ELSEVIER

Computer-Aided Design 36 (2004) 231–240

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

Modelling cloud data using an adaptive slicing approach

Y.F. Wu, Y.S. Wong, H.T. Loh, Y.F. Zhang*

Department of Mechanical and Production Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore, Singapore 119260

Abstract

In reverse engineering, the conventional surface modelling from point cloud data is time-consuming and requires expert modelling skills. One of the innovative modelling methods is to directly slice the point cloud along a direction and generate a layer-based model, which can be used directly for fabrication using rapid prototyping (RP) techniques. However, the main challenge is that the thickness of each layer must be carefully controlled so that each layer will yield the same shape error, which is within the given tolerance bound. In this paper, an adaptive slicing method for modelling point cloud data is presented. It seeks to generate a direct RP model with minimum number of layers based on a given shape error. The method employs an iterative approach to find the maximum allowable thickness for each layer. Issues including multiple loop segmentation in layers, profile curve generation, and data filtering, are discussed. The efficacy of the algorithm is demonstrated by case studies.

Keywords: Reverse engineering; Rapid prototyping; Cloud data; Layer-based model; Shape-error

1. Introduction

Reverse engineering (RE) refers to creating a CAD model from an existing physical object, which can be utilized as a design tool for producing a copy of an object, extracting the design concept of an existing model, or re-engineering an existing part [1]. 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. This normally produces a large cloud data set that is usually arbitrarily scattered. Rapid prototyping (RP) is a material-growing 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 [2,3]. Therefore, modelling point cloud for RP fabrication is essential to achieve rapid product development.

In general, modelling point cloud for RP can be realised in three different approaches [4]. In the first approach, a surface model is reconstructed from point cloud and then converted to a STL file format. The STL file is sliced to generate a series of layers for RP fabrication. The second approach creates a STL-format file of a model directly from point cloud (e.g. triangulation) [5–7]. The third approach goes directly from point cloud to a RP slice file (layer-based model) [8]. 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) between the surface model and the STL model, and (3) between the STL model and the layer-based RP model. Cumulatively, these will make difficult to control the shape error of the RP model. The model generated from the second approach is effectively a STL model. In this case, the shape error of the final RP model comes from two sources: (1) the shape error between the cloud data and the STL model, and (2) between the STL model and the layer-based RP model. The control of the final shape error is still 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 basically only one source of shape error. If this error can be controlled effectively, this approach will have the advantage over the other two modelling approaches in terms of shape error control on the RP model.

Liu [8] developed an automated segmentation approach for generating a layer-based model from point cloud. This is

* Corresponding author. Tel.: +65-6874-2868.
E-mail address: mpezyf@nus.edu.sg (Y.F. Zhang).

0010-4485/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/S0010-4485(03)00097-6

accomplished via three steps. First, the cloud data is adaptively sub-divided into a set of regions according to a given sub-division error (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 (FPs) within the user-defined shape tolerance using a digital image based reduction method. Secondly, based on the FPs 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 faired and subsequently closed to generate the final layer-based RP model. He has demonstrated that the developed system is able to generate layer-based model from point cloud. However, the sub-division error, 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. In this paper, we present an intuitive method 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 machine used for fabrication allow arbitrary thickness to be built.

The remaining of this paper describing the follows. In Section 2, the proposed approach is outlined. In Section 3, we describe the algorithm of polygonal curve reconstruction from planar data points. In Section 4, the method for determining the maximum allowable thickness for each layer is described. In Section 5, some experimental results are given to show the efficacy of the developed algorithm. Finally, conclusions are given in Section 6.

2. The proposed 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 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, results 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. 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.

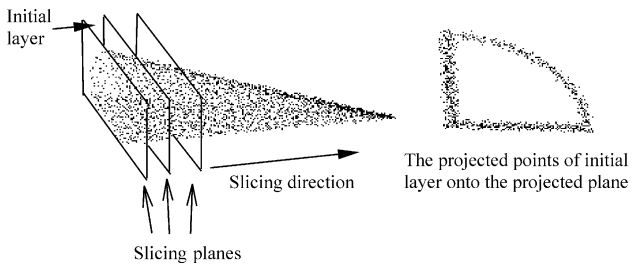


Fig. 1. Point cloud slicing and projecting.

This layer-based model can be used for RP fabrication directly and the final shape of the produced part is exactly the same with the model, thus the shape error can be controlled effectively. The details of the algorithms of this approach are described in the following sections.

3. 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. 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.

Curve reconstruction is to approximate the unorganised points 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 a polygon is formed. To approximate the points set accurately, the polygonal curve must keep the FPs of original shape

defined by points set. Liu [8] 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 its neighbourhood points' centre determine the oriented vector of point, based on which the next point can be retrieved. Repeatedly, the FPs are determined. The data is compressed by removing redundant points other than the FPs. Finally, the curve is obtained by linking all the FPs 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 FPs if the chosen radius is too large; and if the chosen radius is too small, the algorithm lacks of efficiency.

Lee [9] used the concept of correlation in probability theory [10] to compute a regression curve. Here, 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 points set.

3.1. 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

$$\rho(X,Y) = \frac{\mathrm{Cov}(X,Y)}{S(X)S(Y)} \tag{1}$$

where $\mathrm{Cov}(X,Y) = E[(X - E(X))(Y - E(Y)] = E(XY) - E(X)E(Y)$ 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 $\{P_i = (x_i, y_i), |i = 1,\ldots,N\}$, then Eq. (1) can be re-written as

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

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 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.

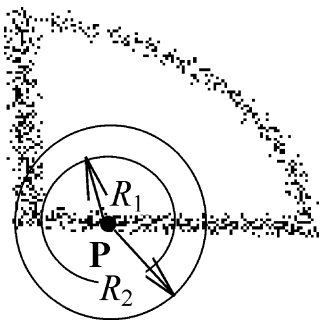


Fig. 2. Correlation coefficients of neighbourhood points of point P.

3.2. Initial point determination

The initial point (IP) is a reference point to start the construction of the first segment of the polygonal curve from planar data points. As the points are unorganised and error-filled, the IP selection is very important. Liu [8] 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. 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.

To resolve this problem, it is necessary to make sure that the points within the first neighbourhood have good linearity. In our approach, we firstly randomly select a start point, and then use a fixed radius to find its

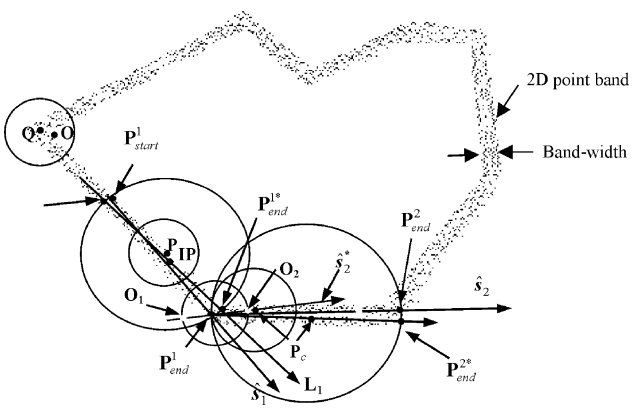


Fig. 3. IP determination and first, second segment construction.

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. 3, point Q 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.

3.3. Constructing the first line segment (S^1)

After the IP is identified, its neighbourhood (for the first line segment, 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 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 P_i from C, such that $\|P_i - IP\| \leq R$, $P_i \in C$, to form a data set C_1.
2. Compute the correlation coefficient ρ of data set C_1 using Eq. (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 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($x_{\text{IP}}, y_{\text{IP}}$) and best fits the points within the neighbourhood. Let $C_1 = \{P_i = (x_i, y_i) | i = 1, \ldots, N\}$ be the neighbourhood points, a straight line, $L_1 : y = a(x - x_{\text{IP}}) + y_{\text{IP}}$, can be computed by minimizing a quadratic function:

$$\varepsilon = \sum_{i=1}^{N} (a(x_i - x_{\text{IP}}) + y_{\text{IP}} - y_i)^2 \tag{3}$$

As shown in Fig. 3, line L_1 has two intersection points, P^{1*}_{start} and P^{1*}_{end}, with the neighbourhood circle (centred at IP with a radius R). In theory, P^{1*}_{start} and P^{1*}_{end} 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 P^{1*}_{start} and P^{1*}_{end}, respectively, within the neighbourhood, as the start and end points, i.e. the closest to P^{1*}_{start} as P^{1}_{start} and the closest to P^{1*}_{end} as P^{1}_{end}. S^1 is therefore obtained. $P^{1}_{\text{start}}P^{1}_{\text{end}}$ also defines the *unit oriented vector* of this neighbourhood ($\hat{\mathbf{s}}_1 = P^{1}_{\text{end}} - P^{1}_{\text{start}})/\|P^{1}_{\text{end}} - P^{1}_{\text{start}}\|$). Using $P^{1}_{\text{start}}P^{1}_{\text{end}}$ as the diameter, a new neighbourhood circle is obtained, we then delete all the other points within this circle. The remaining planar data set 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. 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 neighbourhood points are included for the first round, which gives a poor correlation coefficient. This leads to the reduction of R and the iteration ends with an even smaller R (with 2–3 points inside). This is certainly not desired. 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 0.001 mm. Moreover, if there are fine features on the scanned part, it is assumed that a fine resolution is used so that there are sufficient data points representing these fine features.

3.4. Constructing the remaining segments (S^i)

The method for constructing the remaining segments is slightly different from that of the first segment. We begin

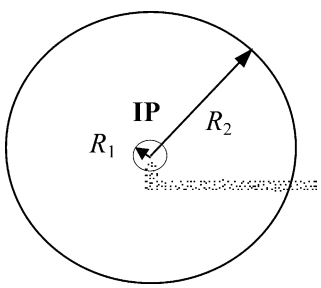


Fig. 4. Possible problems with the selection of the initial R.

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

Algorithm *find_neighbourhood_S*^{i}

Given a planar data set C, 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 P^i_{start} and has a radius of R. Select all the points P_k from C, such that $\|P_k - P^i_{\text{start}}\| \leq R$, to form a data set C_i $(k = 1, 2, \ldots, n)$. Compute ρ of data set C_i.
   If $\rho < \rho_{\text{low}}$
   $R = R - \Delta R$, go to step (1)
   Else
   Use the least-square method (Section 3.3) to compute a regression line that passes through P^i_{start}. This line has two intersection points with the neighbourhood circle, O_1 and O_2. Let $P_{\text{ave}} = \sum_{k=1}^{n} P_k/n$. If $\|P_{\text{ave}} - O_1\| > \|P_{\text{ave}} - O_2\| \hat{\mathbf{s}}^*_i = (O_2 - O_1)/\|O_2 - O_1\|$; otherwise; $\hat{\mathbf{s}}^*_i = (O_1 - O_2)/\|O_2 - O_1\|$.
   End IF
2. Construct a neighbourhood circle that is centred at $P_c(P_c = P^i_{\text{start}} + R\hat{\mathbf{s}}^*_i)$ and has a radius of R. Select all the points P_k from C, such that $\|P_k - P_c\| \leq R$, to form a data set C_i. Compute ρ of data set 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 P^i_{start} and P^{i*}_{end}, and all the points from C_i, stop.
   End if
4. Use the least-square method (Section 3.3) to compute a regression line that passes through P^i_{start}. This line has two intersection points with the neighbourhood circle, P^i_{start} and P^{i*}_{end}. Set $\hat{\mathbf{s}}^*_i = (P^{i*}_{\text{end}} - P^i_{\text{start}})/\|P^{i*}_{\text{end}} - P^i_{\text{start}}\|$. Go to step (2).

Since we do not have any prior knowledge about the neighbourhood of S^i, i.e. the unit oriented vector $\hat{\mathbf{s}}_i$, we need to find a reasonable estimate to start the iterative process. This is achieved in step (1) of Algorithm *find_neighbourhood_S*^{i}. We start by choosing a small R to create a neighbourhood circle (centred at P^i_{start} as shown in Fig. 3) such that the points within this circle have a good linearity. We then compute a regression line that passes through P^i_{start}, which help determine a good estimate of $\hat{\mathbf{s}}^*_i$. From step (2), we start with a neighbourhood circle (centred at $P_c = P^i_{\text{start}} + R\hat{\mathbf{s}}^*_i$) and adaptively find the maximal allowable neighbourhood radius. The example shown in Fig. 3 shows this process for the construction of S^2. From the final neighbourhood circle of S^2, $P^{2^*}_{\text{end}}$ is obtained. The closest point to $P^{2^*}_{\text{end}}$, within this neighbourhood, is then found and used as 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 $\hat{\mathbf{s}}^*_i$ obtained from step (1) throughout the remaining iterative process so that the computation becomes more efficient.

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

4. 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 shaper-error 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 3, 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 error 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}) $> 2\varepsilon$
   $H_{high} = H_{mid}$;
   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 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 (Fig. 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 S_h. We then obtain another layer between h and $h + \Delta h$ and project its data points to the same plane. The 2D points within Δh are placed into $S\Delta_h$. An example of these two sets of data points is shown in Fig. 5. For every point in S_h, P_i $(i = 1, 2, \ldots, n)$, we then find the corresponding point (Q_i) in $S\Delta_h$, such that among all the points in $S\Delta_h$, P_iQ_i gives the minimum distance (L_i). Then, among all the $L_i (i = 1, 2, \ldots, n)$, the maximum one ($L_{max}$) is taken as the band-width of S_h. Using this estimation method, we find that the points in S_h that give the maximum distances are along the interior boundary of 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. 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.

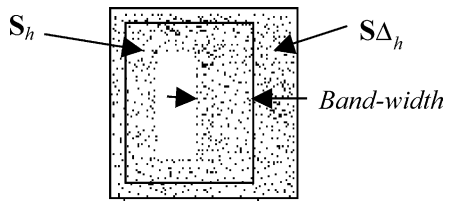


Fig. 5. Estimation of the *band-width* of the 2D data points.

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 properly. In this case, a different slicing direction should be chosen manually.

5. Application examples

The algorithm described above has been implemented with C/C++ in the OpenGL environment. Three case studies are presented here to illustrate the efficacy of the algorithm for constructing the direct RP model. The first two 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. While the third one is a real case and its cloud data are obtained by a laser scanner, and the results after processing are input to a RP machine for fabrication.

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

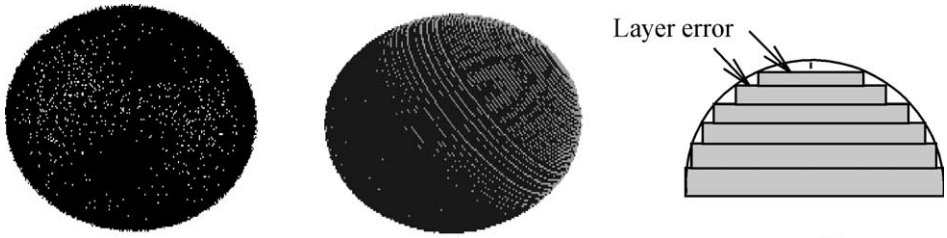


Fig. 6. The original cloud data and the direct RP model in case study one.

to simulate noise in the point cloud):

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

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. 6a.

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 error tolerance of 0.08, the direct RP model of the sphere shown in Fig. 6b is obtained. This model contains 11,812 vertices distributed in 74 layers. Fig. 7a 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 74 layers according to the layer thickness in the generated model and the theoretical shape error, also known as stair stepping error as Fig. 6c shows, is shown in Fig. 7b. It can be seen that the theoretical errors are close to 0.08 too (except the two tip areas).

The second case study uses an object composed of 4 spherical patches (Fig. 8a). The parameter of larger sphere is the same as in the first case study, and the equations of three smaller half-spheres are based on the following basic form:

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

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.

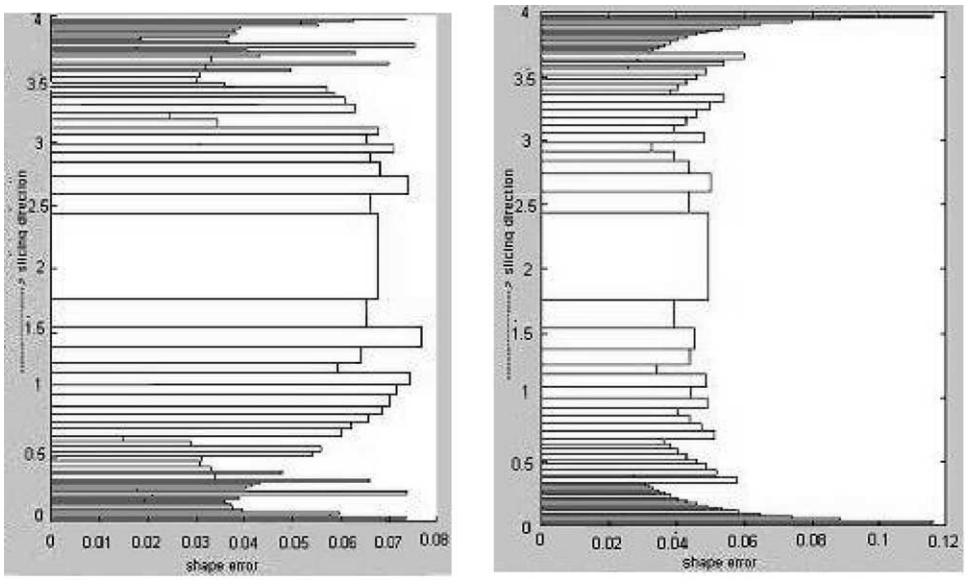
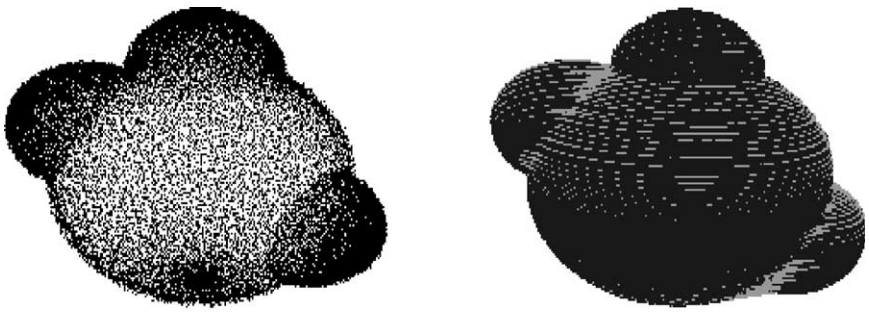


Fig. 7. Shape error comparison in case study ($\varepsilon = 0.08$).



(a) (b)

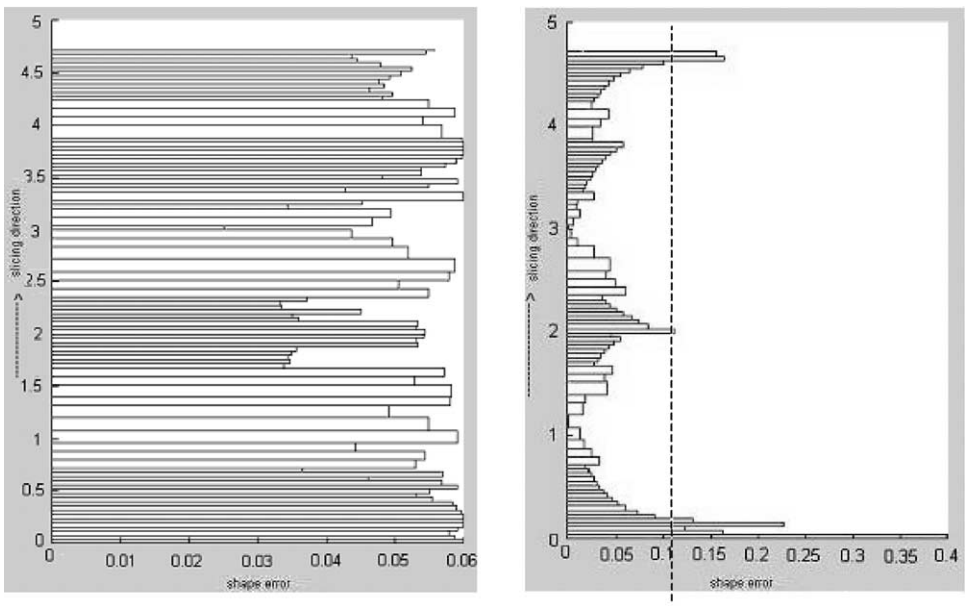
Fig. 8. The original cloud data and the direct RP model of second case study.

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 totally 46,057 points. The original object is shown in Fig. 8a.

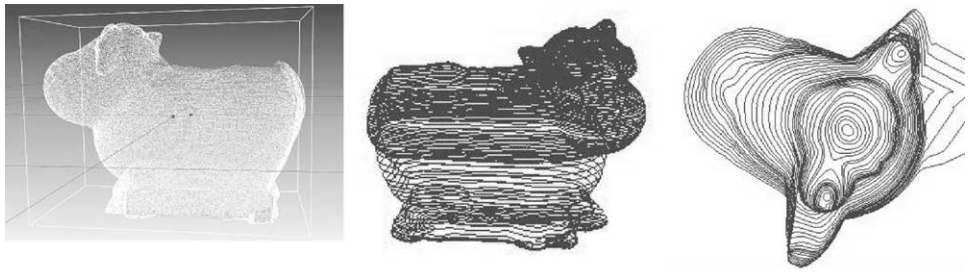
The initial layer thickness is set at 0.04, and the initial neighbourhood radius is 0.1. ρ_{low} and ρ_{high} are set as 0.85 and 0.9, respectively. Employing a shape error tolerance of 0.06, the direct RP model of the sphere shown in Fig. 8b is obtained. This model contains 21,306 vertices distributed in 88 layers. Fig. 9a 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. 9b 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 closed 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. 9b.

The third case study is of a toy-cow as shown in Fig. 10a. The original object occupies a volume of $150 \times 120 \times 90$ mm^3 and was digitised by a laser scanner, Minolta VIVID-900 digitizer. The data sets were obtained from different view angles 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



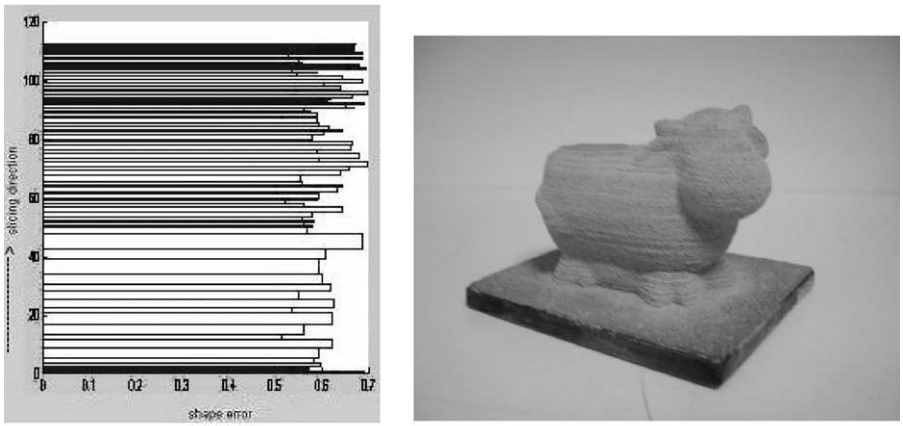
(a) (b)

Fig. 9. Shape error comparison in case study two ($\varepsilon = 0.06$).;



(a) (b) (c)

Fig. 10. The original cloud data, the direct RP model, and a zoom-in view at the head.



(a) (b)

Fig. 11. (a) Shape error of the direct RP model in case study three (b) The toy-cow fabricated by RP machine.

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 in Fig. 10b with 115 layers and 59,686 points. The shape error of each layer in the generated model is shown in Fig. 11a 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 poses a multiple-loop problem. This can be seen clearly in Fig. 10c, in which the multiple-loops are separated successfully and the corresponding layers are generated.

It took about 30 min for the adaptive slicing algorithm to generate the RP model using a PC of 1.5 GHZ 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. 11b shows the real work-piece fabricated by the RP machine based on the direct RP model. However, if a RP machine that can deposit material with different layer thickness could be used, time-saving could be up to 78.5%.

6. Conclusions

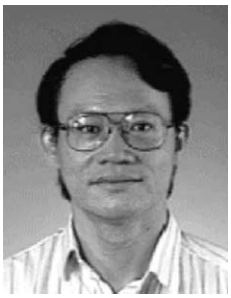
In this paper, a method for generation direct RP models from arbitrarily scattered cloud data is presented. The modelling process consists of several steps: (1) the cloud data are segmented into several layers along the RP building direction, (2) points within each layer is treated as planar data and 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 has been implemented. Testing results on both simulated and real cases show that the algorithm is effective.

The main contribution of this paper is two-fold. First, the polygonal curve construction algorithm is adaptive in nature. It is capable of automatically finding a feasible starting point and identifying the maximally 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.

References

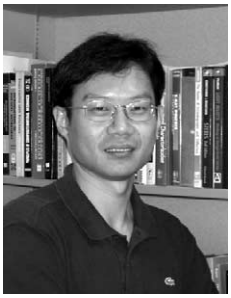
[1] Varady T, Martin RR, Cox J. Reverse engineering of geometric models-an introduction. CAD 1997;29(4):255–68.
[2] Li L, Schemenauer N, Peng X, Zeng Y, Gu P. A reverse engineering system for rapid manufacturing of complex objects. Robotics Comput Integr Manufact 2002;18(1):53–67.
[3] Yan X, Gu P. A review of rapid prototyping technologies and systems. CAD 1996;28(4):307–18.
[4] Lee KH, Woo H. Direct integration of reverse engineering and rapid prototyping. Comput Ind Engng 2000;38(1):pp21–pp38.
[5] Chen YH, Ng CT, Wang YZ. Generation of an STL file from 3D measurement data with user-controlled data reduction. Int J Adv Manufact Technol 1999;15(2):127–31.
[6] Lee SH, Kim HC, Hur SM, Yang DY. STL file generation from measured point data by segmentation and Delaunay triangulation. CAD 2002;34(10):691–704.
[7] Sun W, Bradley C, Zhang YF, Loh HT. Cloud data modelling employing a unified, non-redundant triangular mesh. CAD 2001; 33(2):183–93.
[8] Liu GH. Segmentation of cloud data for reverse engineering and direct rapid prototyping. Master of Engineering Thesis, National University of Singapore; 2001
[9] Lee I-K. Curve reconstruction from unorganized points. Comput Aided Geom Des 2000;17(2):161–77.
[10] Pitman J. Probability. Berlin: Springer; 1992.

Y.F. Wu received a BS in Ship and Sea Engineering from Wuhan Transportation University, China in 1998 and an MS in Mechanical Engineering from Huazhong University of Science and Technology, China in 2001. He is currently a postgraduate student in the National University of Singapore. His research interests include CAD/CAM and reverse engineering.



Y.S. Wong is Associate Professor in the Mechanical Engineering Department of the National University of Singapore. His research interests are in process characterization for modelling, monitoring, control and optimization and rapid product data capture for design and manufacture. In these areas, he has published in refereed journals and proceedings of international conferences, contributed to books, and participated in several local and overseas funded projects.

Dr H.T. Loh is an Associate Professor in the Department of Mechanical Engineering at the National University of Singapore (NUS). He is the Deputy Director (Education) of the Design Technology Institute. He is also a Fellow of the Singapore-MIT Alliance, which is an innovative engineering education and research collaboration between MIT, NUS and the Nanyang Technological University, to promote global education and research in engineering. Dr Loh's research interests are in the areas of data mining, rapid prototyping, robust design and computer aided design.



Y.F. Zhang received his BEng from Shanghai Jiao Tong University, China in 1985 and PhD from the University of Bath in 1991. He is currently an Associate Professor at the Department of Mechanical Engineering, National University of Singapore. He is a senior member of SME. His research interests include reverse engineering, multi-axis tool path planning, and the computational intelligence in design and manufacturing.