Int J Adv Manuf Technol (1999) 15:127–131

The International Journal of
Advanced
Manufacturing
Technology

Generation of an STL File from 3D Measurement Data with User-Controlled Data Reduction

Y. H. Chen, C. T. Ng and Y. Z. Wang

Department of Mechanical Engineering, The University of Hong Kong, Hong Kong

Reverse engineering is a methodology for constructing computer-aided design (CAD) models of physical parts by digitising an existing part, creating a computer model and then using it to manufacture the component. When a digitised part is to be manufactured by means of rapid prototyping machines such as stereolithography apparatus (SLA) and selective laser sintering equipment (SLS), etc., it is not necessary to construct the CAD model of a digitised part. This can be achieved by the proposed novel method which can construct an STL file (the de facto *file format for rapid prototyping machines) directly from digitised part data. Furthermore, the STL file can be constructed with a significant data reduction at the users' discretion.*

Keywords: Rapid prototyping; Reverse engineering; STL file; Triangulation

1. Introduction

Reverse engineering is an important process in the design and manufacturing context. This can be illustrated by two examples. First, a product that exists in a designer's medium, such as clay or wood, must have its surfaces digitised so that it can be converted to a computer-based representation (compatible with computer-aided design (CAD)/computer-aided manufacturing (CAM) systems). Secondly, if part drawings are not available in CAD form, such as for some proven old designs or antiques, CAD models of such parts can be constructed through reverse engineering techniques. Given a CAD model of a part, traditional methods such as CNC machining can be used to produce the parts. However, it is sometimes impossible for CNC machines to machine complicated surfaces both rapidly and cost effectively. Over the past a few years, rapid prototyping machines have been widely used in industry. A variety of rapid prototyping technologies have emerged [1–3]. They include stereolithography (SLA), selective laser sintering (SLS), fused deposition manufacturing (FDM), laminated object manufacturing (LOM), ballistic particle manufacturing (BPM), and three-dimensional printing (3D printing). These technologies are capable of directly generating physical objects from a CAD model. They have an important common feature: physical parts are produced by adding materials layer by layer. This is the opposite of traditional machining methods which make physical parts by removing material.

In rapid prototyping, the STL file format has become the *de facto* standard [4]. An STL file consists of a list of triangular facet data. Each facet is uniquely identified by a unit normal and three vertices. The normal and each vertex are specified by three coordinates each, so there is a total of 12 numbers stored for each facet. Apart from this, triangular facets in an STL file must also obey the following two rules.

1. *Facet Orientation* The facets define the surfaces of a 3D object. Each facet is part of the boundary between the interior and the exterior of the object. The orientation of the facet is specified redundantly in two ways which must be consistent. First, the direction of the normal is outward. Second, the vertices are listed in counterclockwise order when looking at the object from the outside. This rule is illustrated in Fig. 1(*a*).

2. *Vertex-to-Vertex* Each triangular facet must share two vertices with each of its adjacent triangles. In other words, a vertex of one triangle cannot lie on the side of another. This rule is illustrated in Figs. 1(*b*) and 1(*c*).

Most recent commercial CAD/CAM software systems are capable of generating STL files directly from a surface model. The majority of triangulation methods are based on a known surface model [5, 6]. Triangulation of scattered data in 3D space often has the objective of constructing smooth surfaces [7, 8]. Literature about optimised STL file generation from reverse engineering data cannot be found. Humann's paper [9] is the most relevant. His proposal removes a triangle based on curvatures at the three vertices of a triangle. Again, his research has smooth surface fitting in mind. This paper describes a data reduction method for automatic STL file generation directly from reverse engineering data. In order to allow more flexi-

Correspondence and offprint requests to: Dr Y. H. Chen, Department of Mechanical Engineering, The University of Hong Kong, Haking Wong Building, Pokfulham Road, Hong Kong. E-mail: yhchen@hkucc.hku.hk

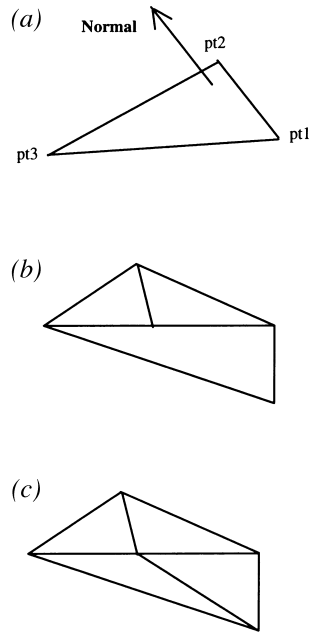

Fig. 1. STL file rules. (*a*) Orientation of a triangular facet. (*b*) Violation of vertex-to vertex rule. (*c*) Correct triangulation.

bility, data reduction with both percentage and bounded errors are implemented.

2. Automated STL File Generation

There are various methods for part digitisation. We use a Mitutoyo BLN 122 coordinate measuring machine (CMM) to digitise a part. In order not to miss any detail of the part geometry, a large number of measurement points are normally defined. If all points are used in STL file generation, the file can easily become so large that the whole rapid prototyping process will be slowed down significantly. Keeping lots of data points in nearly planar regions is unnecessary. Removal of data points in these regions would not affect the accuracy using rapid prototyping machines. A criterion based on normals of neighbouring triangles has been defined to determine which points can be removed in accordance with users' requirements.

Point data from CMM measurement is usually very regular and can be arranged in matrix form. If all measurement points are to be used for STL file construction, the problem becomes quite simple. In fact, the proposed method initially uses all points to form triangles in the following simple way:

```
Given a data point matrix M(i,j)
  where i = 1, …, m;   j = 1, …, n
  For j = 1 to n
    for i = 1 to (m − 1)
      join M(i,j) to  M(i + 1,j);
  For j = 1 to (n −1)
    for i = 1 to m
      join m(i,j) to M(i,j + 1);
    for i = 1 to (m−1)
      join M(i,j) to M(i+1, j+1);
End.
```

With the above procedure, data points can be triangulated as shown in Fig. 2. It can be seen that triangles formed in this way conform to the STL file vertex-to-vertex rule.

To generate a complete STL file, normalised surface normals of triangles must be specified. Given a triangle as shown in Fig. 1(*a*) (pt_1, pt_2, and pt_3 must be arranged in a counterclockwise direction), normalised normals can be calculated from the following procedure.

1. Form two vectors $\mathbf{V}_{31}$ and $\mathbf{V}_{32}$:

$$\mathbf{V}_{31} = \mathbf{P}t_1 - \mathbf{P}t_3$$
$$\mathbf{V}_{32} = \mathbf{P}t_2 - \mathbf{P}t_3 \quad (1)$$

2. Calculate the surface normal from the cross-product:

$$\mathbf{n} = \mathbf{V}_{31} \times \mathbf{V}_{32} \quad (2)$$

3. Normalise $\mathbf{n}$ through the division:

$$\hat{\mathbf{n}} = \frac{\mathbf{n}}{|\mathbf{n}|} \quad (3)$$

With the above information, the STL file can be readily generated.

3. Data Reduction

3.1 Data Reduction in Percentage

If the digitised object has large planar or near planar regions, it is desirable to remove some of the points in these regions. In order to identify the points that can be removed, a point weighting scheme based on a point's surrounding triangles' normals is developed and described as follows:

1. Search all triangles Δ that share the point concerned.
2. Let one of the surface normals $\mathbf{U}$ in Δ be the reference vector, calculate substractions of $\mathbf{U}$ with the rest of surface normals in Δ.

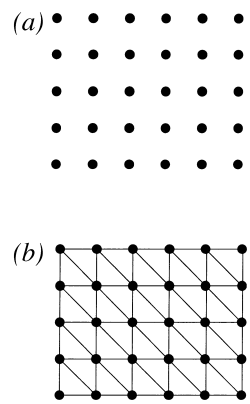

Fig. 2. Initial triangulation. (*a*) Data points. (*b*) Triangulation.

$$\mathbf{UV}_1 = \mathbf{V}_1 - \mathbf{U}$$
$$\mathbf{UV}_2 = \mathbf{V}_2 - \mathbf{U}$$
$$\vdots$$
$$\mathbf{UV}_n = \mathbf{V}_n - \mathbf{U} \quad (4)$$

3. Transform all the resulting vectors to the first quadrant by taking the absolute values of all vector components, e.g. $\mathbf{UV}_2 = a\mathbf{i} + b\mathbf{j} + c\mathbf{k}$ is transformed to $\mathbf{UV}_2 = |a|\mathbf{i} + |b|\mathbf{j} + |c|\mathbf{k}$.
4. Sum all the resulting vectors.

$$\mathbf{W} = \mathbf{UV}_1 + \mathbf{UV}_2 + \ldots + \mathbf{UV}_n \quad (5)$$

5. Calculate the average quantity d by

$$d = \frac{|\mathbf{W}|}{n}\mathbf{V}_n \quad (6)$$

If d is small for a given point, the point stands a higher chance of being removed. Which points are to be removed depends on the users' specifications. If a user wants 10% of the point data to be removed, then points with the lowest 10% of d will be removed.

3.2 Data Reduction by Bounded Error

In computer visualisation, polygonal meshes, especially triangles are usually used to approximate object surfaces. In the triangulation of an object surface, accuracy is often controlled by a bounded error [10]. The bounded error concept is also adapted here to control errors resulting from data reduction of an STL file in rapid prototyping.

In data reduction, two strategies can be used. One is to remove the points one at a time. The other is to remove several points, or a patch, at the same time. From consideration of error accumulation and computational efficiency, the patch removal strategy is implemented here.

Suppose a patch has k triangles and N points. The normals and areas of the k triangles are $\mathbf{n}_1$ to $\mathbf{n}_k$ and S_1 to S_k, respectively. Let

$$S = \sum_{i=1}^{k} S_i \quad (7)$$

A weighted patch direction normal $\hat{\mathbf{n}}$ can be defined as:

$$\mathbf{n} = \sum_{i=1}^{k} \frac{S_i}{S}\mathbf{n}_i \quad (8)$$

$$\hat{\mathbf{n}} = \frac{\mathbf{n}}{|\mathbf{n}|} = (\cos\alpha)\,\mathbf{i} + (\cos\beta)\,\mathbf{j} + (\cos\gamma)\,\mathbf{k} \quad (9)$$

$\hat{\mathbf{n}}$ indicates that each individual triangle in the patch tries to pull the patch normal towards its own normal direction. The larger the area of a triangle, the more it pulls. Based on the patch direction normal, a minimum enclosing box of a patch can be calculated. Let the heights of the box before and after data reduction be D_b and D_a, respectively, as shown in Fig. 3.

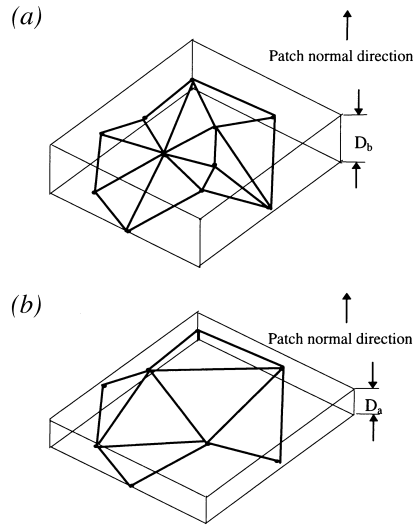

Fig. 3. Error definition. (*a*) Before removal. (*b*) After removal.

The bounded error E_p is measured along the patch direction normal, that is

$$E_p = D_b - D_a \quad (10)$$

This error can be specified by the user so that the accuracy after data reduction is still within the user-specified limit.

To calculate D_b and D_a, a reference plane is defined, which is vertical to the patch normal direction and passes the origin point of the coordinate system. That is

$$(\cos\alpha)\,x + (\cos\beta)\,y + (\cos\gamma)\,z = 0 \quad (11)$$

The distances from all points, P_1 to P_N to the reference plane, can be calculated, which are $\mathbf{d}_{b1}$ to $\mathbf{d}_{bN}$, respectively. That is

$$\mathbf{d}_{bi} = \pm\,(p_{ix}\cos\alpha + p_{iy}\cos\beta + p_{iz}\cos\gamma) \quad (12)$$

Thus

$$D_b = |\mathbf{d}_{b\max} - \mathbf{d}_{b\min}| \quad (13)$$

Suppose M points in the middle of the patch are removed, $\mathbf{d}_{a1}$ to $\mathbf{d}_{a(N-M)}$ can be calculated in the same way. Hence,

$$D_a = |\mathbf{d}_{a\max} - \mathbf{d}_{a\min}| \quad (14)$$

For illustration purposes, Fig. 3(*a*) shows a patch and its bounding box. After data reduction (suppose the three points in the middle are removed), the new patch and its bounding box are shown in Fig. 3(*b*).

4. Re-triangulation

When a point or a set of points is to be removed, triangles sharing these points must be removed too. This leaves a "blank" region which must be covered through re-triangulation. The implemented re-triangulation method is described as follows:

1. Form a list **L** of points that are affected by the removal of a point or a set of points. In Fig. 4(*a*), if point P is to be

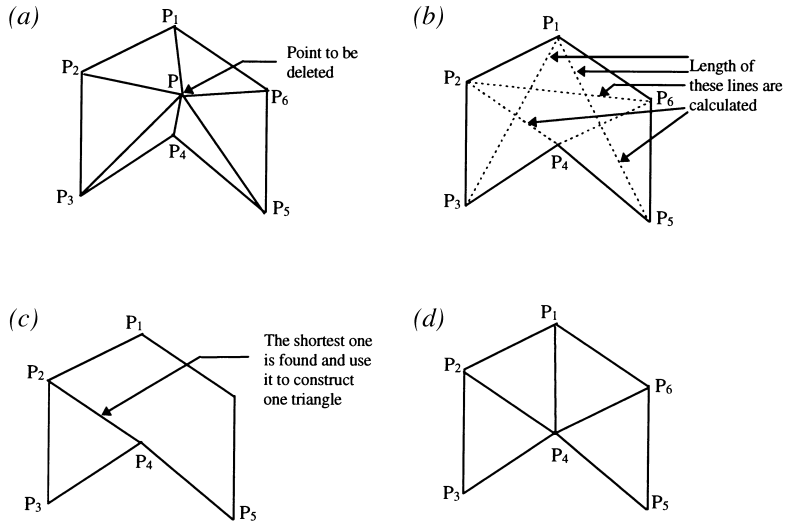

Fig. 4. Re-triangulation. (*a*) Point *P* to be removed. (*b*) Distance calculation. (*c*) Form a new triangle. (*d*) The remaining triangles are formed in the same way.

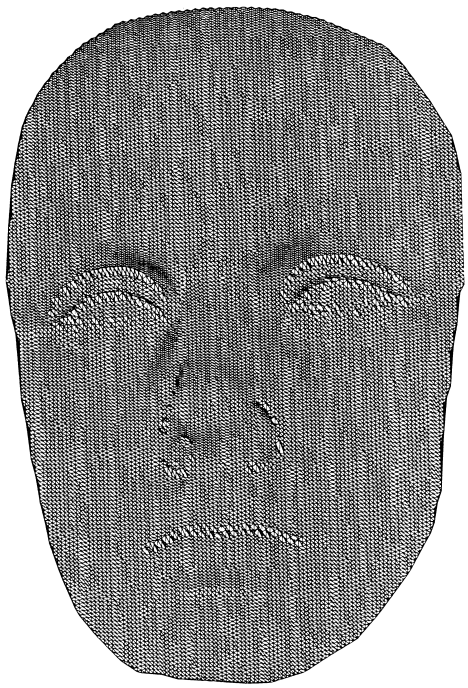

Fig. 5. STL file without data reduction.

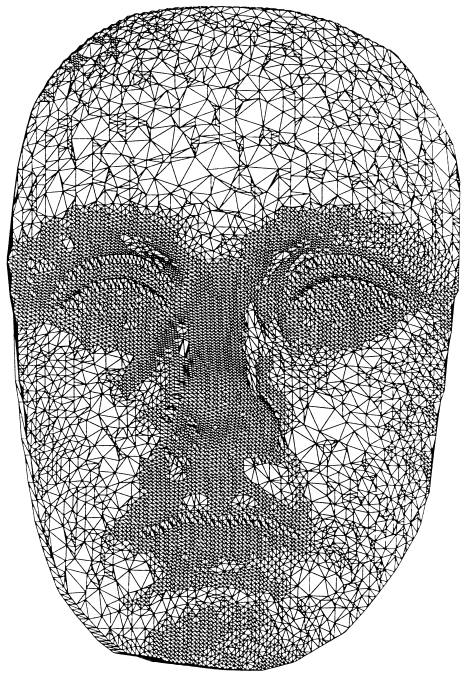

Fig. 6. STL file after 50% data reduction.

removed, the affected point list will be $\mathbf{L} = \{P_1, P_2, P_3, P_4, P_5, P_6\}$.

2. Take three consecutive points from the point list **L** and check whether the triangle formed by the three points is inside or outside the region formed by **L**. If not, take another three consecutive points and check again.
3. Calculate the length of the line formed by the first and the last point of the three points, as shown in Fig. 4(*b*) in dashed lines.
4. Repeat steps 2 and 3 until all combinations of three consecutive points have been exhausted.
5. Find the shortest length and use the corresponding three points to form a real triangle as shown in Fig. 4(*c*). Calculate the normal of the newly formed triangle and delete the middle point of the three points from **L**, forming $\mathbf{L}_1$.
6. Repeat from step 2 with point list $\mathbf{L}_1$ until no more than four points left in $\mathbf{L}_1$.

Fig. 4(*d*) shows a re-triangulated region after a point is removed. Point removal and re-triangulation will go on iterat-

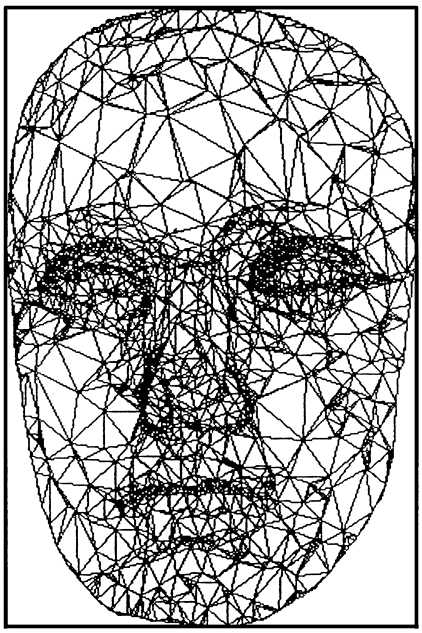

Fig. 7. STL file with a bounded error of 0.5 mm.

ively until certain criteria are met. Two criteria are given for the user to specify. One is the percentage of points to be removed and the other is the bounded error caused by point removal.

5. Experimental Results

With the implemented method, a number of examples have been tested. One of the examples is a human face. It was digitised using a Mitutoyo BLN 122 coordinate measuring machine as shown in Fig. 5. As some parts of the human face have very low curvature change, some data points can be removed. In the experiment, data reduction of 20%, 50% and 80% have been specified for prototype fabrication. Figure 6 shows the STL file after 50% of data reduction. If a bounded error of 0.5 mm is specified, the resulting STL file can be generated as shown in Fig. 7. From Figs 6 and 7, it can be seen that regions with small curvature changes have large data reduction. This is exactly what is desired.

6. Conclusion

This paper has presented a novel method for STL file construction directly from CMM measurement data. When desired, the user can specify either the percentage of points to be removed or the bounded error caused by points removal in order to speed up computation and to reduce storage space. Several prototypes with different amounts of data reduction have been built using SLS equipment which shows satisfactory results.

Acknowledgement

This research is carried out with a CRCG grant (code: 337/064/0023) from the University of Hong Kong.

References

1. X. Yan and P. Gu, “A review of rapid prototyping technologies and systems”, Computer-Aided Design, 28(4), pp. 307–318, 1996.
2. J. L. Miller and K. H. Grote, “Solid freeform manufacturing technologies as an important step in the product development process”, Computers in Industry, 28, pp. 11–16, 1995.
3. M. B. Wall, K. T. Ulrich and W. C. Flowers, “Evaluating prototyping technologies for product design”, Research in Engineering Design, 3, pp. 163–177, 1992.
4. 3D Systems, Stereolithography Interface Specification, USA.
5. L. L. Schumaker, “Triangulations in CAGD”, IEEE Computer Graphics and Applications, pp. 47–52, January, 1993.
6. X. Sheng and B. E. Hirrch, “Triangulation of trimmed surfaces in parametric space”, Computer-Aided Design, 24(8), pp. 437–444, 1992.
7. A. Oxley, “Surface fitting by triangulation”, The Computer Journal, 28(3), pp. 335–339, 1985.
8. H. Park and K. Kim, “An adaptive method for smooth surface approximation to scattered 3D points”, Computer-Aided Design, 27, pp. 929–939, 1995.
9. B. Humann, “A data reduction scheme for triangulated surfaces”, Computer-Aided Geometric Design, 11, pp. 197–214, 1994.
10. A. D. Kalvin and R. H. Taylor, “Superfaces: polygonal mesh simplification with bounded error”, IEEE Computer Graphics and Applications, pp. 64–77, May 1996.