ELSEVIER

Journal of Materials Processing Technology 138 (2003) 53–57

Journal of Materials Processing Technology

www.elsevier.com/locate/jmatprotec

Modelling cloud data for prototype manufacturing

G.H. Liu, Y.S. Wong, Y.F. Zhang*, H.T. Loh

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

Abstract

In this paper, the authors have developed a novel method to integrate reverse engineering (RE) and rapid prototyping (RP). Unorganised cloud data are directly sliced and modelled with two-dimensional (2D) cross-sections. Based on such a 2D CAD model, the data points are directly converted into RP slice data and fed to an RP machine for fabricating. In this process, neither a surface model nor a STL file is generated. This is accomplished from the 3D data points in several steps: first, the cloud data are sliced into a number of layers along a user-specified direction. The points in each layer are projected onto a plane. Secondly, the points on each plane are sorted and compressed. Data point smoothing is then carried out using a discrete curvature based method. Thirdly, a local interpolating method is used for adding additional points to the slice-lines having insufficient points. Fourthly, the cross-sections between every two neighbouring planes are created by directly connecting the feature points (FPs) with straight-line segments. Finally, an RP layered file is generated for an SLA machine. The developed methods have been implemented with C/C++ on the Unigraphics platform.

Keywords: Cloud data; Rapid prototyping; Reverse engineering; Segmentation

1. Introduction

Reverse engineering (RE) refers to creating a CAD model from an existing physical object, which can be utilised 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 product model designed by the stylist, usually in the form of wood or clay mock-up is first 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 utilising optical non-contact measuring techniques, e.g. laser scanner. This normally produces a large cloud data set that is usually arbitrarily scattered.

Approaches to transform a dense unstructured data set to a CAD representation can be classified into two categories: triangular polyhedral mesh based method and segment-and-fit based method [1–3]. In the former approach, an initial triangular mesh is constructed to capture the unknown topological structure of the scattered data. Then the mesh is optimised to reduce the redundant vertices and afterwards a curvature-continuous surface is reconstructed based on this structure. Many triangulation techniques can be found in the literature, e.g. Delaunay triangulation algorithm [4,5], triangulation based on signed distance function [6] and triangulation based on α-shapes [7]. Triangulation for cloud data is however a computationally inefficient process [6,8].

For the second, the cloud data is divided into a suitable patchwork of surface regions to which an appropriate single surface is fitted [9]. Since manually segmenting 3D measured data is a laborious and error-prone process, some researchers have attempted to implement an automatic segmentation algorithm [2,9,10]. Nevertheless, in general, present segmentation algorithms are sensitive, computationally complex or can only be applied to simple topology data [10].

Hence, in recent years, some novel RE approaches take into account a direct manufacturing of cloud data without involving surface reconstruction for more efficient rapid product development (RPD) [8]. However, these algorithms can only be applied to structured point data.

In this paper, the authors propose an error-based segmentation approach to arbitrarily scattered cloud data. Our goal is to integrate RE and rapid prototyping (RP) to assist manufacturers and designers in meeting the demands of reduced product development time. This is accomplished in the following steps. First, the cloud data are sliced into a number of layers along a user-specified direction. The points in each layer are projected onto a plane. Secondly, the points on each plane are sorted and compressed. An initial curve for each layer (so-called *C*-curve) is then generated. Thirdly, a local interpolating method (constructing *R*-curves using the

* Corresponding author.
E-mail address: mpezyf@nus.edu.sg (Y.F. Zhang).

0924-0136/03/$ – see front matter
doi:10.1016/S0924-0136(03)00048-7

corresponding points in each layer) is used for adding additional points to the slice-lines having insufficient points. Finally, a layer-based RP model is constructed and directly fed to RP machine for prototype manufacturing. Based on these techniques, experiment results are presented to illustrate the efficacy of the developed approach.

2. Cloud data segmentation

In essence, a *C*-curve is made up of a series of feature-point based planar curves. It is constructed in three steps. First, cloud data is pre-processed through slicing, projecting and sorting to generate initial *C*-curve. Secondly, the initial *C*-curve is compressed by removing all the redundant points except feature points (FPs) that represent the original shape information recorded in the data link. Lastly, the *C*-curve is constructed by linking all the FPs using straight-line segments. In the constructing process of these two curves, the shape accuracy is controlled by several user-specified parameters.

2.1. Data pre-processing

The first step towards constructing *C*-curve is to directly slice cloud data along a user-specified slicing direction and interval. The whole data set is then uniformly sliced into many subsets by computing the projecting errors of the points. Suppose that user-defined slicing direction and a data subset is $X = \{P_1, P_2, \ldots, P_n\}$, respectively. The centre of the data set X is calculated by

$$P_c = \frac{1}{n}\sum_{i=1}^{n} P_i \tag{1}$$

Assume that point $O \in X$ is the closest to P_c, denote a plane associated with the centre point O and unit normal vector $\hat{\boldsymbol{n}}$ by $T(O, \hat{\boldsymbol{n}})$, we define $T(O, \hat{\boldsymbol{n}})$ to be the projecting plane of data set X. The projecting error of point $P_i \in X$ to $T(O, \hat{\boldsymbol{n}})$ is then calculated by

$$e_i = |(P_i - O) \cdot \hat{\boldsymbol{n}}| \tag{2}$$

If e_i is greater than the pre-defined shape error, data set X is uniformly re-subdivided into two subsets. This process repeats for all the subsets until all e_i meet the demand.

After slicing, all the points in the subset are projected onto the corresponding projecting planes and a "cleaning" process is carried out to sort the projected points. The aim of sorting is to set up a topological structure for the planar data as well as remove spurious data such as peaks and duplicate data to construct initial *C*-curves in the projecting plane. Due to the projected data points in the layer are unorganised and error-filled, it is important to pick up a correct point as the start in the sorting algorithm. The sorting algorithm is based on an estimated oriented tangent vector, which consists of five steps as follows:

(1) Estimate the oriented tangent vector of each point.

Suppose $P_i \in X$ is a point in the plane, we call k points of X nearest to P_i to be the k-neighbourhood of P_i, and denote this data set with $\mathrm{Nhbd}_{i\in\{1,\ldots,k\}}(P_i)$. k is an user-specified parameter used to control the estimating accuracy of the originated tangent vector. As shown in Fig. 1(a), the originated tangent vector of P_i is calculated as

$$\hat{\boldsymbol{s}}_i = \frac{P_c - P_i}{\| P_c - P_i \|} \tag{3}$$

where P_c is the centre of $\mathrm{Nhbd}_{i\in\{1,\ldots,k\}}(P_i)$ determined by Eq. (1).

(2) Retrieve the next point of the start point.

Any point $P_i \in X$ can be selected as the first start point. Assume that the first start point and its oriented tangent vector is P_0, $\hat{\boldsymbol{s}}_0$, respectively (Fig. 1(b)), the retrieving algorithm would start with P_0 and $\hat{\boldsymbol{s}}_0$. When an end point is detected, algorithm goes back to P_0 and retrieving resumes in the direction of $-\hat{\boldsymbol{s}}_0$, and stops when the other end point is determined.

For each point $P_i \in X$ and its oriented tangent vector $\hat{\boldsymbol{s}}_i$, we determine the next point of P_i, P_{i+1}, by

$$\mathrm{dist}(P_{i+1}) = \min_{P_k \in X}(|(P_k - P_i) \cdot \hat{\boldsymbol{s}}_i|) \le \rho \tag{4}$$

where ρ is the given shape error. P_{i+1} is then taken as a new start point and retrieving continues with this point and its oriented tangent vector.

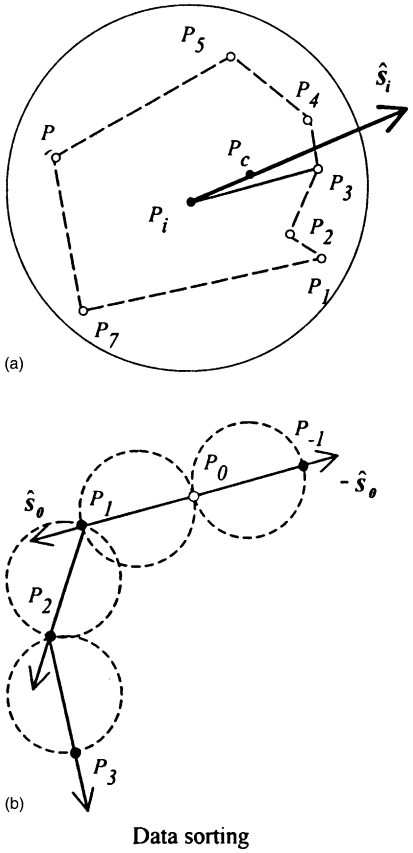


Fig. 1. Data sorting.

(3) Delete the spurious points.

In *step* (2), when the next point of the start point is determined, we define a circle using these two points whose diameter is equal to the distance of the points and delete all the points within the circle.

(4) Terminate the sorting algorithm.

Steps (1)–(3) form a retrieving cycle. When a cycle is completed, all the retrieved points in the cycle are flagged and linked together to form a *C*-curve. If all points in the plane are flagged, the algorithm terminates. Otherwise, a new retrieving cycle resumes with an un-flagged point that produces a new *C*-curve in the plane. The number of *C*-curves in different projecting planes may be different.

2.2. Data compressing

Data pre-processing constructs initial *C*-curves for each projecting plane. However, the data size of the curve is extremely large that could plummet the computational robust and exceed the limited memory storage of the hardware. To resolve this, a feature-point based compressing algorithm is developed to compress redundant points on each plane using parametric cubic polynomial. To obtain a solution for a parametric form of a cubic polynomial curve, there should be at least four points. Hence, for each projecting plane, the *C*-curve is taken into account only when the corresponding point number is more than 4. Employing a least-square method, the unknown coefficient of the cubic polynomial can be obtained.

Since the basic shape information of a planar curve is exhibited by its FPs: corners or high curvature points, we define the parameter values corresponding to the FPs of the approximating curve as follows:

$$\mathrm{FP} = \begin{cases} \text{end point} : u_f = 0, 1 \\ \text{peak point} : u_f = \{\zeta | f'(\zeta) = 0\} \\ \text{inflection point} : u_f = \{\eta | f''(\eta) = 0\} \end{cases}$$

The present compressing algorithm is based on the above-mentioned FPs, which consists of the following steps (see Fig. 2):

(1) Find all the FPs.

(2) Link all the FPs with straight-line segments to generate a polygon.

(3) Calculate the distances from the points in the *C*-curve to the polygon. If the distance is greater than the pre-defined shape error, this point is added as a new FP.

(4) Go to *step* (2). This process repeats until all the points in the curve satisfy that their corresponding distances are within the given shape error.

All the points except FPs are deleted. The final *C*-curve is then constructed by connecting all FPs using straight-line segments.

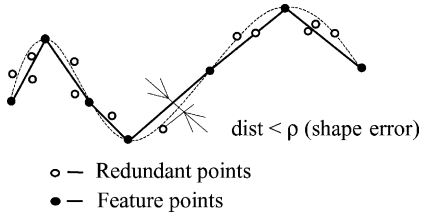


Fig. 2. Data compressing.

3. Data interpolation

After the *C*-curves are established, the data in the *C*-curves serve to construct curves across the layers (rows) to form the so-called *R*-curves. In essence, the goal of construction of *R*-curve aims to establish a topological structure for points in rows, thus assisting in interpolating new points for *C*-curves with inadequate points. Take the example shown in Fig. 3(a), the construction of *R*-curve is based

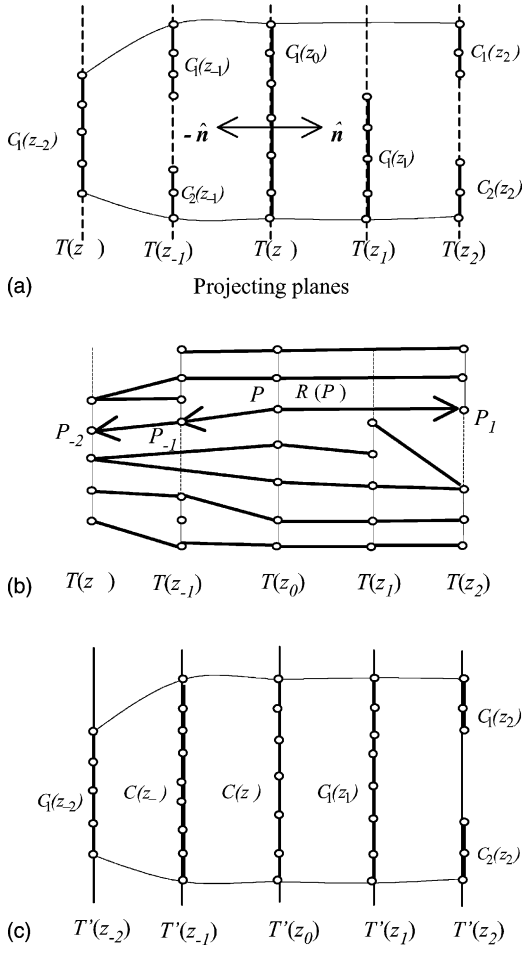


Fig. 3. Data interpolation.

on an estimated tangent plane, which consists of the following steps:

(1) Estimate the tangent plane of each point.

Suppose point $P_i \in X$, denote m-neighbourhood of P_i with $\mathrm{Nhbd}_{i\in\{1,\ldots,m\}}(P_i)$, and m is a user-specified parameter used to control the estimating accuracy of the tangent plane. The unit normal vector of the tangent plane can be determined by calculating the unit eigenvectors of a 3×3 symmetric matrix $\boldsymbol{A}$ as described as follows:

$$\boldsymbol{A} = \sum_{x_k \in \mathrm{Nhbd}(x_i)} (P_k - P_i)\cdot(P_k - P_i)^{\mathrm{T}} \tag{5}$$

If $\lambda_i^1 \geq \lambda_i^2 \geq \lambda_i^3$ denote the eigenvalues of $\boldsymbol{A}$ associated with unit eigenvectors $\hat{\boldsymbol{v}}_i^1, \hat{\boldsymbol{v}}_i^2, \hat{\boldsymbol{v}}_i^3$, respectively, the estimated tangent plane is determined by $T_{\mathrm{p}}(P_i, \hat{\boldsymbol{v}}_i^3)$.

(2) Identify the oriented points in rows.

Denote C-curve with $C_i(z)$, R-curve with $R_i(p)$, point data set with X and the tangent plane of point $P_i \in T(z_0)$ with $T_{\mathrm{p}}(P_i, \hat{\boldsymbol{v}}_i)$. Assume that the algorithm commences with projecting plane $T(z_0)$, as shown in Fig. 3(b), the oriented point of P_i in rows is determined by searching all the points $p \in X$ in C-curves using

$$\mathrm{dist}(p) = \min_{p\in X}(p - P_i)\cdot\hat{\boldsymbol{v}}_i \leq \rho \tag{6}$$

where ρ is the given shape error. If the above inequality is tenable, p is termed as the next/front row point of P_i. Otherwise P_i has no oriented row points. In order to improve the efficiency, only points in n-neighbourhood of projecting plane $T(z_i)$ are selected for calculating (n is specified by user). If n is selected along the positive normal of $T(z_i)$, the oriented point identified in rows is defined as the positive/next row point of P_i. If n is done along correspondingly negative normal, the identified point is the negative/front point of P_i. In this way, for each point in the C-curve, its two oriented points in rows can be determined.

(3) Generate R-curves.

Based on the point relationship in rows determined in *step* (2), generation of R-curve is carried out plane by plane in two directions. In our algorithm, a projecting plane that contains the maximal number of points is chosen as the start position. As show in Fig. 3(b), suppose that the plane is $T(z_0)$, we take point P_0 as an example. First, along the positive normal direction $\hat{\boldsymbol{n}}$, the oriented point of P_0, P_1, is linked as the next row point with P_0. Because there is no more rows in this direction, the algorithm returns to P_0 and searches for the oriented points along the negative direction $-\hat{\boldsymbol{n}}$. P_{-1} is thus added into the link as a front row point of P_0. Searching continues with P_{-1} and hereby P_{-2} is added. The data link $P_{-2} \rightarrow P_{-1} \rightarrow P_0 \rightarrow P_1$ is an R-curve. R-curves are constructed in this way by retrieving all the points in C-curves (Fig. 3(b)).

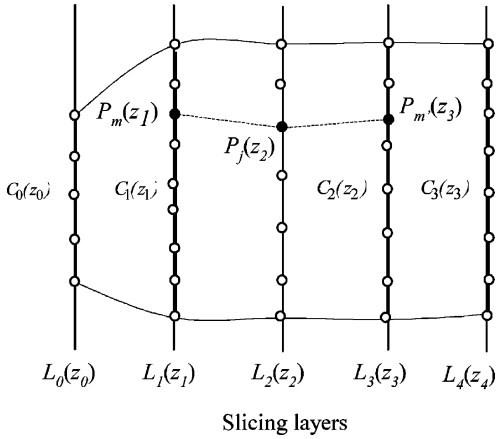


Fig. 4. Data points in slicing layers and C-curves.

(4) Interpolate new points.

The constructed R-curves are then intersected with every plane. A new point is therefore added to the respective C-curve on a plane if it does not exist before the intersection (Fig. 3(c)).

The achieved C-curves are then faired using an algorithm based on discrete curvatures.

4. Construction of RP model

Although STL file has been the de facto industrial standard for RP machine, layer-based slice file such as CLI or SLC file format is also acceptable by RP machine. Here, the RP model is constructed based on the faired C-curves using a constant slicing strategy. As shown in Fig. 4, assume that the slicing layer $L_i(z_i)$ lies between C-curve $C_i(z_k)$ and $C_{i+1}(z_k{}')$, the corresponding point in $L_i(z_i)$, $P_j(z_i)$, is calculated by

$$P_j(z_i) = \frac{z_k{}' - z_i}{z_k{}' - z_k}\cdot P_m(z_k) + \frac{z_i - z_k}{z_k{}' - z_k}\cdot P_{m'}(z_k{}') \tag{7}$$

where $P_m(z_k)$ and $P_{m'}(z_k{}')$ are the two nearest points located in $C_i(z_k)$ and $C_{i+1}(z_k{}')$, respectively.

When all the points for slicing layers are generated, points in each layer are closed to generate an RP model that is directly sent to an RP machine for fabrication.

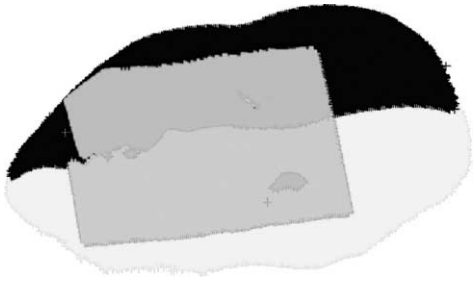
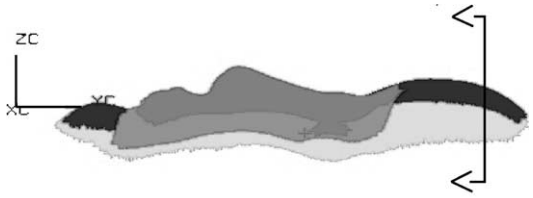
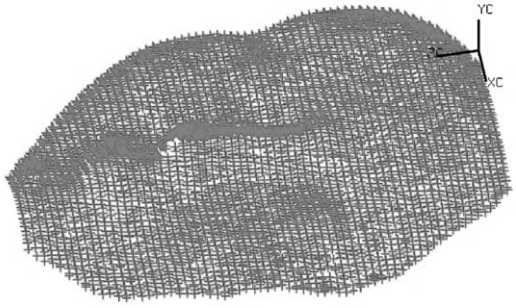


Fig. 5. The original data cloud.



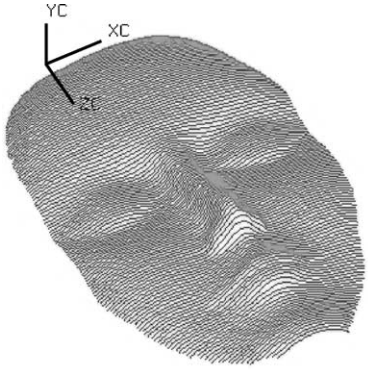
AA' is the slice direction defined by user

(a) Original data with the slicing direction

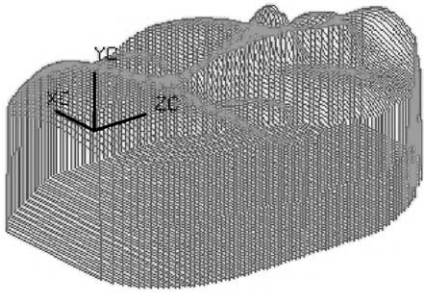


(b) Data points after sorting and compressing

Fig. 6. The modelling process.



(a) The final *C*-curve model



(b) The RP model

Fig. 7. The final models.

5. Application example

The algorithm described above has been implemented with C/C++ on HP-C200 workstation in the Unigraphics environment. A case study is presented here to illustrate the efficacy of the algorithm. The case is of a mask, which is composed of four range data patches. The original data cloud contains 104,175 points (see Fig. 5).

The slicing direction was given interactively as shown in Fig. 6(a). Based on this direction, the cloud data is processed to construct an RP model. Using a user-specified shape error $\varepsilon_{\text{shape}} = 0.1$ mm, the thickness of each layer is chosen as 0.2 mm. After data sorting and compressing, only 16,324 points are kept for modelling (see Fig. 6(b)). It took about 1 h for the algorithm to run on the HP-C200 workstation. The final *C*-curved model is shown in Fig. 7(a) and the constructed RP model in Fig. 7(b). This model is fed to an SLA RP machine and needs approximate 3 h to complete the fabrication.

6. Conclusions

In this paper, a new approach to deal with arbitrarily scattered cloud data for RP is presented. In our approach, a layer-based RP model is directly extracted from the cloud data through slicing, projecting, compressing and intersecting. Neither a surface model nor the STL file is generated. The method has been proven effective in dealing with complex surfaces and computational efficient through experiments. However, our approach currently can only handle cloud data that has no holes and lobes. Research on the multi-loops caused by the slicing of cloud data is still under developing. On the other hand, research on setting the slicing direction and other parameters automatically is also underway.

References

[1] T. Varady, R.R. Martin, J. Coxt, Reverse engineering of geometric models—an introduction, Comput. Aided Des. 29 (4) (1997) 255–268.
[2] B. Sarkar, C.-H. Menq, Smooth-surface approximation and reverse engineering, Comput. Aided Des. 23 (9) (1991) 623–628.
[3] R. Hoffman, K. Jain, Segmentation and classification of range images, IEEE Pattern Anal. Mach. Intell. 9 (5) (1987) 608–620.
[4] P. Cignoni, C. Montani, R. Scopigno, A fast divide and conquer Delaunay triangulation algorithm, Comput. Aided Des. 30 (5) (1998) 333–341.
[5] H. Park, K. Kim, An adaptive method for smooth surface approximation to scatter 3D points, Comput. Aided Des. 27 (12) (1995) 929–939.
[6] H. Hoppe, T. Derose, Duchamp, surface reconstruction from unorganized points, Comput. Graphics 26 (2) (1992) 71–76.
[7] B. Guo, Surface reconstruction from points to splines, Comput. Aided Des. 29 (4) (1997) 269–277.
[8] A.C. Lin, H.-T. Liu, Automatic generation of NC cutter path from massive data points, Comput. Aided Des. 30 (1) (1998) 77–90.
[9] M.J. Milroy, C. Bradley, G.W. Vickers, Segmentation of a wrap-around model using an active contour, Comput. Aided Des. 29 (4) (1997) 299–320.
[10] S. Liu, W. Ma, Seed-growing segmentation of 3-D surfaces from CT-contour data, Comput. Aided Des. 31 (1999) 517–536.