satisfactory layer thickness quite efficiently. It is, however, worth noting that when the surface of the part is parallel to the slicing plane, this slicing approach will not work. In this case, a different slicing direction should be chosen manually.

2.4 Summary

In this chapter, a method for generation direct RP models from arbitrarily scattered cloud data is presented. The modelling process consists of several steps: (1) the cloud data are segmented into several layers along the RP building direction, (2) points within each layer is treated as planar data and a polygon is constructed to best-fit the points, (3) the thickness of each layer is determined adaptively such that the surface error is kept just within a given error bound. The algorithm based on this method has been implemented. Test results on both simulated and real cases will be given in Chapter 4 to demonstrate the efficiencies of the algorithm.

The main contribution of this approach is two-fold. Firstly, the polygonal curve construction algorithm is adaptive in nature. It is capable of automatically finding a feasible starting point and identifying the maximum allowable neighbourhood for each segment. It is also able to deal with segments with multiple-loop profile effectively. Secondly, the thickness of each layer is determined adaptively, based on a given surface tolerance. This provides an intuitive control parameter to users and the resultant model a close-to-minimum RP building time.

The main remaining challenging issue is the adaptive determination of the lower and upper linearity bounds in polygonal curve construction. We have observed that this is related to the given shape tolerance and the random errors in the original cloud data.

CHAPTER 3 WAVELETS-BASED ADAPTIVE SLICING

This chapter describes an application of B-spline wavelets for direct RP model construction from cloud data points. The method models the cloud data by adaptive slicing along a user-specified direction, and consists of two main steps.

Firstly, thickness for each layer close to the allowable is determined with the control of band-width of projected points. This estimated band-width is controlled by user-specified shape tolerance. The boundary points between two regions in one layer are extracted and sorted by a tangent-vector based method, which uses a fixed neighbourhood size to quicken the sorting process.

Secondly, for each layer, the profile curve is generated with the wavelets method. Wavelets are applied to the curve construction from the sorted data points from coarser to finer level under the control of shape error.

This approach has better error control and is more robust than ANS-based method because fewer parameters are used. Moreover, the approach is very fast, since fixed neighbourhood size is applied in the sorting process, and fast wavelets decomposition and reconstruction are used in the curve construction, and a parallel algorithm can be used for curve construction in different layers.

3.1 Adaptive Segmentation Approach

In our approach, the cloud data is segmented into a number of layers by slicing the point cloud along a user-specified direction. The data points in each layer are projected onto an appropriate plane and then these projected data points are used to construct a polygon approximating the profile curve.

As shown in Fig 3.1, the problem of segmentation in our application can be stated roughly as follows: Given cloud data **X** and a slicing direction **n**, use a sequence of slicing planes $\mathbf{P}_i$, i=1, 2, …m, which are perpendicular to the direction **n**, to segment **X** into subsets $\mathbf{X}_i$, i=1, 2, …m-1. Each subset $\mathbf{X}_i$, determined by two slicing planes $\mathbf{P}_i$ and $\mathbf{P}_{i+1}$, is then projected onto $\mathbf{P}_i$ to form a planar data set $\mathbf{D}_i$. Thus, the goal of segmentation is to determine these planes $\mathbf{P}_i$, such that there are curves $\mathbf{C}_i$ to fit $\mathbf{D}_i$, and the distance d between $\mathbf{C}_i$ and $\mathbf{D}_i$ is within the user-specified shape tolerance ε.

To solve this problem, there are two approaches to slice the cloud data **X**. One is the uniform slicing and the other is adaptive slicing. The distance between $\mathbf{P}_i$ and $\mathbf{P}_{i+1}$, is called layer thickness $\mathbf{T}_i$. If layer thickness $\mathbf{T}_i$, i=1, 2, …m-1, are uniform, the slicing will be a uniform slicing; otherwise it is an adaptive slicing. These two slicing methods are compared in (Wu et al., 2003).

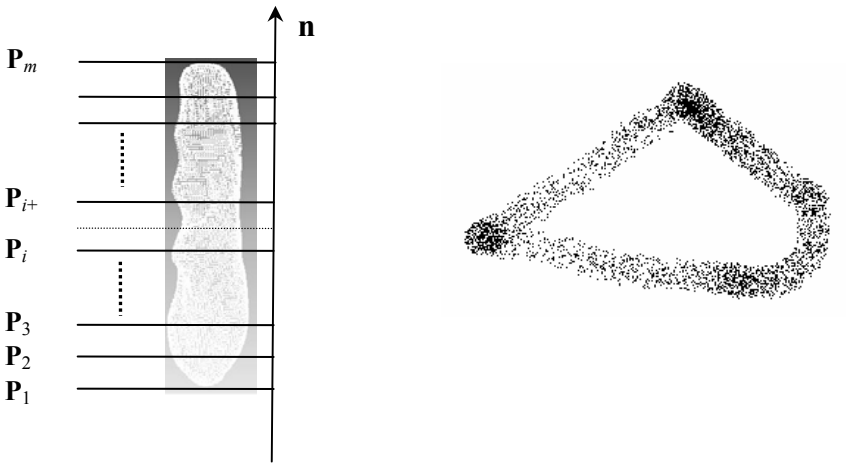
In (Liu 2001, 2002), an adaptive data subdivision algorithm was presented based on subdivision error tolerance ε, specified by the user. In each layer, the subdivision error can be defined as the average distance of these distances between each data point in $\mathbf{X}_i$ and the projecting plane $\mathbf{P}_i$. A method based on binary subdivision was presented to segment the cloud data **X** layer by layer, from one end of **X**, with the control of subdivision error. However, the subdivision error tolerance ε is difficult to determine, because it cannot give a straight relation to the projected data point $\mathbf{D}_i$. If ε is too small, maximal layer thickness is not achieved since $\mathbf{D}_i$ is very thin.

When we slice the object with similar cross-section, we will meet this case. Even worse, $\mathbf{D}_i$ may be discontinous when ε is small enough. On the other hand, if ε is too large, $\mathbf{D}_i$ becomes very thick and it is very difficult to reconstruct the profile curve from $\mathbf{D}_i$.

To address this problem, Wu et al., (2003) presented an adaptive slicing approach to form a relation between layer thickness and shape tolerance directly.

The first slicing plane $\mathbf{P}_1$ is defined by passing the end point of cloud data. To obtain $\mathbf{T}_1$, Wu et al. (2003) used a binary search method, i.e., starting with a large value $\mathbf{T}_1$ and $\Delta\mathbf{T}_1$, get subset $\mathbf{X}_1$ and $\Delta\mathbf{X}_1$, which are projected onto $\mathbf{P}_1$ to form planar set $\mathbf{D}_1$ and $\Delta\mathbf{D}_1$, and estimate band-width as:

$$R = \underset{\mathbf{P}_i \in \mathbf{D}_1}{Max} \left\{ \underset{\mathbf{Q}_j \in \Delta\mathbf{D}_2}{Min} \left\| \mathbf{P}_i - \mathbf{Q}_j \right\| \right\} \tag{3.1}$$



(a) slice cloud data (b) projected data points $\mathbf{D}_i$

Fig. 3.1: Cloud data segmentation

This estimated band width determines the layer thickness to be increased or reduced by half, until $\mathbf{R}$ is close to and less than 2ε. Then, polygon curve $\mathbf{C}_1$ is constructed from $\mathbf{D}_1$ and actual shape error, the distance between $\mathbf{D}_1$ and $\mathbf{C}_1$, is calculated and compared with the shape tolerance ε to decide if the layer thickness should be changed. Then an iteration process for thickness determination is carried on, until a maximum allowable

thickness for the first layer is determined. Similarly, thicknesses for subsequent layers are obtained layer by layer.

However, with a large thickness value, the projected points may be diffuse, and the algorithm of polygon construction may fail to work in this case. Moreover, polygon construction in each round of obtaining layer thickness is computationally expensive, and the band width with a 2ε bound may require a large neighbourhood, which will lead to filtering of small features, such as corners.

In this chapter, we extend the idea of (Wu at el. 2003) and try to overcome these difficulties. The difference is that, for subset $\mathbf{X}_1$, we use a plane $\mathbf{P}_{12}$ to divide $\mathbf{X}_1$ into two sets as $\mathbf{X}_{11}$ and $\mathbf{X}_{12}$, which are projected onto plane $\mathbf{P}_1$ to form two projected data set $\mathbf{D}_{11}$ and $\mathbf{D}_{12}$ respectively, which satisfies:

$$R_1 = \underset{\mathbf{P}_i \in \mathbf{D}_{11}}{Max} \left\{ \underset{\mathbf{Q}_j \in \mathbf{D}_{12}}{Min} \left\| \mathbf{P}_i - \mathbf{Q}_j \right\| \right\} < \varepsilon \tag{3.2}$$

$$R_2 = \underset{\mathbf{P}_i \in \mathbf{D}_{12}}{Max} \left\{ \underset{\mathbf{Q}_j \in \mathbf{D}_{11}}{Min} \left\| \mathbf{P}_i - \mathbf{Q}_j \right\| \right\} < \varepsilon \tag{3.3}$$

With these two constraints, boundary points $\mathbf{B}_1$ can be extracted from $\mathbf{D}_{11}$ and $\mathbf{D}_{12}$ as:

$$\mathbf{B}_1 = \left\{ \mathbf{P} \mid \underset{\mathbf{Q}_j \in \mathbf{D}_{12}}{Min} \left\| \mathbf{P} - \mathbf{Q}_j \right\| \right\} and \quad \mathbf{P} \in \mathbf{D}_{11} \tag{3.4}$$

We use $\mathbf{B}_1$ to construct profile curve to approximate $\mathbf{D}_1$. Since $\mathbf{B}_1$ is thinner than $\mathbf{D}_1$, small neighbourhood size can be used which can recover sharp corners. Similarly, subsequent layers $\mathbf{X}_i$ and its corresponding projected data set $\mathbf{D}_i$ and boundary points $\mathbf{B}_i$ are obtained with equations (3.2), (3.3) and (3.4). We assume that $\mathbf{B}_i$ has the same topology as $\mathbf{D}_i$, and assume that $\mathbf{B}_i$ lies in the middle of $\mathbf{D}_i$. This assumption is not rigorous in theory, but in practise, it does work. In our simulation and real cases, we find that $\mathbf{B}_i$ approximate $\mathbf{D}_i$ well. Details of curve construction from planar points will

be discussed in next section. In the example of Fig. 3.1b, the $\mathbf{B}_i$ is obtained as shown in Fig. 3.2.

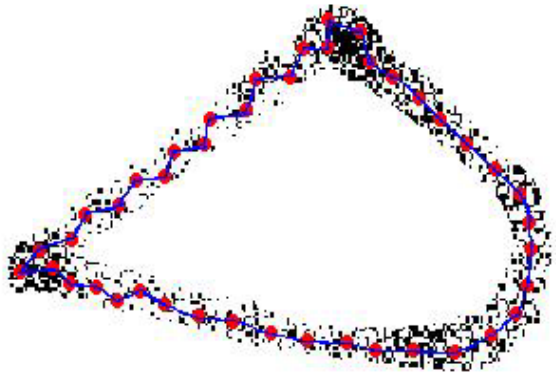


Fig. 3.2: Boundary points

After rough slicing, the cloud data $\mathbf{X}$ is segmented into subsets $\mathbf{X}_i$ and its projected points $\mathbf{D}_i$ and boundary points $\mathbf{B}_i$ are obtained. Then, contour curve construction is carried out from $\mathbf{D}_i$ and $\mathbf{B}_i$ layer by layer. Actual shape errors are obtained from the largest distance between $\mathbf{D}_i$ and its curve, and the comparison of actual shape error and shape tolerance will determine whether fine slicing need to carry out or not. Details will be given in the following sections.

3.2 Polygonal Curve Construction from Cloud Data

After rough slicing, we obtain subset points and their corresponding projected point set. The next step is to construct one or several closed polygon curves to accurately represent the shape defined by these points. Since each polygon curve is closed, these polygon curves are constructed one by one and different curves (in the case of multiple-loop) are split naturally. Here, we only discuss the single loop curve construction problems.

The problem of curve construction from planar points can be stated as follows: Given planar data point set **D** that lie on an unknown curve **UC**, create a curve **C** to approximate **UC**, such that the constructed curve **C** should have the same topological type as **UC**, and can be every where close to **UC**.

In our application, there are few assumptions about the planar points **D**. The points **D** may be noisy, and no structure or other information is assumed within them. The shape of unknown curve **UC** may have arbitrary topological type, including sharp features such as corners.

To reconstruct a curve from a 2D data set, two different methods can be used, i.e., interpolation and approximation (Amidror 2002). Interpolation methods approximate the underlying function by finding a curve that passes through the known data points and these data points are highly accurate, reliable and sparse. However, data fitting methods find an approximate curve that passes close to the known value but not necessarily exactly through them, and these points are dense with relatively high noise. From this aspect, we focus on data fitting in our application.

There are many approaches for curve construction from organized data points, however little research work devotes to curve construction from unorganized data points. This paper focuses on data fitting from unorganized data points.

Different approximation approaches are presented, which can be classified into global methods and local methods. Global methods assume that the unknown curve **UC** is a continuous curve, and usually a least square method is used to achieve the approximated curve **C** to fit **UC** (Taubin and Ronfard 1996). Obviously, global methods are not suitable to unknown curves with an arbitrary topology. Local fitting methods use piecewise curves to fit the “nearby points” piece by piece, such that a

complex shape can be approximated. In our application, we focus on local data fitting method.

Apparently, how to decide "nearby points", i.e., neighbourhood points, such that a piecewise curve can fit to these points, is an important and difficult issue in local data fitting. Two general methods can be used. One is the fixed neighbourhood size, i.e., the number of neighbourhood points is specified by user, and neighbourhood points are determined by calculating the distance between neighbourhood point and a related point and selecting the points according to the distance from small to large (Liu 2001). Fixed neighbourhood size can give a fast computation, however it causes severe problems in practise. As show Fig 3.3, when the number of data points is too small, the constructed curve will be of zigzag or even has over-fitting problem. On the other hand, if the number is too large, unwanted points will be included, and small features such as corners of the curve are filtered, thus topology of the curve is different with that unknown curve.

(a) neighbourhood number too small (b) neighbourhood number too large

Fig. 3.3: Problems caused by fixed neighbourhood point's number

The other method is to use adaptive neighbourhood size (Lee 2000, Wu et al., 2003). Both of them used correlation coefficient of neighbourhood points to decide the neighbourhood size. In (Lee 2000), triangulating the given data set prior to the actual

curve fitting is necessary, and line segment is rotated iteratively during process of curve fitting. While in (Wu et al. 2003), triangulation is not needed, and line segment orientation is determined by previously selected points. However, there are some problems in both of them.

In the procedure for constructing the line segment, the selection of the initial neighbourhood size R is a difficult issue. This can be illustrated by the example shown in Fig. 2.4 in which the cloud data represent two linear segments. Starting from the **IP**, if the initial R is too small, e.g., R_1, only a few neighborhood points are included for the first iteration, which gives a poor correlation coefficient. This will lead to the reduction of R and the iteration ends with an even smaller R (with 2-3 points inside). This is certainly not what we want. On the other hand, if the initial R is too large, e.g., R_2, we may have a satisfactory correlation coefficient at the first try, but this may lead to losing the fine corner feature. Similar cases are met to decide the increased neighborhood size ΔR. If ΔR is too large, sharp corner is easy to be missed, thus fast algorithm such as binary search method cannot be used to determine the final neighborhood size.

In (Wu et al. 2003), to solve this problem, a tradeoff is obtained that the initial neighborhood R is to select such that the number of data points in this selected region is no less than a user specified value. This generally produces satisfactory result. However, this number depends on not only the scanning resolution but also the complexity of the unknown curve shape. Hence, it is difficult to determine. Moreover, this user specified value together with user specified correlation coefficient bounds would make the algorithm not robust. If the bound is to strict, the actual correlation coefficient is out of the bounds, while at the same time the points number reaches the

user specified tolerance. Hence these two conflicted conditions cause the algorithm breakdown.

Here a multiresolution method based on wavelets is used to construct the polygon curve. We state our goal of curve construction here: Given planar data point set **D**, 2D medial point set **B**, to construct a curve **C** from **B** such that $\|\mathbf{D}-\mathbf{C}\| < \varepsilon$ and **C** is concise.

In the fit-and-fair method, there are two progressive errors: one is the shape error between planar data points and fitting curve, the other one is the shape error between faired curve and fitting curve. However, the control of these two progressive errors cannot guarantee that the final error between data points and final curve is within tolerance. Hence, we present an algorithm based on wavelets to determine a curve with certain fairness from planar data points directly.

Mainly there are two steps. Firstly, sorting method is applied to sort the medial point set B. We use the same algorithm of (Wu et al. 2003), except that we do not use adaptive neighbourhood size. From section 3.1, we know that the band-width of **B** is less than shape tolerance ε. Hence, we can use a fixed neighbourhood size as shape tolerance ε, and there will be no self-intersection problems. The algorithm will be robust, since we need not use the user-specified initial neighbourhood size, nor the correlation coefficient bounds. Also, sharp corners can be kept because small neighbourhood size is used. However, as we mentioned above, fixed neighbourhood size can cause zigzag shape, and the small size will also give a dense sorted data points. Thus, further processing will be needed.

Secondly, a multiresolution method is applied to decompose the sorted data points into coarser levels, and then polygon reconstructed from coarse-to-fine level under the control of shape tolerance. Details are given in following sections.

A multiscale technique for shape representation has been developed based on wavelets (Chung and Kuo 1996, Chung 2000). In Computer Graphics, wavelet methods are developed for the multiresolution representation of parametric curves and surfaces, and it is mainly used as a powerful tool for curve and surface hierarchical design. Wang et al. (1999) presented a multiscale curvature-based shape representation using B-spline wavelets. This wavelet transforms are used to efficiently estimate the multiscale curvature functions. Based on the curvature scale-space image, they introduced a coarse-to-fine matching algorithm which automatically detects the dominant points and uses them as knots for curve interpolation. Esteve et al. (2001) presented a multiresolution method for implicit curves and surfaces based on wavelets to simplify the topology.

However the above mentioned methods for curve representation started with a source of such data in digital ordered form. To represent a curve from unorganized scattered data points based on wavelets is the interest in our application. Moreover, the curve construction is under the control of shape error, and data reduction method is necessary to be integrated into wavelets based curve construction.

3.2.1 Wavelets and Multiresolution Analysis

Wavelet transform is a highly developed technique used in several fields such as signal processing, image processing, communications, computer graphics and mathematics. In computer graphics, it can deal with the representation and manipulation of geometric shapes such as curves and shapes suitable for hierarchical design. This section reviews some basic concepts and methods of wavelet transformation and multiresolution analysis as applicable to the development of our application.

Comprehensive study of the subject can be found in Wavelets texts such as (Daubechies 1992, Chui 1992, Stollnitz et al., 1996).

(a) Multiresolution analysis (MRA)

A **MRA** involves approximation of functions in nested subspaces of a linear vector space $\mathbf{V}^j$ such that

$$\{0\} \longleftarrow \ldots \mathbf{V}^{-2} \subset \mathbf{V}^{-1} \subset \mathbf{V}^0 \subset \mathbf{V}^1 \subset \mathbf{V}^2 \ldots \subset \mathbf{V}^j \ldots \longrightarrow \mathbf{L}^2 \tag{3.5}$$

The basis functions of $\mathbf{V}^j$ are called scaling functions at the resolution level j and denoted by ϕ_i^j (i=1, 2, …). With the increase of level j, ϕ_i^j represents a finer level of the function.

Let the subspace $\mathbf{V}^n$ constitute a direct sum (denoted as $\oplus$) of mutually orthogonal subspaces $\mathbf{W}^j$ as:

$$\mathbf{V}^n = \mathbf{W}^{n-1} \oplus \mathbf{W}^{n-2} \oplus \mathbf{W}^{n-3} \oplus \ldots\ldots \mathbf{W}^{-\infty} \tag{3.6}$$

Then we can get

$$\mathbf{V}^{n+1} = \mathbf{V}^n \oplus \mathbf{W}^n \tag{3.7}$$

The subspace $\mathbf{W}^n$ (the wavelet subspace) is said to be the orthogonal complementary subspace of $\mathbf{V}^n$ in $\mathbf{V}^{n+1}$. The basis functions of $\mathbf{W}^j$ are called wavelets at the resolution level j and denoted by ψ_i^j (i=1, 2, …). A function $f^j(u)$ at scale j is represented by the linear sum of the scaling functions $\phi_i^j(u)$, i.e., $f^j(u) \in \mathbf{V}^j$ and $f^j(u) = \sum_{i=-\infty}^{+\infty} p_i \phi_i^j(u)$.

Since $\mathbf{V}^j = \mathbf{V}^{j-1} \oplus \mathbf{W}^{j-1}$, $f^j(u)$ can be decomposed into $f^{j-1}(u) (\in \mathbf{V}^{j-1})$ and $g^{j-1}(u) (\in \mathbf{W}^{j-1})$.

(b) End-point B-spline Wavelets

We employ the cubic endpoint-interpolating B-spline blending functions that are defined on a uniformly spaced knot sequence as the scaling functions, because they can describe a smooth curve and are refinable. In practice, they are well-understood and used (Chui, 1992). These functions have a compact support, and the vector space has a limited number of basis functions. Hence, we can decompose and reconstruct multiresolution functions using matrix calculations (Stollnitz et al., 1996). Because the endpoint-interpolating B-spline blending functions and its corresponding wavelets are defined in bounded domains, a function $f^j(u) \in \mathbf{V}^j$ is represented by a limited number of scaling functions (blending functions) as

$$f^j(u) = \mathbf{c}_0^j \phi_0^j(u) + \mathbf{c}_1^j \phi_1^j(u) + ... \mathbf{c}_{m^j}^j \phi_{m^j}^j(u)$$

$$= [\phi_0^j(u) .... \phi_{m^j}^j(u)] \bullet [\mathbf{c}_0^j ... \mathbf{c}_{m^j}^j]^T = \phi^j(u) \bullet \mathbf{c}^j \tag{3.8}$$

Where $\phi^j(u)$ is a row matrix of scaling functions and coefficients $\mathbf{c}^j$ form a column matrix of the scaling coefficients (control points), and m^j is the dimension of $\mathbf{V}^j$. Similarly, we can also represent a function $g^j(u) \in \mathbf{W}^j$

$$g^j(u) = \psi^j(u) \bullet \mathbf{d}^j \tag{3.9}$$

With $\psi^j(\boldsymbol{u})$ is a row matrix of wavelet basis functions and n^j is the dimension of $\mathbf{W}^j$ and detail coefficients $\mathbf{d}^j$ is a matrix of column. Because $\mathbf{W}^j$ is the complement of $\mathbf{V}^j$ in $\mathbf{V}^{j+1}$, the dimensions of these spaces satisfy $m^{j+1}=m^j+n^j$ .The subspaces $\mathbf{V}^j$ is nested thus it is equivalent to having scaling function that are refinable. That is for all j=1, 2, …, there must exist a matrix of constants $\mathbf{P}^j$ such that

$$\phi^{j-1}(u) = \phi^j(u)\mathbf{P}^j \tag{3.10}$$

Obviously, $\mathbf{P}^j$ is a $m^j \times m^{j-1}$ matrix. Similarly, since $\mathbf{W}^j$ is also a subspace of $\mathbf{V}^j$, a matrix $\mathbf{Q}^j$ can also be found that satisfies:

$$\psi^{j-1}(u) = \psi^{j}(u)\mathbf{Q}^{j} \tag{3.11}$$

and $\mathbf{Q}^j$ is a $m^j \times n^{j-1}$ matrix. Equations (3.2) and (3.3) are said to be two-scale relations for $\phi^j(u)$ and $\psi^j(u)$, respectively, and the matrices $\mathbf{P}^j$ and $\mathbf{Q}^j$ are called synthesis filters. The following shows the synthesis filters at level j=2.

$$\mathbf{P}^2 = \frac{1}{16}\begin{bmatrix} 16 & 0 & 0 & 0 & 0 \\ 8 & 8 & 0 & 0 & 0 \\ 0 & 12 & 4 & 0 & 0 \\ 0 & 3 & 10 & 3 & 0 \\ 0 & 0 & 4 & 12 & 0 \\ 0 & 0 & 0 & 8 & 8 \\ 0 & 0 & 0 & 0 & 16 \end{bmatrix} \qquad \mathbf{Q}^2 = \frac{1}{2064}\begin{bmatrix} -1368 & 0 \\ 2064 & 240 \\ -1793 & -691 \\ 1053 & 1053 \\ -691 & -1793 \\ 240 & 2064 \\ 0 & -1368 \end{bmatrix}$$

(c) Wavelet decomposition and reconstruction

If we wish to create a low-resolution version $\mathbf{c}^{j-1}$ of $\mathbf{c}^j$ with a smaller number of coefficients n^{j-1}, there is a matrix $\mathbf{A}^j$ that can perform such a function that:

$$\mathbf{c}^{j-1} = \mathbf{A}^{j}\mathbf{c}^{j} \tag{3.12}$$

Where $\mathbf{A}^j$ is a constant $m^{j-1} \times m^j$ matrix. Since $\mathbf{c}^{j-1}$ contains fewer entries than $\mathbf{c}^j$, it is intuitively clear that some amount of detail is lost in this filtering process. Thus the lost information $\mathbf{d}^{j-1}$ can be obtained with a matrix $\mathbf{B}^j$ such that:

$$\mathbf{d}^{j-1} = \mathbf{B}^{j}\mathbf{c}^{j} \tag{3.13}$$

Where $\mathbf{B}^j$ is a constant $n^{j-1} \times m^j$ matrix. The pair of matrix $\mathbf{A}^j$ and $\mathbf{B}^j$ are called analysis filters. The process of splitting the coefficients $\mathbf{c}^j$ into a low-resolution version $\mathbf{c}^{j-1}$ and detail $\mathbf{d}^{j-1}$ is called decomposition. On the other hand, the original coefficients $\mathbf{c}^j$ can be recovered from $\mathbf{c}^{j-1}$ and $\mathbf{d}^{j-1}$ by using the matrices $\mathbf{P}^j$ and $\mathbf{Q}^j$ as:

$$\mathbf{c}^{j} = \mathbf{P}^{j}\mathbf{c}^{j-1} + \mathbf{Q}^{j}\mathbf{d}^{j-1} \tag{3.14}$$

Recovering $\mathbf{c}^j$ from $\mathbf{c}^{j-1}$ and $\mathbf{d}^{j-1}$ is called synthesis or reconstruction. According to equations (3.12) (3.13) and (3.14), we can derive the following relation:

$$\begin{bmatrix} \mathbf{A}^j \\ \mathbf{B}^j \end{bmatrix} = [\mathbf{P}^j \mid \mathbf{Q}^j]^{-1} \tag{3.15}$$

With these formulations, we can directly apply them to the multiresolution representations of curves. Hence, we can decompose and synthesis the endpoint interpolating B-spline curves at different scales.

3.2.2 Polygonal curve construction from cloud data based on wavelets

There are many methods for curve fitting from cloud data, however the fitting curve will be of zigzag or self-intersected, which need further fair. But most fairing methods will cause the difference between faired curve and original planar data points out of shape tolerance. Multiresolution based on wavelets can give a good locality which helps to decompose curves into lower levels. However, using wavelets to construct a smoothed curve directly from cloud data is still a challenging issue.

In our application of cloud data modelling, using a preprocessing method, we can obtain the sorted data point set **S** from medial point set **B**. From this stage, our goal becomes: Given sorted data point **S** and projected data point set **D**, to construct a concise curve **C** to fit **D** such that $\|\mathbf{D}-\mathbf{C}\| < \varepsilon$.

(a) Curve decomposing.

Wavelets can be used to represent parametric curves simply by computing the wavelet decomposition of each coordinate function separately. For a planar curve $\mathbf{C}(u)=(x(u),y(u))$, $\mathbf{C}(u)$ is a vector-valued function of the independent variable, u, then we can decompose its $x(u)$ and $y(u)$ function separately. Here, we denote the scaling coefficients **c** and detail coefficients **d** as the vector-valued decomposed coefficients of the planar curve.

To decompose the curve, the basic idea is to assume this initial data point set **S** as control points $\{\mathbf{c}^j\}$ of a curve at level j. We thus use a end-point interpolating B-spline as scaling basis and B-spline wavelets to decompose this initial curve into lower levels at j, j-1… until level r (suppose). Scaling coefficients and detail coefficients are obtained in each level decomposing as shown in Fig 3.4.

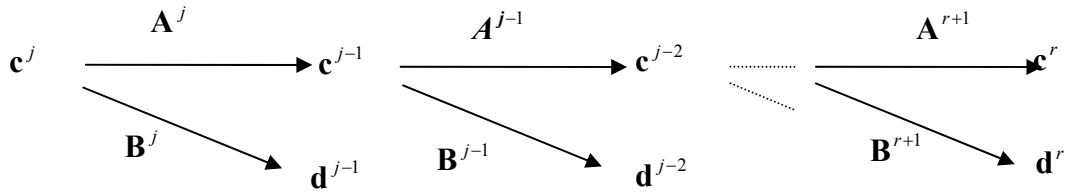


Fig. 3.4: Wavelets decomposition

To store these scaling and detail coefficients, we can use a data set with constant size. From section 3.1.1, we know that the dimension of scaling coefficients equals to the dimension of scaling coefficients plus that of detail coefficients at its lower level. In a higher level, the dimension of scaling coefficients is nearly two times as that of its neighbouring lower level.

At finer scales, we can obtain a good localization of feature points such as corner points. But due to the influence of noise or high frequency details, the spurious points are included. At coarser scale, small features together with spurious points are filtered, then we can obtain the overall picture of the curve, but the differences between the curve and the original planar data points are larger due to the smoothing procedure. Therefore, we have to combine the multiscale information to extract the scaling coefficients in different levels. In coarser level, if a line segment constructed by the two scaling coefficients meet shape tolerance, we will extract these two coefficients,

and the portion of planar points related to this line segment is approximated by this line segment. Then we continue to extract scaling coefficients from coarser to finer.

Our method is different from that for curve compression. For curve compression, at a certain level, a standard procedure is to set all wavelet coefficients below a given threshold value to zero, but this kind of removal wavelets coefficients only give a least square sense error (Chui, 1992). However, the threshold has no meaning with shape error. Hence, the threshold is not so easy to be determined by users, especially for different levels, that the thresholds have to be different. We extend the idea of coarse-to-fine (Wang et al., 1999) to extract segments from coarser level to finer level by the shape error control. Since we need construct a curve from cloud data, we use different error checking and data reduction method. And to quicken the process of error calculation, we try to use scaling coefficients to approximate the curve, not B-spline curve segments as they mentioned. The details will be presented in the following sections.

(b) Extract scaling coefficients at the coarsest level

After decomposition, we use the scaling coefficients to the approximate original planar data points. Before we extract the scaling coefficients, we need to identify the storable scaling coefficients. Given scaling coefficient $\mathbf{c}_i$ and its two neighbour coefficients $\mathbf{c}_{i-1}$ and $\mathbf{c}_{i+1}$, $\mathbf{c}_i$ is termed as a storable point associating with the planar projected data points $\mathbf{P}$ when the following criterion is satisfied for every data point $\mathbf{P}_{ci}(j)$ in $\mathbf{P}_{ci}$:

$$\min\{|\mathbf{P}_{ci}(j)-\mathbf{c}_{i-1}\mathbf{c}_i|,|\mathbf{P}_{ci}(j)-\mathbf{c}_i\mathbf{c}_{i+1}|\}\le\varepsilon \tag{3.16}$$

Where ε is the shape tolerance and the data point set $\mathbf{P}_{ci}$, is formed from $\mathbf{P}$, as the distance between each point $\mathbf{P}_{ci}(m)$ in $\mathbf{P}_{ci}$ and $\mathbf{c}_i$ is the shortest distance among those between $\mathbf{P}_{ci}(m)$ and all scaling coefficients $\mathbf{c}_k$ (k=1,2,..), namely,

$$\left|\mathbf{P}_{ci}(m)-\mathbf{c}_i\right| = \min_{c_k \in c}\left|\mathbf{P}_{ci}(m)-\mathbf{c}_k\right| \tag{3.17}$$

Obviously, if two neighbouring scaling coefficients $\mathbf{c}_i$ and $\mathbf{c}_{i+1}$ are storable, and also their other neighbouring coefficients $\mathbf{c}_{i-1}$ and $\mathbf{c}_{i+2}$ are storable, we can link $\mathbf{c}_i$ and $\mathbf{c}_{i+1}$ into a line segment and hence the coefficients $\mathbf{c}_i$ and $\mathbf{c}_{i+1}$ are extracted as the desired points as we need.

Thus, the scaling and detail coefficients at higher levels corresponding to these two coefficients can be flagged, and this issue will be discussed further in the next section. Also, to reduce the calculation time, we delete data points from the planar data set that is nearest to this line segment. For every point $\mathbf{P}_{ci,ci+1}(j)$ in data set $\mathbf{P}_{ci}$ and $\mathbf{P}_{ci+1}$, as $\mathbf{P}_{ci}$ and $\mathbf{P}_{ci+1}$ are obtained with $\mathbf{c}_i$ and $\mathbf{c}_{i+1}$ based on equation (3.17), if this point satisfies

$$\left|\mathbf{P}_{ci,ci+1}(j)-\mathbf{c}_i\mathbf{c}_{i+1}\right| \le \varepsilon \tag{3.18}$$

Where ε is the shape tolerance, then we can delete data point $\mathbf{P}_{ci,ci+1}(j)$ from planar data set $\mathbf{P}$.

Similarly, when we identify more than 3 continuous scaling coefficients which are storable, we can extract the inner scaling coefficients within these scaling coefficients (two end points are not considered). We call this case as Multi-Storable-Extracting (MSE). Within in MSE, we will reduce the data points from planer data set by one and one line segment with equation (3.18). To store the extracted coefficients, we can use a global dynamic list to save the coefficients and its corresponding spatial index.

The process for extracting scaling coefficients on coarsest level, i.e, level r is described as follows:

Algorithm *extract_scalingcoefficients* _$\mathbf{c}^r$

Given a planar data set $\mathbf{P}_i$,(i=0, 1,…,N) the scaling coefficient set $\mathbf{c}_j$,(j=0, 1,…,2^r+3) shape tolerance ε, the algorithm extracts scaling coefficients into a global list GList;

1. Find the first sacling coefficients $\mathbf{c}_i$ that is storable with index i.
   If no storable scaling coefficients existed in **c**
   go to step 4
   Else
   k=1
   go to step 2
   EndIf
2. While(i=i+1<2^r)
   {
   If $\mathbf{C}_i$ is storable
   k=k+1
   Else if (k>3)
   a. j=i-k+1
   Extract scaling coefficients $\mathbf{c}_j$ and store it into Glist with its index j
   j=j+1
   b. Extract scaling coefficients $\mathbf{c}_j$ store them into Glist together with its index j
   Reduce data points around line segment $\mathbf{c}_{j-1}\mathbf{c}_j$ from P based on equation (3.17)
   c. j=j+1, go back to step b, so long as j<i-2

   k=0
   Else
   k=0
   EndIf
   }
3. If (k>3)
   d. j=i-k+1
   Extract scaling coefficients $\mathbf{c}_j$ and store it into Glist with its index j
   j=j+1
   e. Extract scaling coefficients $\mathbf{c}_j$ store them into Glist together with its index j
   Reduce data points around line segment $\mathbf{c}_{j-1}\mathbf{c}_j$ from **P** based on equation(3.17)
   f. j=j+1, go back to step e, so long as j<i-2
   Else
   go to step 4
   End If
4. Terminate

With this method, we can extract some MSEs, such as $\{\boldsymbol{c}_i, \mathbf{c}_{i+1}, \ldots\mathbf{c}_{i+k-1}\}$ as an k number MSE, and at the same time, we can reduce data points from planar data point set. As shown in Fig. 3.5, there are three MSE segments consisted of storable points, at

coarsest level, and in this case the level is 5 (there are 2^5+3 scaling points). These scaling coefficient points are stored in a data set as Fig. 3.8 a shows, and the MSEs are easily recognized.

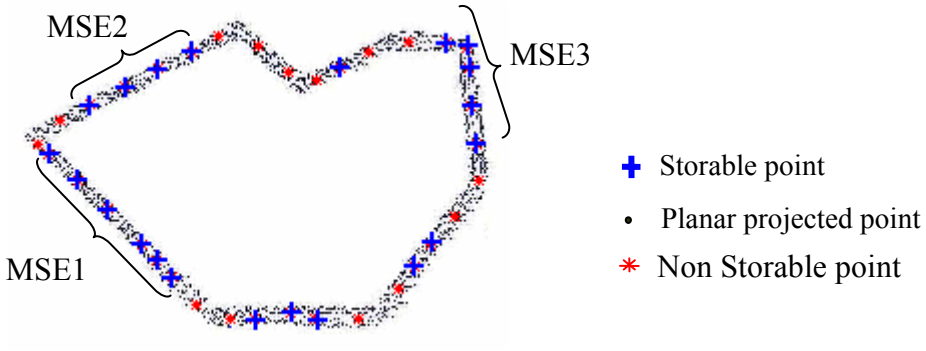


Fig. 3.5: Extracting scaling coefficients at coarsest level

(c) Extract scaling coefficients at the remaining levels

The method for extracting scaling coefficients of the remaining levels is slightly different from that of the coarsest level. To perform extracting scaling coefficients in higher level, we firstly need to reconstruct these scaling coefficients from the coarser level scaling coefficients and detail coefficients. This process can be shown in Fig. 3.6. With synthesis filters **P** and **Q**, we can construct the scaling coefficients in an efficient way.

However, in our application, the smooth section of planar data points is approximated by the extracted scaling coefficients in coarser level, so the corresponding portion of this section in finer level need not be recalculated, i.e., the smooth section approximated by scaling coefficients that extracted in coarser levels need remain fixed in finer levels.

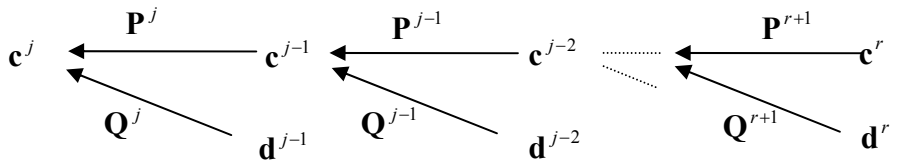


Fig. 3.6: Wavelets reconstruction

In coarser level, we extract some MSEs, which can approximate the portions of planar points within the shape tolerance. Hence, we need fix these MSEs in their corresponding portions in the finer levels. For example, a MSE at level j is $\{\mathbf{c}_i^j, \mathbf{c}_{i+1}^j, \ldots \mathbf{c}_{i+k-1}^j\}$, after reconstruction with the synthesis matrixes $\mathbf{P}^{j+1}$ and $\mathbf{Q}^{j+1}$, we get $\{\mathbf{c}^{j+1}\}$ at level j+1 to replace the whole scaling coefficients $\{\mathbf{c}^j\}$ and the detail coefficients $\{\mathbf{d}^j\}$ at level j, and this MSE has a corresponding area as $\{\mathbf{c}_{2i}^{j+1}, \mathbf{c}_{2i+1}^{j+1}, \mathbf{c}_{2(i+1)}^{j+1}, \mathbf{c}_{2(i+1)+1}^{j+1} \ldots \mathbf{c}_{2(i+k-1)}^{j+1}, \mathbf{c}_{2(i+k-1)+1}^{j+1}\}$, this updated MSE can be approximated by the old MSE, since the old one satisfies the shape tolerance as we checked in level j. Fig. 3.6 shows the updated MSE from the MSE in Fig. 3.5. We need use the coefficients in the old MSE to replace the coefficients in the new MSE. After replacing, we can use the similar algorithm to extract new scaling coefficients as what presented in section 3.1.2. Fig. 3.7 shows some new MSE. However, two issues we need pay attention. First, to reduce computation time, we need not carry on recalculating inner scaling coefficients in the updated MSE. Second, we need store the spatial indices of scaling coefficients, so that the indices of scaling coefficients at any levels have a global sequence. To address these problems, we use the updated MSEs to divide the scaling coefficients into some sections, and each section needs include the end scaling coefficient of its neighbouring updated MSE or MSEs. We call these sections as Non-Updated MSE (NUMSE).

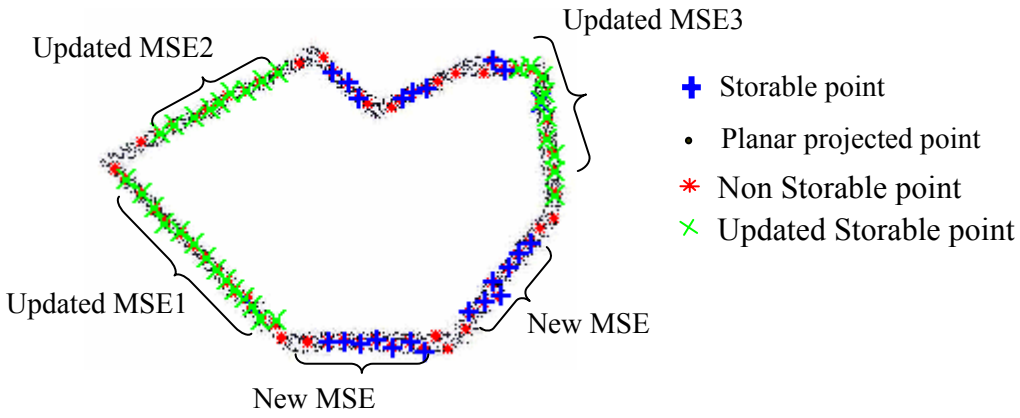


Fig. 3.7: Finer level of a decomposition curve

With these NUMSEs, we can directly use the similar algorithm as mentioned above. But we still need care about the indices problem as above mentioned. We use a global list GList to store the extracted coefficients and their corresponding indices. Reconstruction from a level to one level higher, the number of scaling coefficients will be doubled. Hence we need double the index of scaling coefficients in the GList so that the stored coefficients extracted at coarser level have fixed indices compared to the extracted coefficients at the higher level. Fig. 3.8 a shows three MSEs at level j, and Fig. 3.8b shows the reconstructed scaling coefficients at level j+1, and there are three updated MSEs. Fig. 3.8c shows the four NUMSEs and Fig. 3.8d shows two new MSEs extracted from NUMSEs. Fig. 3.8 illustrates data structure of the scaling points in Fig. 3.6 and Fig. 3.7.

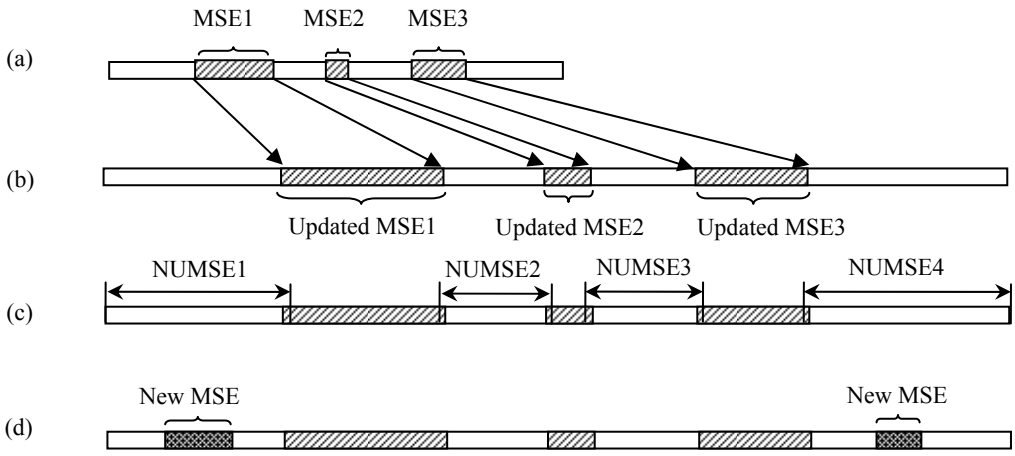


Fig. 3.8: Scaling coefficients extracting at finer level

The main steps for extracting scaling coefficients on the next finer level, i.e., level j+1 is described as follows:

(1) Reconstruct scaling coefficients $\mathbf{c}^{j+1}$ from $\mathbf{c}^{j}$ with equation (3.14)

(2)With the index of MSEs in $\mathbf{c}^{j}$, find out their corresponding updated MSEs in $\mathbf{c}^{j+1}$,

(3)Using these updated MSEs, get NUMSEs

(4)For each existed element (extracted scaling coefficients at level j) of global GList, update its index with double its value

(5)For each NUMSEs, using algorithm presented in section 3.2 to extract New MSEs

To store the extracted coefficients of a New MSE, we need compute the number of the elements in MSEs that occurs before this New MSE.

After extracting scaling coefficients $\mathbf{c}^{j+1}$, we reconstruct the scaling coefficients $\mathbf{c}^{j+2}$, and use the New MSEs and Updated MSEs at level j to get Updated MSEs at level j+2. The above algorithm is then applied to extract the coefficients at higher level, until

there is no NUMSE at a level. Finally, we can sort the scaling coefficients in the global list according to their indices value. Thus the whole polygonal curve is obtained.

3.3 Adaptive Layered-based Direct RP Model Construction

In our application to construct models for RP, the cloud data is segmented into a number of layers by slicing the point cloud along a user-specified direction. The data points in each layer are projected onto the mid-plane and then these projected data points will be used to reconstruct a polygon approximating the profile curve. Shape tolerance is usually given to indicate the maximum allowable deviation between the generated model and the cloud data points.

In our application, there are two main steps for layer-based Direct RP model construction. The first step is to roughly slice the cloud data object. From an end of the object, we use a binary search (Wu et al. 2003) to obtain a subset $\mathbf{X}_{11}$ such that equation (3.2) is satisfied. Then a subset $\mathbf{X}_{12}$ can be obtained with the binary search algorithm such that equation (3.3) is satisfied. With these two subsets and their projected point set $\mathbf{D}_{11}$ and $\mathbf{D}_{12}$, extract their medial points based on equation (3.4). Hence, the first layer is obtained. Similarly, the subsequent layers are obtained. With rough slicing, the cloud data is sliced into layers with adaptive layer thickness and medial points in each layer.

In the second step, we carry on polygonal curve construction algorithm based on the multiresolution method as presented in this chapter. Medial points are firstly sorted with a small fixed neighbourhood radius, and then B-spline wavelets are used to extract the polygon points in sense of filtering small noise of sorted medial points, and the extracting process is a coarse-to-fine process under the control of shape tolerance. Thus with these two steps, the polygon for each layer is obtained.

However, one drawback should be mentioned in our approach. Using wavelets to filter small noise of sorted points may lead to the divergence of the algorithm. To state clearly, we use the first layer as an example. In theory, the medial points $\mathbf{B}_1$ lies in medial of projected points $\mathbf{D}_1$ because of the control of bandwidth tolerance ε, but after multi-resolution extracting of sorted point set $\mathbf{S}_1$, the distance between $\mathbf{D}_1$ and final polygon $\mathbf{C}$ may be out shape tolerance ε. To solve this problem, an easy way is to restrict the bandwidth tolerance, say as $0.7\sim0.9\varepsilon$ in most of our case studies. Thus, with this stricter bandwidth tolerance, the medial points B_1 will lie in a thinner projected $\mathbf{D}_1$, thus the algorithm will converge. However, stricter bandwidth will cause that the layer thickness is less than maximal. To resolve this confliction, we need tradeoffs.

Assume that the unknown object is T of thickness along slicing direction, and the minimal layer thickness for slicing is T_{min}, thus the number of layers N with the minimal thickness is $T/T_{min.}$ Assume each layer has n data points, the computation time can be roughly estimated. The approach in (Wu et al. 2003) need $O(n^4m)$, while the algorithm in this method can be of $O(mn^2\lg n)$. Moreover, in this approach, the boundary data points sorting and the profile curve construction in different layers can be implemented in parallel, which will further quicken the computation.

3.4 Summary

A practical method for achieving Direct RP model construction from 3D cloud data, with accuracy control, has been described. The method commences with an adaptive slicing cloud data along a direction with the control of shape tolerance. The information of boundary points in each layer helps to carry out the quick sorting algorithm and then, the multiresolution method is applied to the sorted data points and projected points to construct a polygon that fit data set within the shape tolerance. The

neighbourhood size and bounds of linearity, which are necessary and difficult in the method presented in chapter 2, are avoided in this method. Testing results on both simulated and real cases will be given in Chapter 4 to show that the algorithm is effective.

CHAPTER 4 CASE STUDIES

The algorithms for Direct RP model construction based on the two methods, ANSAS and WAS, have been implemented with C/C++ in the OpenGL environment. Both simulation results and real case studies are presented here to illustrate the efficacy of the algorithm. The simulation case studies are based on simulated data sets in which the original cloud data are generated by mathematical equations, so that the theoretical shape errors can be obtained accurately and comparison can be made directly. The real case is based on measured data points that are obtained by a laser scanner, and the results after processing are input to a RP machine for fabrication.

4.1. Application Examples of ANSAS

Three case studies are presented here to illustrate the efficacy of the algorithm based on ANSAS. The first two are simulated cases, and the third one is a case study of an actual object, for its cloud data points are obtained from the laser scanner, VIVID 900 system.

4.1.1 Case study 1

In the first case study, a sphere is selected by taking the advantage of its known geometry so that the shape error of the actual slicing can be compared with the

theoretical one accurately. The equation of a sphere with a radius of 2 is given as (note random error is incorporated in the equations to simulate noise in the point cloud):

$$\begin{cases} x = 2cos(\beta)cos(\alpha)+\tau & \tau \text{ is randomly distributed in [-0.01, +0.01]} \\ y = 2cos(\beta)sin(\alpha)+\tau & \beta = [-\pi/2, +\pi/2] \\ z = 2sin(\beta) & \alpha = [0, 2\pi] \end{cases}$$

We use a sampling increment of 0.01 and 0.02 to sample β and α respectively to obtain the cloud data of the sphere. There are totally 98,721 points generated, and the original cloud data is shown in Fig. 4.1a.

For data processing, the initial layer thickness is set at 0.04, and the initial neighbourhood radius is 0.1. ρ_{low} and ρ_{high} are set as 0.85 and 0.9 respectively. Employing a shape tolerance of 0.08, the direct RP model of the sphere shown in Fig. 4.1b is obtained. This model contains 11,812 vertices (points in constructed polygons) distributed in 74 layers. Fig. 4.2a shows the maximum shape error of each layer in the generated model. It can be seen that the maximum shape errors of all the layers are very close to 0.08. As shown in Fig. 4.2b, the sphere with a radius of 2 is then sliced into 74 layers according to the layer thickness in the generated model and the theoretical shape error, also known as stair stepping error as Fig. 4.1c shows. It can be seen that the theoretical errors are close to 0.08 too (except the two tip areas).

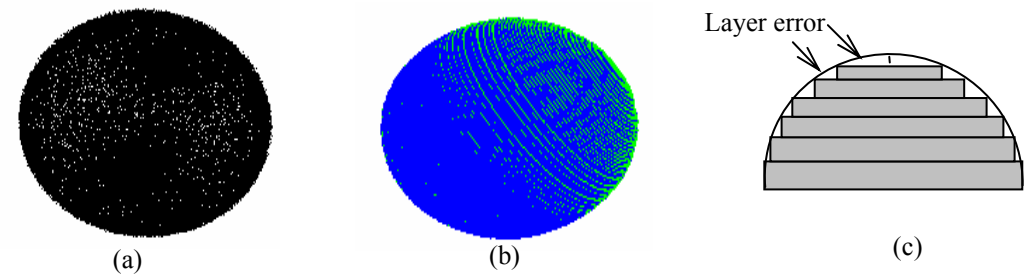
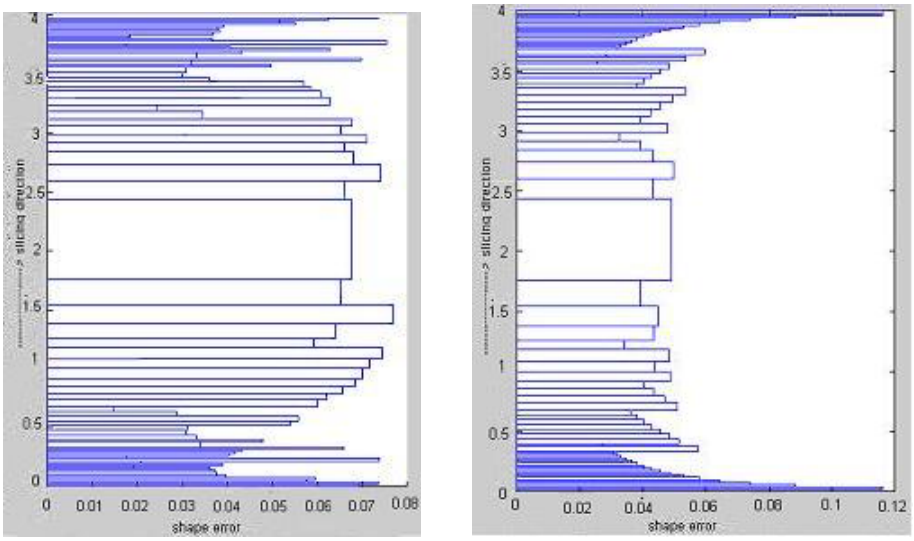


Fig. 4.1: The original cloud data and the direct RP model in case study one

In the two tips, the projected data points are diffuse, and it is difficult to construct a contour curve from these projected points. In our algorithm, we will reduce the layer thickness to as small as possible, up to the required minimal layer thickness of RP machine. When we use this required minimal thickness to slice the cloud data, the actual shape error may be larger than shape tolerance, hence tip problem occurs.



(a) (b)

Fig. 4.2: Shape error comparison in case study one ($\varepsilon = 0.08$)

4.1.2 Case study 2

The second case study uses an object composed of 4 spherical patches (see Fig. 4.3a). The parameter of larger sphere is the same as that in the first case study, and the equations of 3 smaller half-spheres are based on the following basic form:

$$\begin{cases} x = cos(\beta)cos(\alpha)+\tau & \tau \text{ is randomly distributed in [-0.001, +0.001]} \\ y = cos(\beta)sin(\alpha)+\tau & \beta = [0, +\pi/2] \\ z = sin(\beta) & \alpha = [0, 2\pi] \end{cases}$$

The three half-spheres in the object are then formed by the transformation of the basic form as follows:

(1) Translate the basic form along z-axis by 1.732 to obtain the first half-sphere.

(2) Rotate the basic form around y-axis clockwise by 60˚ and translate it along z-axis by 1.732 to form the second half-sphere.

(3) Rotate the basic form around y-axis counter-clockwise by 120˚ and translate it along z-axis by 1.732 to form the third half-sphere.

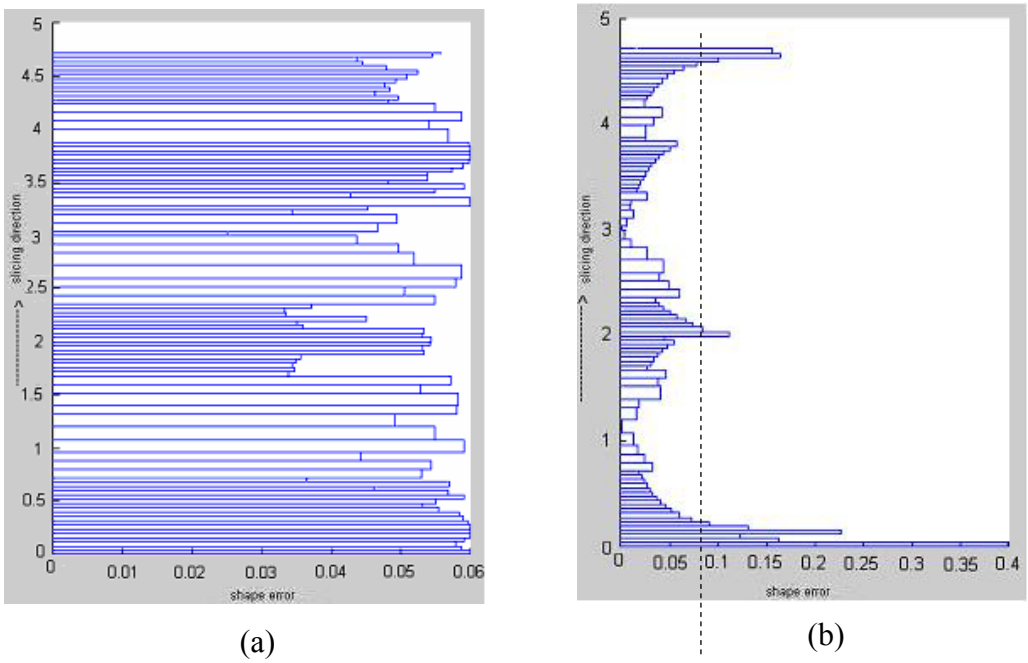
We use sampling increments of 0.02 and 0.05 to sample β and α respectively to obtain the cloud data of the spheres. There are altogether 46,057 points. The original object is shown in Fig. 4.3a.

(a) (b)

Fig. 4.3: The original cloud data and the direct RP model of second case study

The initial layer thickness is set at 0.04, and the initial neighbourhood radius is 0.1. ρ_{low} and ρ_{high} are set as 0.85 and 0.9 respectively. Employing a shape tolerance of 0.06, the direct RP model of the sphere shown in Fig. 4.3b is obtained. This model contains 21,306 vertices distributed in 88 layers. Fig. 4.4a shows the maximum shape error in each layer and it can be seen that the shape errors of all the layers are very close to 0.06. Fig. 4.4b shows the theoretical maximum shape error of the object in each layer sliced using the same pattern as in the generated RP model. Most of the shape errors in each layer (theoretical) are close to 0.06, although it is not as uniform as in case study one. This may be due to the complexity of the object. On the other hand, in both case

studies one and two, the theoretical shape errors in the two tip areas are much larger than the given shape error. This is caused by the zero-radius at the tips. Similarly, there exist tip problems in three small half- spheres, which can be seen from Fig. 4.4b.



(a) (b)

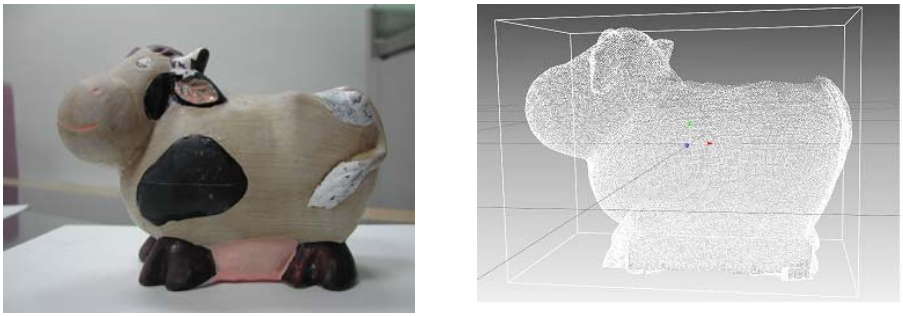
Fig. 4.4: Shape error comparison in case study two ($\varepsilon = 0.06$)

4.1.3 Case study 3

The third case study is that of a toy-cow as shown in Fig. 4.5. The original object can be boxed in a volume of 150mm×120mm×90mm and was digitised by a laser scanner, Minolta VIVID-900 digitizer. The data sets were obtained from different view angles, and the noisy data points and background data points were filtered and the holes were filled to produce a cloud data set of 1,098,753 points. The adaptive slicing algorithm was applied to the cloud data employing an error tolerance of 0.7 mm, initial layer thickness of 0.2 mm, and the initial neighbourhood radius of 0.2 mm. ρ_{low} and ρ_{high} were set as 0.85 and 0.9 respectively. This resulted in a direct RP model as shown Fig. 4.6a with 115 layers and 59,686 points. The shape error of each layer in the generated

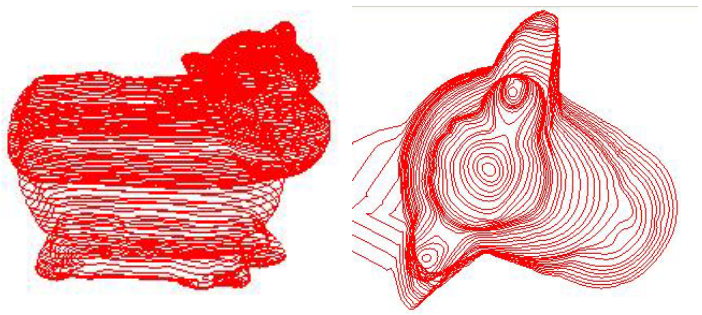
model is shown in Fig. 4.7a and it clearly shows that the shape errors are within 0.7 mm. In the model construction process, the head area (ears and horns) has a very complex shape, and it posed a multiple-loop problem. This can be seen clearly in Fig. 4.6b, in which the multiple-loops are separated successfully and the corresponding layers are generated.

It took about 30 minutes for the adaptive slicing algorithm to generate the RP model using a PC of 1.5GHZ CPU. The direct RP model was then converted to a layer-based RP slice-data file in CLI format and fed to a RP machine, high-temperature Laser Manufacturing System (HTLMS). For this RP machine, a uniform thickness of layers is required, and hence the direct RP model of 115 layers was further sliced into 535 layers, with the layer thickness of 0.2 mm (the thinnest layer in the model). It took about six hours to complete the fabrication. Fig. 4.7b shows the real work-piece fabricated by the RP machine based on the direct RP model. However, in a RP machine that can deposit material with different layer thickness, time-saving could be up to 78.5%.



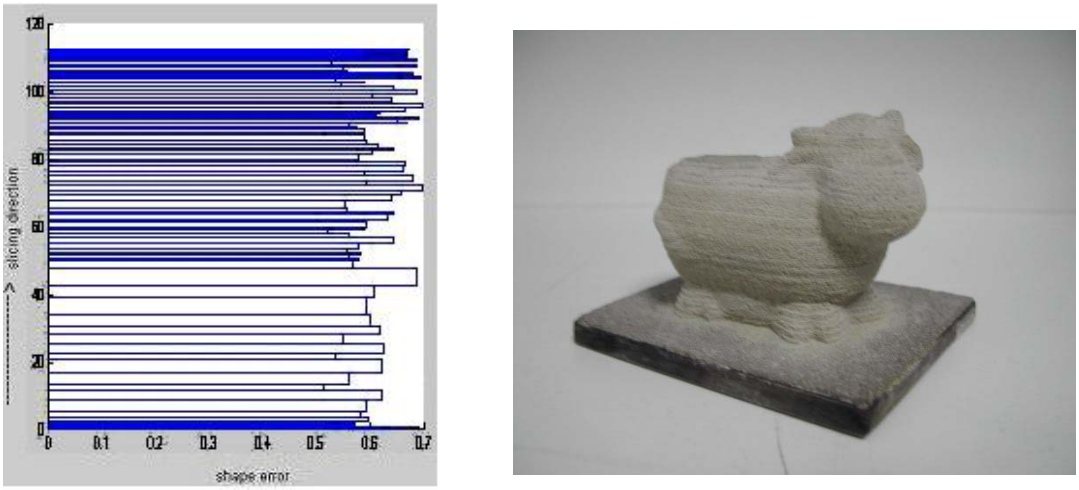
(a) (b)

Fig. 4.5: The original object and cloud data,



(a) (b)

Fig. 4.6: The direct RP model and a zoom-in view at the head



(a) (b)

(a) Shape error of the direct RP model in case study three
(b) The toy-cow fabricated by RP machine

Fig. 4.7: Shape error of the direct RP model

4.2. Application Examples of WAS

Three case studies are presented here to illustrate the efficacy of the algorithm presented in Chapter 3. To compare the performance of this algorithm with that in Chapter 2, we use the same data points as in case 4.1.1 and case 4.1.2. Moreover, to illustrate the advantage of dealing with sharp corners over the former algorithm, we also present a trihedron as a case study.

4.2.1 Case study 1

The data points of the first case study are the same as the case in section 4.1.1. For data processing, the initial layer thickness is set at 0.04. Employing a shape tolerance of 0.08, the direct RP model of the sphere shown in Fig. 4.8 is obtained. In this case study, the profile curve is decomposed into around five levels. This model contains 9,923 vertices distributed in 67 layers. Fig. 4.9a shows the maximum shape error of each layer in the generated model. It can be seen that the maximum shape errors of all the layers are very close to 0.08. The sphere with a radius of 2 is then sliced into 67 layers according to the layer thickness in the generated model and the theoretical shape error, is shown in Fig. 4.9b. It can be seen that the theoretical errors are close to 0.08 too (except the two tip areas).

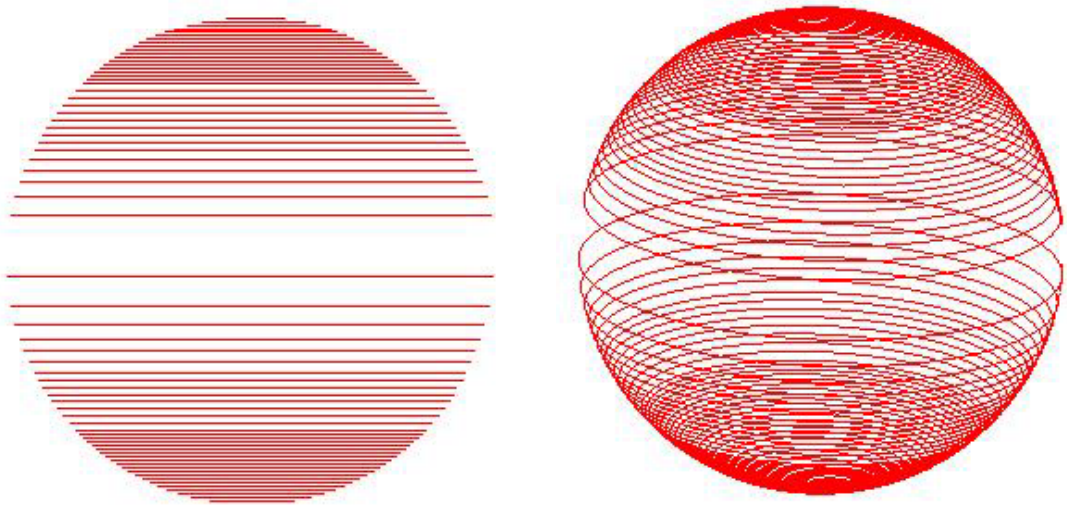
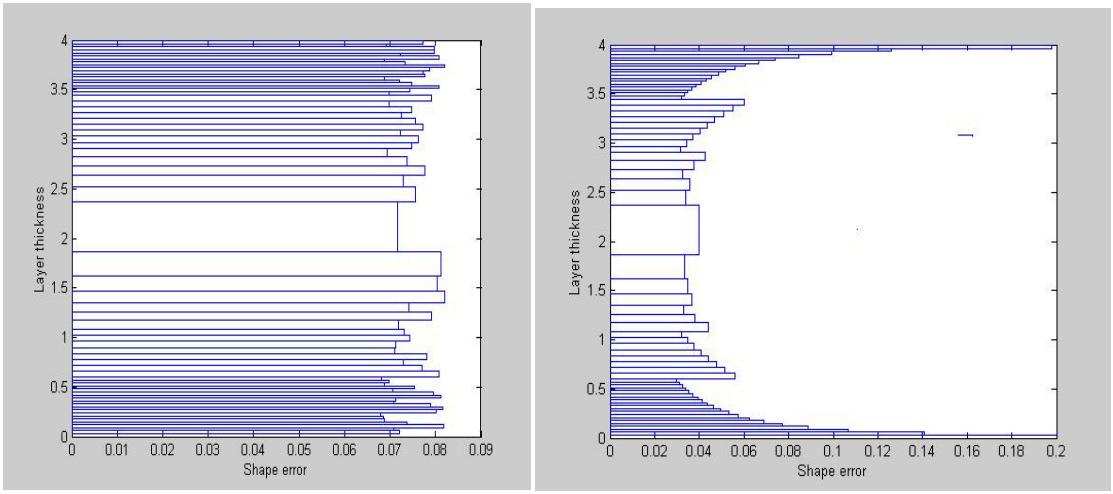


Fig. 4.8: The Direct RP model of shpere



(a) (b)

Fig. 4.9: Shape error comparison in case study one ($\varepsilon = 0.08$)

The number of layers is only 67. The WAS and ANSAS algorithms are running on a PC with a 1.8GHZ CPU, and the memory is 256MB. WAS took 4 minutes to complete the slicing this case, however ANSAS took 11 minutes. It shows that WAS has the advantage over ANSAS in this respect. Moreover, the number of layer thickness and vertices after processing are not so different in these two cases, and hence, WAS did not have an obvious advantage to reduce the data points.

4.2.2 Case study 2

In the second case study, the data points are the same as the case in section 4.1.2, and the original cloud data is shown in Fig. 4.3 a. The minimal layer thickness is set at 0.04, and the shape tolerance is 0.06, then the direct RP model of the sphere shown in Fig. 4.10 is obtained. There are totally 18,419 points that are distributed in 76 layers. We can see that the number of layers is less than that of the result dealt by ANSAS with the same shape tolerance.

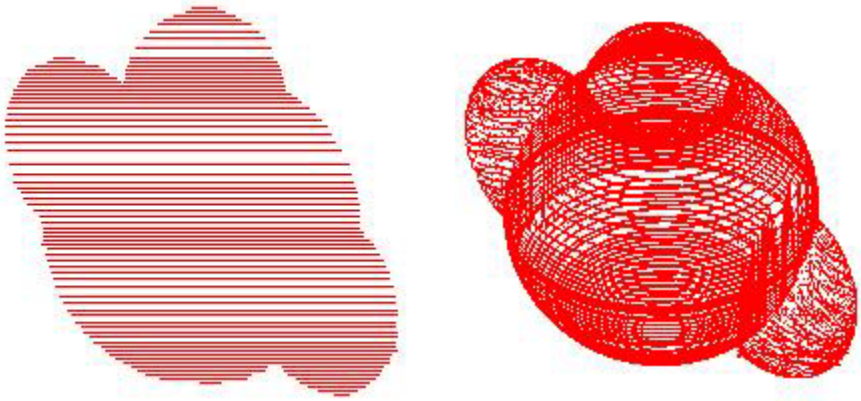
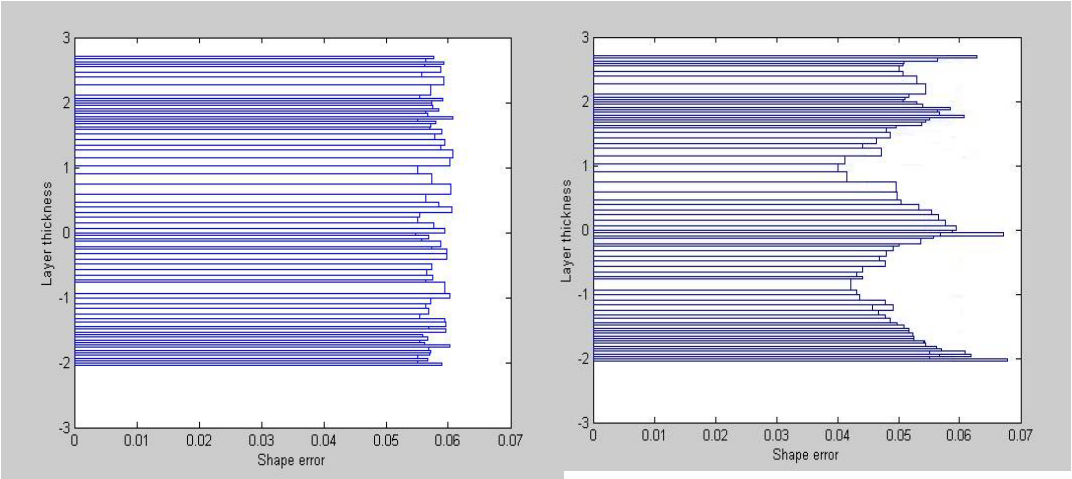


Fig. 4.10: The Direct RP model of spheres

Fig. 4.11a shows the maximum shape error in each layer and it can be seen that the shape errors of all the layers are very close to 0.06. Fig. 4.11b shows the theoretical maximum shape error of the object in each layer sliced using the same pattern as in the generated RP model. Most of the shape errors in each layer (theoretical) are close to 0.06, although it is not as uniform as in case study 1 in section 4.2.1. This may be due to the complexity of the object. On the other hand, in both case studies 1 and 2, the theoretical shape errors in the two tip areas are much larger than the given shape error. This is caused by the zero-radius at the tips. The profile curve is decomposed into five levels during the construction with the wavelets. WAS took 7 minutes to complete this case study, while ANSAS took 15 minutes. Obviously, WAS method is faster than ANSAS method.



(a) (b)

Fig. 4.11: Shape error comparison in case study one ($\varepsilon = 0.06$)

4.2.3 Case study 3

In the third case study, the object used has corners. As we stated in previous chapters, the WAS has the advantage of dealing with sharp corners, we present a simulation model here.

This case is a sphere patch together with a trihedron. The equation of a sphere with a radius of 2 is given as (note random error is incorporated in the equations to simulate noise in the point cloud):

$$\begin{cases} x = 2cos(\beta)cos(\alpha)+\tau & \tau \text{ is randomly distributed in [-0.01, +0.01]} \\ y = 2cos(\beta)sin(\alpha)+\tau & \beta = [-\pi/2, +\pi/6] \\ z = 2sin(\beta) & \alpha = [0, 2\pi] \end{cases}$$

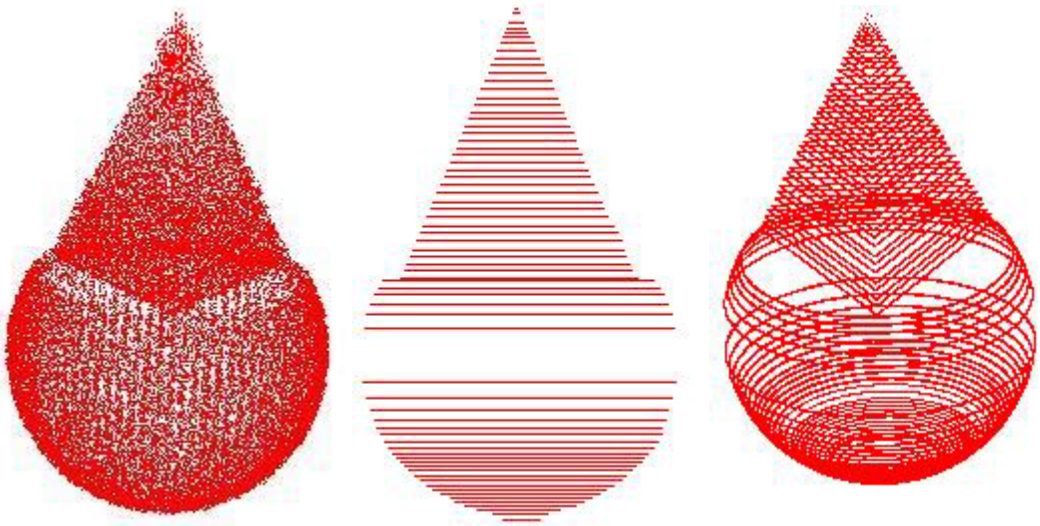
The four vertices of the trihedron are:

$$\begin{cases} \mathbf{A}=[0, -1.732, 1]; \\ \mathbf{B}=[-1.5, 0.866, 1]; \\ \mathbf{C}=[1.5, 0.866, 1]; \\ \mathbf{D}=[0, 0, 4.464]; \end{cases}$$

Thus, the trihedron is on the top of the sphere patch, and **A**, **B**, and **C** lie on the sphere patch. The centre **E** of triangle **ABC** is: [0, 0, 1].

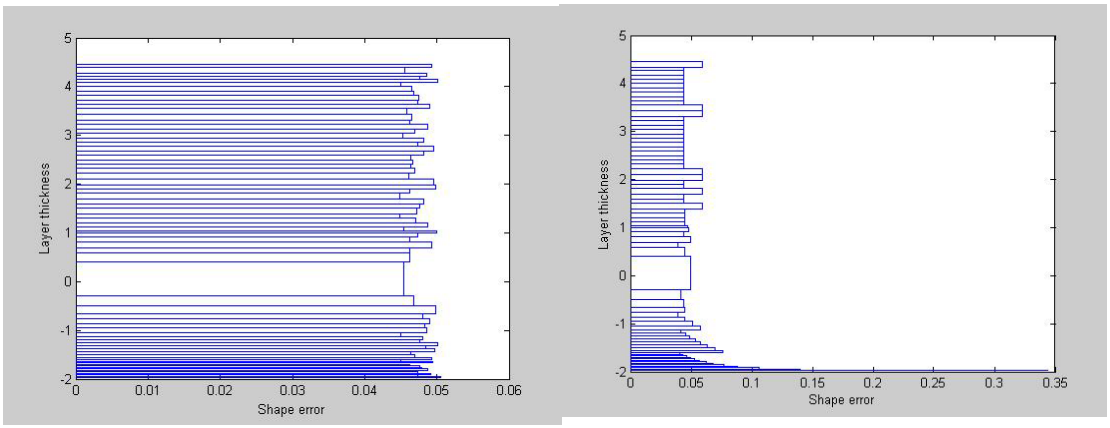
We use a sampling increment of 0.01 and 0.02 to sample β and α respectively to obtain the cloud data of the sphere patch. To sample the trihedron, we first use parallel planes along **ED** to slice trihedron to obtain the intersection points **O**, **P**, and **Q** on **DA**, **DB** and **DC**. Then, we use the linear interpolation formula to sample the line **OP**, **PQ** and **OQ**. The linear sampling formula is as follows: Given start point **S** and end points **E**, the points in the line segment **SE** are sampled as: α**S**+(1-α)**E**, and α is a variable among [0,1].

The distances between slicing planes are 0.01, and the linear parameter α is sampled as 0.01, and then the trihedron is sampled. In our simulation, a random error distributed in [-0.01, 0.01], is added into the sampled data points. There are totally 468, 512 points generated, and the original cloud data is shown in Fig. 4.12a.



(a) (b) (c)

Fig. 4.12: The original cloud data and the direct RP model of third case study



(a) (b)

Fig. 4.13: Shape error comparison in case study one ($\varepsilon = 0.05$)

For data processing, the initial layer thickness is set at 0.04, and the shape tolerance is set at 0.05, the direct RP model of the sphere shown in Fig. 4.12b and Fig. 4.12c are obtained. This model contains 8,875 vertices distributed in 71 layers. Fig. 4.13a shows the maximum shape error of each layer in the generated model. It can be seen that the maximum shape errors of all the layers are very close to 0.05. The CAD model of this case is then sliced into 71 layers according to the layer thickness in the generated model and the theoretical shape error is shown in Fig. 4.13b. It can be seen that the theoretical errors are close to 0.05 too (except the two end areas). When employing ANSAS method to slice this case study with the shape tolerance at 0.05, neighbourhood size at 0.1, and linearity bounds at 0.85 and 0.9 respectively, it terminated and gave a result as shown in Fig. 4.14. If parameters of the bounds and neighbourhood size are selected suitably, ANSAS can slice this object. We did not show the final results here, because this case study is used to show the efficiency of WAS method to slice the object with sharp corners.

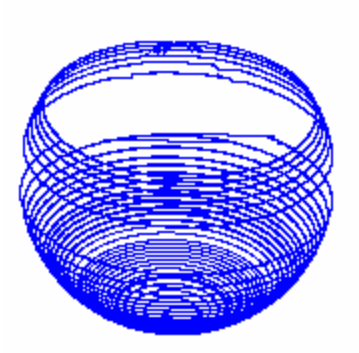
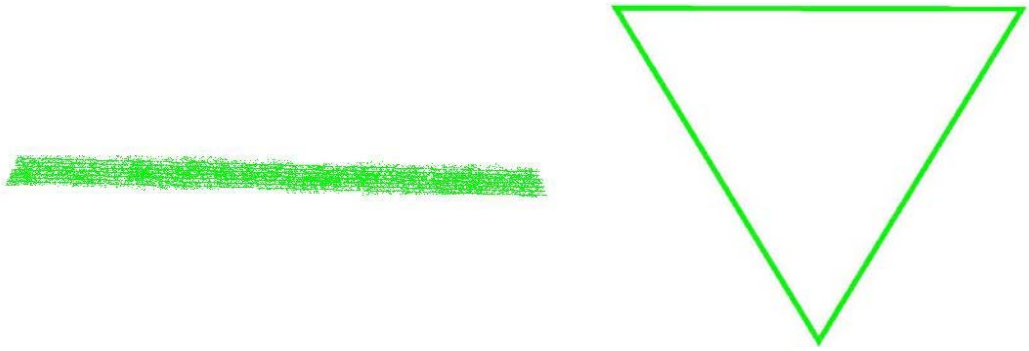


Fig. 4.14: Direct RP model of case study 3 based on ANSAS

To see the sharp corner construction, we select one layer of the sliced cloud data to show the process. As Fig. 4.15a shows, the thickness of cloud data is 0.087, and it is the 38th layer of the cloud data from Fig. 4.12. Fig. 4.15b shows the projected data points of this layer. There are totally 2,510 points.

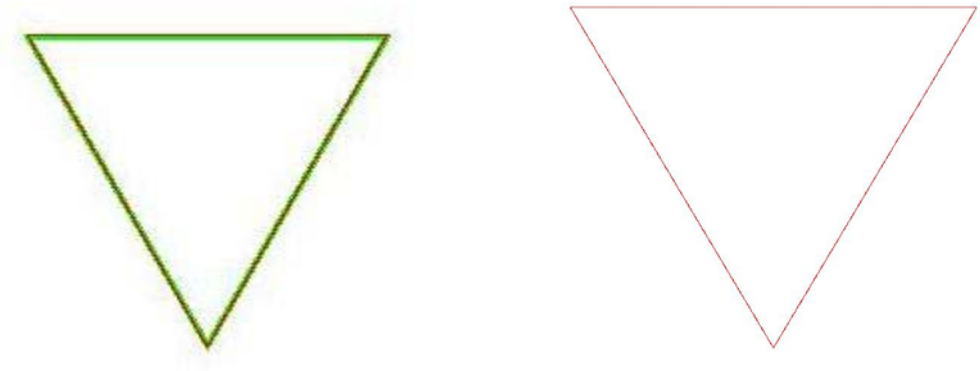


(a) (b)

Fig. 4.15: Cloud data and its planar data set in one layer

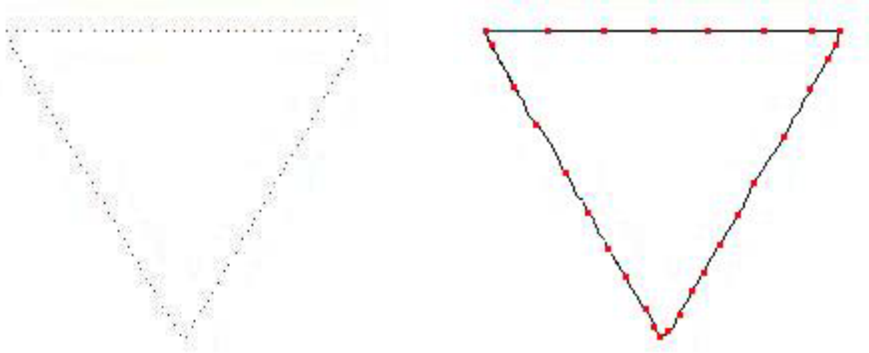
Using the segmentation method in Chapter 3, the boundary points in this layer are obtained as shown in Fig.4.16, which shows the boundary points together with the projected points, and we can see that the boundary points are nearly in the middle of the projected point band. Fig. 4.16b shows the boundary points, and we can see that it is dense but it keeps the topology of the whole projected points. There are totally 528 points.

To carry on curve construction, we first use the fixed neighborhood size to sort the boundary points, as shown in Fig. 4.17a, and there are 128 points left. We set these sequenced data points as the initial scaling coefficients of wavelets, and then decompose it level by level. Then, using WAS, we can get the final curve shown in Fig. 4.17, and there are 29 points left. The shape error, maximal distance from planar data points in Fig. 4.15, to this curve, is 0.048, which is close to the shape tolerance 0.05.



(a) (b)

Fig. 4.16: Boundary points in one layer



(a) (b)

Fig. 4.17: Curve decomposition and reconstruction based on wavelets

To compare the WAS with ANSAS, we use the ANSAS to reconstruct the curve from planar data points in Fig. 4.15. Employing the shape tolerance 0.05, the

initial neighborhood size 0.05, and the correlation coefficient bound at 0.85 and 0.9, we get the curve as shown in Fig. 4.18. There are 28 data points left, and the shape error is 0.052. We can see that at the corners, the curve shows zigzags, and more seriously, there exists self-intersection. Hence, WAS is better to deal with small sharp corners than ANSAS. To solve this problem, we need to use a curve smoothing method, which is a time-consuming process.

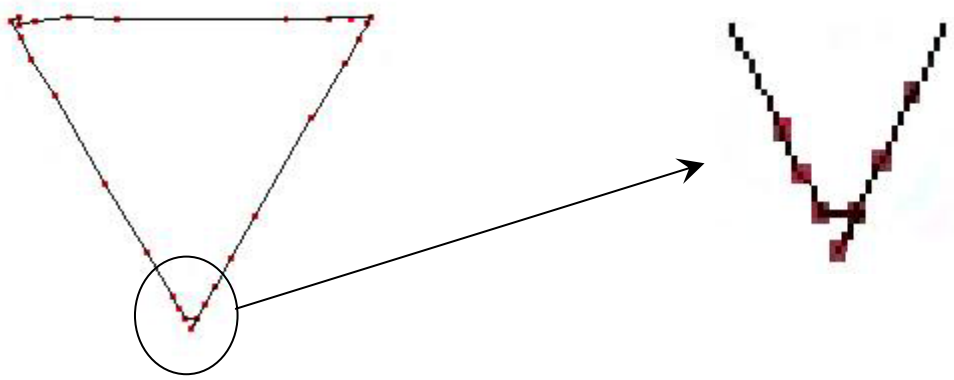


Fig. 4.18: Curve construction based on adaptive neighborhood size.

4.2.4 Case Study 4

The part used in the fourth case study is a lower jaw model as shown in Fig. 4.19. The original object can be boxed in a volume of 78mm×72mm×52mm and was digitised by the laser scanner, Minolta VIVID-900 digitizer. The data sets were obtained from different view angles, and the noisy data points and background data points were filtered and the holes were filled to produce a cloud data set of 276,591 points. The adaptive slicing algorithm was applied to the cloud data employing an error tolerance of 0.8 mm, and initial layer thickness of 0.2 mm. This resulted in a direct RP model as shown Fig. 4.20a with 68 layers and 9,438 points. The shape error of each layer in the generated model is shown in Fig. 4.20b and it clearly shows that the shape errors are within 0.8 mm.

We employed a shape tolerance of 0.5 mm and initial layer thickness of 0.2 mm. This resulted in a direct RP model as shown Fig. 4.21a with 84 layers and 15, 347 points. The shape error of each layer in the generated model is shown in Fig. 4.21b and it clearly shows that the shape errors are within 0.5 mm. However, the shape tolerance cannot be arbitrarily small, because the cloud data are not sampled dense enough, and there are errors during scanning process. Moreover, the RP machine cannot fabricate the model with arbitrary small thickness, and its required layer thickness will cause the shape error to be larger than shape tolerance. In this case, we find 0.5mm is nearly the minimal shape tolerance we could use.

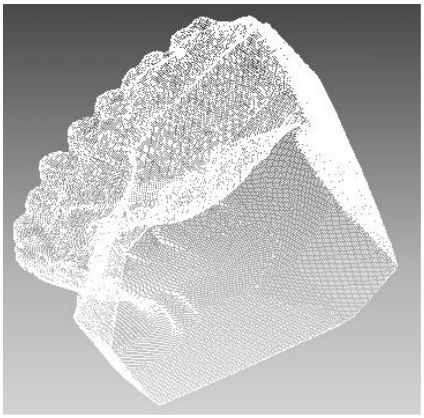
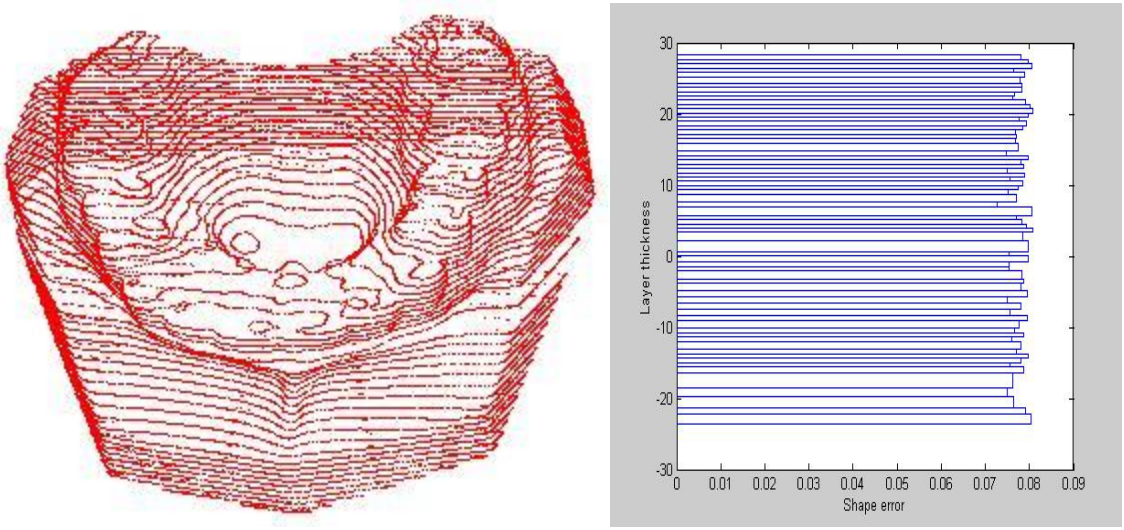
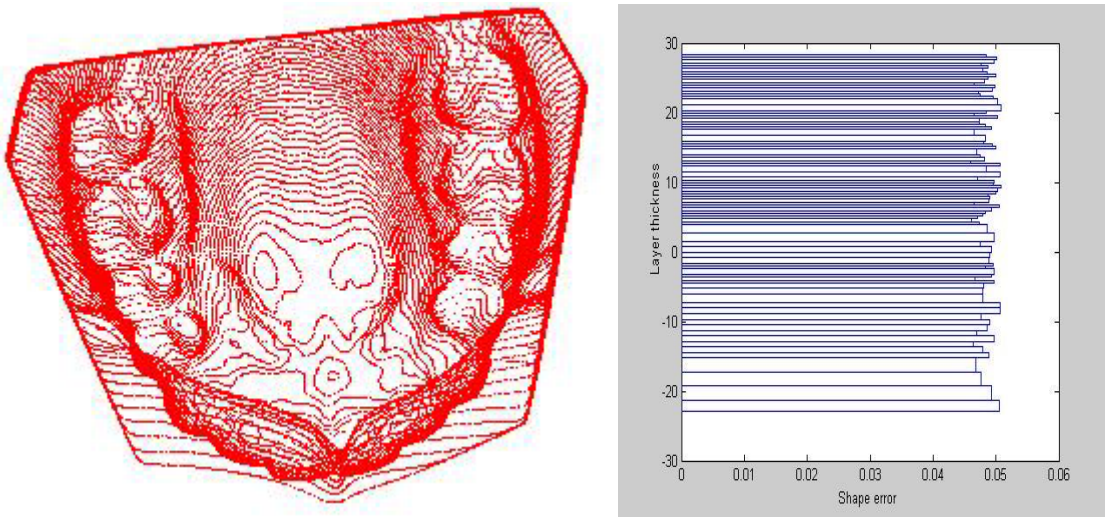


Fig. 4.19: Cloud data of lower jaw



(a) (b)

Fig. 4.20: Direct RP model (WAS) and shape error (ε=0.8mm)



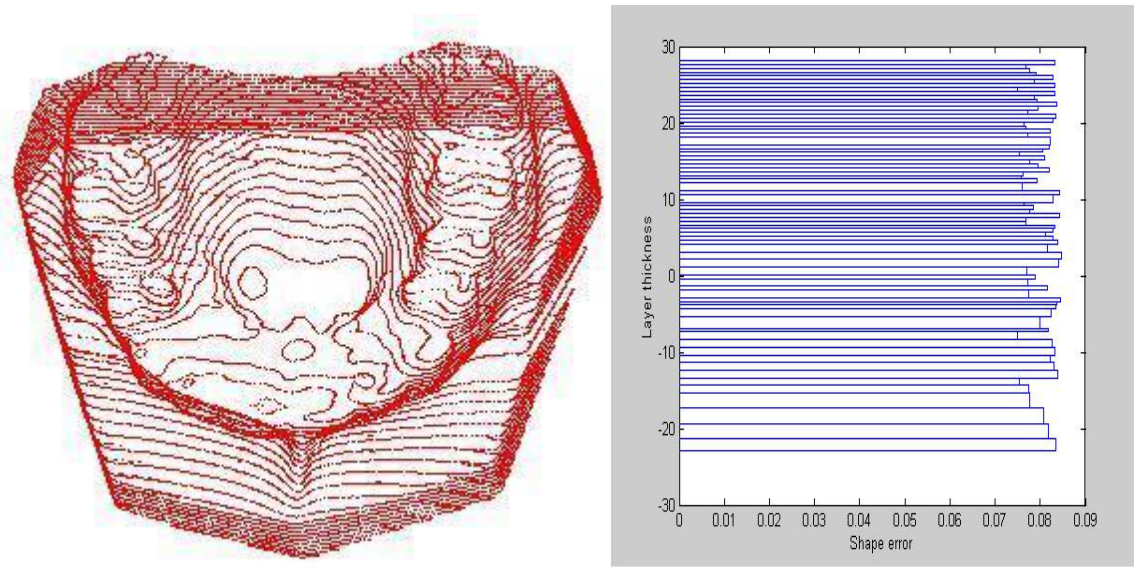
(a) (b)

Fig. 4.21: Direct RP model (WAS) and shape error (ε=0.5mm)

To compare the result of this case generated by WAS with that generated by ANSAS, we use ANSAS to construct the RP model from the same cloud data. We employ the shape error at 0.8mm, the initial neighbourhood size at 0.4mm, the initial layer thickness at 0.2mm, and the correlation coefficient bounds at 0.85 and 0.09. The Direct RP model is obtained shown as Fig.4.22a, and there are 72 layers together with

13,214 data points. The shape error of each layer in the generated model is shown in Fig. 4.22b and it clearly shows that the shape errors are within 0.8 mm. We can see that using WAS result in fewer number of layers than ANSAS, when both of them use the same shape tolerance, in this case, 0.8mm. As for speed comparison, WAS took 18 minutes, while ANSAS took 34 minutes.

However, ANSAS is not so robust in this case study, because it needs a tradeoff among the parameters of neighbourhood size, number of neighbourhood data points of the certain point, and correlation coefficients bounds, as we mentioned in Chapter 2. When we select the shape tolerance 0.5mm, ANSAS does not work, even though we tried many different parameters. Hence, we can see that WAS is more robust than ANSAS not only in principle but also in practise.



(a) (b)

Fig. 4.22: Direct RP model (ANSAS) and shape error (ε=0.8mm)

CHAPTER 5 CONCLUSIONS AND FUTURE WORKS

In this thesis, two methods for generating RP models directly from arbitrarily scattered cloud data are presented. In these two methods, the modelling process consists of several steps: (1) the cloud data are segmented into several layers along the RP build direction; (2) points within each layer are treated as co-planar and a polygon is constructed to best-fit the points; (3) the thickness of each layer is determined adaptively such that the surface error is kept just within a given error bound.

Basically, the two methods differ in step 2 by using a different curve construction method. The first method, ANSAS, uses the correlation coefficient to control the neighbourhood size in its adaptive polygon construction. This algorithm is efficient in dealing with smooth surfaces without sharp corners.

The second method uses wavelets to decompose the curve level by level. The curve is constructed by matching the desired resolution to the required error. This method is compact relative to the segmentation results of cloud data. But it is very fast, and good at dealing with small and sharp features, such as corners and creases.

Algorithms based on the two methods have been implemented with C/C++ in OpenGL platform. The results of both simulated and practical cases show that the algorithms are effective.

The main contribution of this thesis is two-fold. Firstly, the polygon construction algorithm is adaptive in nature. It is capable of automatically finding a feasible starting point and identifying the maximum allowable neighbourhood for each segment. It is also able to deal with segments with multiple-loop profile effectively. Secondly, the thickness of each layer is determined adaptively, based on a given surface tolerance.

This provides an intuitive control parameter to users and the resulted model needs a close-to-minimum RP building time.

Further challenging issues are as follows: The first one is on the adaptive determination of the lower and upper linearity bounds for the polygonal curve construction in ANSAS method. We have observed that this is related to the given shape tolerance and the random errors in the original cloud data. Future study into controlling these two bounds can be pursued by considering the shape tolerance and the accuracy level of the scanner. Another challenge is to determine the bandwidth tolerance more accurately, such that the multiresolution curve construction algorithm in the wavelet-based method is always convergent. A simple method has been employed to restrict the bandwidth in this thesis. However, this simplification may reduce the final layer thickness. Hence, a future study is to control the bandwidth based on shape tolerance.

REFERENCES

Amidror, I., Scattered data interpolation methods for electronic imaging systems: a survey. Journal of Electronic Imaging, 2002, 11(2), pp.157-176

Boissonnat, J. D., Geometric structures for three-dimensional shape representation. ACM Trans Graph ,1984, 3(4), pp. 266-286

Boissonnat, J. D. and Cazals, F., Smooth surface reconstruction via natural neighbour interpolation of distance functions. Computational Geometry, 2002, 22(1-3), pp. 185-203

Chen, L.C., and Lin G.C.I., An integrated reverse engineering approach to reconstructing free-form surfaces. Computer Integrated Manufacturing System, 1997, 10(1),pp. 49-60

Chen, X. and Schmitt, F., Surface modelling of range data by constrained triangulation. CAD, 1994, 26(8), pp. 632-645

Chen, Y. H., Ng, C. T. and Wang, Y. Z., Generation of an STL file from 3D measurement data with user-controlled data reduction. International Journal of Advanced Manufacture Technology, 1999, 15(2), pp. 127-131.

Chivate, P. N. and Jablokow, A. G., Solid-model generation from measured point data. CAD, 1993, 25(9), pp. 587–600

Chua, C. K. and Leong, K. F., Rapid prototyping: Principles and Applications in Manufacturing. John Wiley & Sons. Inc., 1996, pp. 22-83

Chuang, G. and Kuo, C., Wavelet descriptor of planar curves: Theory and applications. IEEE Trans. on Image Processing, 1996, 1(5), pp.56-70

Chui, C.H., Wavelet Analysis and Its Applications. Academic Press, 1992

Chung, K., The generalized uniqueness wavelet descriptor for planar closed curves. IEEE Trans. on Image Processing, 2000, 9(5), pp.834-845

Daubechies, I., Ten Lectures on Wavelets, SIAM, 1992

Eck, M. and Hoppe, H., Automatic reconstruction of B-spline surfaces of arbitrary topological type. ACM SIGGRAPH, 1996, pp. 325-334

Edelsbrunner, H. and Mucke, E. P., Three-dimensional alpha shapes. ACM Trans Graph, 1994, 13(1), pp. 43-72

Sabourin, E., Houser, S. A. and Born, J. H., Accurate exterior, fast interior layered manufacturing, Rapid Prototyping Journal, Vol. 3, No.2, 1997: 44-52

Esteve, J., Brunet, P. and Vinacua, A., Multiresolution for Algebraic Curves and Surfaces Using Wavelets. Computer Graphics Forum, 2001, 20(1), pp. 47-59

Hamann, B., Curvature approximation for triangulated surfaces. Computing Supplementum, 1993, 8, pp. 139-153

Hoffman, R. and Jain, K., Segmentation and classification of range images. *IEEE* Pattern Analysis and Machine Intelligence 1987, 9(5), pp. 608–620.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W., Surface reconstruction from unorganized points. ACM SIGGRAPH, 1992, pp. 71–78.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W., Mesh optimization. ACM SIGGRAPH, 1993, pp. 19-26

Hosni, K. C., A non-contact automated measurement for free-from surface profile. Computer Integrated Manufacturing System, 1997, 10(4), pp. 277-285

Jacobs, P. F., Rapid prototyping and manufacturing. Dearborn: Society of manufacturing Engineers, 1992.

Jun, Y., Raja, H. Y. and Park, S., Geometric feature recognition for reverse engineering using neural networks. International Journal of Advanced Manufacturing Technology, 2001, 17, pp. 462-470

Kamash Tata.et al., Efficient slicing for layered manufacturing. Rapid Prototyping Journal, 1998, 4(4), pp. 151-167

Kobbelt, L., Discrete fairing and variational subdivision for free-form surface design. Visual computer, 2000, 16(3-4), pp. 142-158

Kulkarni, P. and Dutta, D., An accurate slicing procedure for layered manufacturing. CAD, 1996, 28(9), pp. 683-697

Lee, In-Kwon, Curve reconstruction from unorganized points. Computer Aided Geometric Design, 2000, 17(2), pp. 161-177.

Lee, K. H. and Woo, H., Direct integration of reverse engineering and rapid prototyping. Computers & Industrial Engineering, 2000, 38 (1), pp.21-38.

Lee, S. H., Kim., H. C., Hur, S. M. and Yang, D. Y., STL file generation from measured point data by segmentation and Delaunay triangulation. CAD, 2002, 34(10), pp.691-704.

Li, L., Schemenauer, N., Peng, X., Zeng, Y. and Gu, P., A reverse engineering system for rapid manufacturing of complex objects. Robotics and Computer Integrated Manufacturing, 2002, 18 (1), pp. 53-67.

Liu, G. H., Segmentation of cloud data for reverse engineering and direct rapid prototyping. Master of Engineering Thesis, National University of Singapore, 2001.

Liu, G. H., Wong, Y. S., Zhang, Y. F. and Loh, H. T., Error-based segmentation of cloud data for direct rapid prototyping. CAD, 35(7), pp.633-645

Mani, K. et al, Region-based adaptive slicing. CAD 1999,31, pp. 317-333

Menq, C. and Chen, F. L., Curve and surface approximation from CMM measurement data. Computers & Industrial Engineering, 1996, 30(2), pp. 211-225

Milroy, J., Bradley, C. and Vickers, G. W., Segmentation of a wrap-around model using an active contour. CAD, 1997, 29, pp. 299-320

Peng, Q. J. and Loftus, M., A new approach to reverse engineering based on vision information. International Journal of Machine Tools and Manufacture, 1998, 38(8), pp. 881-899

Piegl, L. and Tiller, W., *The NURBS Book*, Springer, 1997

Pigounakis, K.G. and Kaklis, P.D., Convesity-preserving fairing. Computer Aided Design, 1996, 28(12),pp.981-984

Pitman, J., Probability, Springer, Berlin, 1992.

Reqicha, A., A representation for rigid bodies: Theories, methods, and systems. Computer Survey, 1990, 12(4), pp. 437-463

Sapidis, N. and Farin, G., Automatic fairing algorithm for B-spline Curves. Computer Aided Design, 1990, 22(2), pp.121-129

Stollnitz, E.J., Derose, T.D. and Salesin, D.H., Wavelets for Compute Graphics: Theory and applications. Morgan Kaufmann Publishers, 1996

Sun, W., Bradley, C., Zhang, Y. F., and Loh, H. T., Cloud data modelling employing a unified, non-redundant triangular mesh. CAD, 2001, 33(2), pp. 183-193.

Tai, C.C., and Huang, M. C., The processing of data points basing on design intent in reverse engineering. International Journal of Machine Tools & Manufacture, 2000, 40, pp. 1913-1927

Taubin, G. and Ronfard, R., Implicit simplicial models for adaptive curve reconstruction. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1996, 18, pp. 321-325

Ueng, W. D., Lai, J. Y. and Doong, J. L., Sweep-surface reconstruction from three dimensional measured data. CAD, 1998, 30(10), pp. 791-805

Varady, T., Martin, R. R., and Cox, J., Reverse engineering of geometric models-an introduction. CAD, 1997, 29 (4), pp. 255-68.

Wang, Y.P., Lee, S.L. and Kazuo Toraichi, Multiscale Curvature-Based Shape Representation Using B-Spline Wavelets. IEEE Trans. on Image Processing, 1999, 8(11), pp.1586-1592

Weir, D. J., Milroy, M., Bradley, C. and Vickers, G. W., Reverse engineering physical models employing wrap-around B-spline surfaces and quadrics. Proceedings of the Institution of Mechanical Engineers-Part B, 1996, 210, pp. 147–157.

Wu, Y.F., Wong, Y.S, Loh, H.T. and Zhang, Y.F., Modelling Cloud Data Using an Adaptive Slicing Approach. Computer Aided Design, (to appear).

Xu, F., Integrated decision support for part fabrication with rapid prototyping & manufacturing systems. Ph.D thesis, National University of Singapore, 1999.

Yan, X. and Gu, P., A review of rapid prototyping technologies and systems. CAD, 1996, 28(4), pp. 307-318

Yang, M. and Lee, E., Segmentation of measured point data using a parametric quadric surface approximation. CAD, 1999, 31, pp. 449-457

PUBLICATION

[1] Y.F. Zhang, Y.S. Wong, H.T. Loh, and Y.F. Wu, An Adaptive Slicing Approach to Modelling Cloud Data for Rapid Prototyping. To be presented at the 6th Asia Pacific Conference on Materials Processing in September, 2003 in Taipei.

[2] Wu, Y.F., Wong, Y.S., Loh, H.T. and Zhang, Y.F., Modelling Cloud Data Using an Adaptive Slicing Approach. Computer Aided Design, (Article in Press).