Slicing 3D CAD Model in STL Format and Laser Path Generation

Munir Eragubi

***Abstract*—The program in this paper is an essential component of the Rapid Prototyping Machine Manufacturing Project. The process of Rapid Prototyping is an integrated product manufacturing process without human interference. Product data which has been designed by CAD software is inserted into the program, then send directly to the Rapid Prototyping Machine, which manufactures the final product by converting a slice of the raw material powder to solid material, then bond it to the previous layer automatically. The program reads the data of 3D design, which has been saved in STL format file, this requires a decoder for the codes used to store these type of files, the program then displays the design graphically to insure the form and dimensions of the model, then the process of slicing of this model digitally starts from the bottom of the design and finishes at the highest point; the outer contour line of the slice defines the material of the model, the inner contours defines the cavities within the model, a tool path for the laser beam is generated to covers all the areas that represent the material area, this Tool Path is used to generate instruction codes sent to the laser source which controls the direction of the Laser beam, which converts the raw material powder to a solid final product.**

***Index Terms*—STL file format, rapid prototyping, slicing 3D model, laser path.**

I. Introduction

Rapid Prototyping (RP) refers to fabrication of parts layer-by-layer. It involves adding raw material successively, in layers, to create a solid of a predefined shape. The process is fully automatic and it offers many advantages over traditional manufacturing processes.

The first step to produce the part using RP technology is to put the design of the required part in a numerical data. There is a wide range of sources for 3-D model data input for RP process.

Most of RP systems use a CAD model as the main source for the data input. The CAD model can be a solid model or a surface model [1].

To fabricate any design in current rapid prototyping systems the 3-D model (surface or solid) has to be transferred to STL (standard transform language) [2] format, which is the most common standard interface between CAD and RP systems.

The STL model is mathematically sliced by intersecting it with horizontal planes. Each slice represents a cross-section data for the part. The layer thickness is the distance between these planes.

Manuscript received November 15, 2012; revised February 5, 2013.

Munir Eragubi is with the College of Engineering Technology/ Department of Mech. Eng., Janzor, Libya (e-mail: eragubi@gmail.com).

II. STL File Format

STL format was developed and published in 1987 by 3D systems for converting 3D CAD models for use in stereo lithography and has become the de-facto standard for the data input for all types of RP systems [3] and [4].

The STL file format is generated using a tessellation process, which generates triangles to represent the CAD model, these triangles are described by a set of X, Y and Z coordinates for each of three vertices, and a unit normal vector to indicate which side of the triangle contains the mass. STL file can be in ASCII or in binary format, the ASCII STL format file is larger but legible see Table I [5] and [6].

TABLE II: Example of STL File and Description of ASCII Representation Format

STL File ASCII Format	Description
solid ascii	The start of solid
facet normal 9.914449e-001 -1.305262e-001 0.000000e+000	Identifies the material side
outer loop	The start of the triangle vertex
vertex 1.862222e+001 1.402943e+001 6.000000e+000 vertex 1.875000e+001 1.500000e+001 0.000000e+000 vertex 1.875000e+001 1.500000e+001 6.000000e+000	x, y, z for each vertex
endloop	End of the triangle vertex
endfacet	End of the triangle information
Endsolid	End of the solid information

Fig. 1. Example of many objects in STL format.

STL is a very simple format, yet contains the potential for defining any shape with any number of edges see Fig. 1.

III. Slice the Geometric

Slicing the geometric requires the data from the bottom to the top.

A. Find the Cutting Plan (Z_c)

The program scans STL file, picks the Z-coordinate of all the facets, compare, and find the top, the bottom of the geometric (Z_{min}, Z_{max}), add the layer thickness to Z_{min}.

$$Z_c=Z_{min}+[\text{layer thickness}]$$

DOI: 10.7763/IJIMT.2013.V4.431

B. Find the Facets that Lintersect with the Cutting Plan

The program scans STL file to pick one facet at a time, the Z-coordinate of its three vertices compared with the Z-height of the current plane,

$$Z_{min} \le Z_c \le Z_{max}$$

$Z_{mi}Z_{max}$ are in one triangle

The program ignores the facet that does not intersect with the cutting plane, and proceeds to check the next facets.

C. Find the Lines that Intersect with the Current Plan

When the facet which intersect with the cutting plane is found; the program slices the facet according to the seven possible cases divided into 5 groups, see Fig. 2.

Group 1- all vertices away from the cutting plane

Group 2- one point in the cutting plane with the two remaining vertices in different regions.

Group 3- one point in the cutting plane with the two remaining vertices can be above or below the cutting plane.

Group 4- two vertices lie in the cutting plane the remaining vertex can be above or below the cutting plane.

Group 5-all the three vertices in the cutting plane

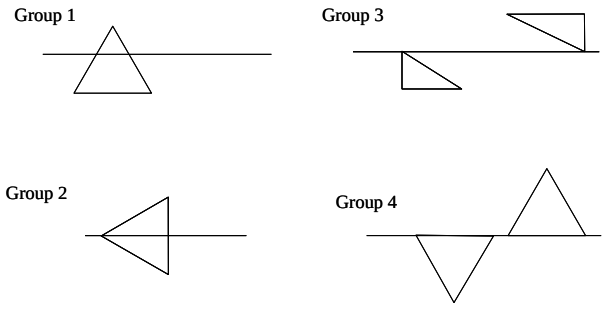


Fig. 2. The possible cases examined for facet-plane slicing.

For each of these triangles, the program has to find which lines intersect with the slicing plane, by checking

$$Z_a \le Z \le Z_b \text{ Or } Z_a \le Z \le Z_c \text{ Or } Z_b \le Z \le Z_c$$

Z_a &Z_b& Z_c of one triangle see Fig. 3.

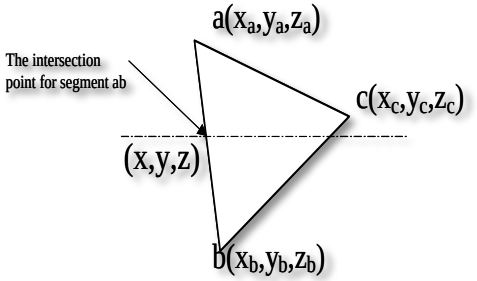


Fig. 3. Three vertices of one triangle and the entersection points.

D. Find X ¯Y Coordinates of The Intersections

For each of these lines, will find (X, Y) coordinate of the points where the edge is intersected by the slicing plane. The general equation to find the points coordinate is:

$$\frac{X - X1}{X2 - X1} = \frac{Y - Y1}{Y2 - Y1} = \frac{Z - Z1}{Z2 - Z1} \quad (1)$$

By connecting the intersection points in each triangle a sets of straight-line will be formed see Fig. 4.

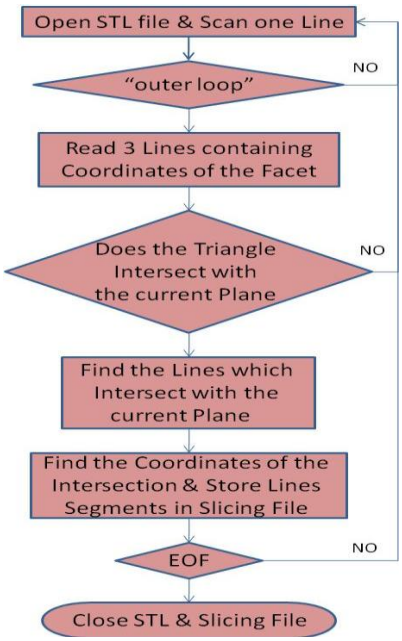


Fig. 4. Slicing algorithm flow chart.

The slicing output is a list of lines -in a random order - forms a closed contour.

E. Contour Construction

Contours are closed polygons that do not intersect with each other. The start and the end vertices of a line are called the head and the tail of the line, respectively. The position of the head for one line is the same poison of the tail for the neighbouring line.

The head-to-tail search connects the lines by cheeking the coordinate of the head of one line with the tail of a neighboring line, see Fig. 5.

This comparison process should produce a correct slice contour. [7]

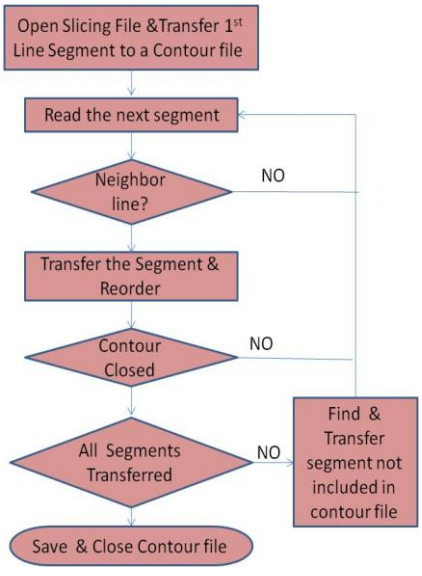


Fig. 5. Contour construction algorithm flow chart.

F. Contour Filling

Contour filling is one of the most common problems in graphics and picture analysis as in rapid prototyping, where the contour is defined in a polygon, and the interior of the

region has to be found.

There are many ways to solve this problem, party check algorithms is a technique used to decide whether a point is in the interior of the polygon [8].

Party check algorithms are based on the fact that a straight line intersects any closed curve -such as the contour of a region- an even number of times, see Fig. 6, by counting the number of intersections, and finding the first intersection, the segment between the first and the second intersection is inside the contour, the segment between the second and the third intersection is outside the contour. When the lines are tangent to the contour the point of contact must be counted twice as intersection [9].

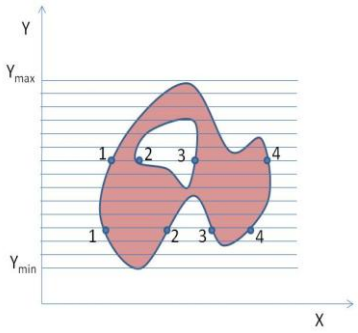


Fig. 6. Contour filling.

The algorithm is not restricted to a simple connected polygon, it will fill any number of regions if their contours have been sorted together, see Fig. 7.

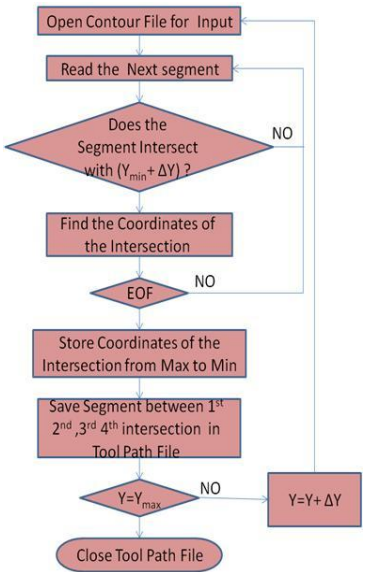


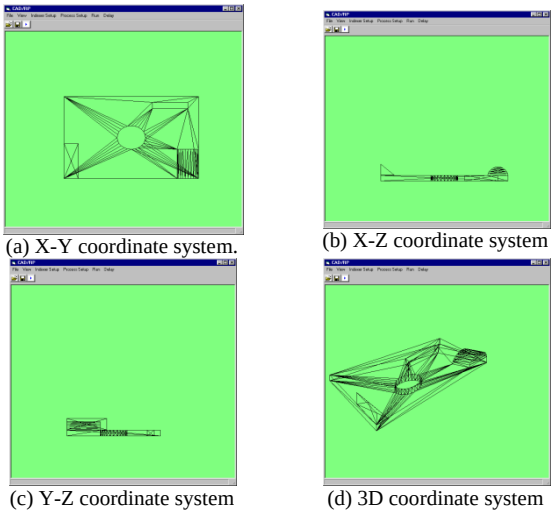
Fig. 7. Flow chart for the contour-filling algorithm

IV. Result And Discussion

The Algorithm proved the capability to process and slice STL files, generate the tool path for RP operation.

A. View Geometry

The algorithm provides capability to view the design in 2D and 3D. This gives the ability to ensure that the correct geometry is imported before starting the slicing process. Algorithm views the model in x-y, x-z, y-z and 3D coordinate systems, see Fig. 8.



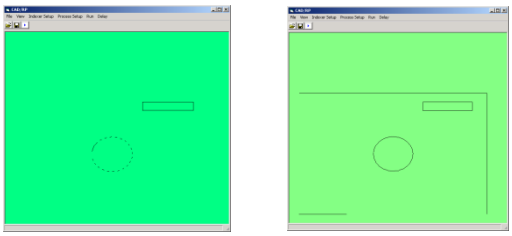
(a) X-Y coordinate system. (b) X-Z coordinate system

(c) Y-Z coordinate system (d) 3D coordinate system

Fig. 8. Various views for the imported geometry.

B. Slicing Process

The algorithm is capable of slicing any STL file with any layer thickness. The result of this operation is a random segment order contour. When a long delay time is specified a random construction of the contour can be observed see Fig. 9.

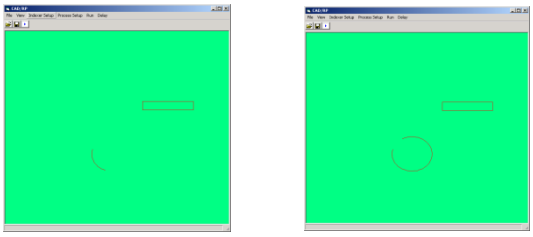


(a) 25% of contour (b) 50% of contour

Fig. 9. Construction of contour as found in STL file.

C. Contour Construction

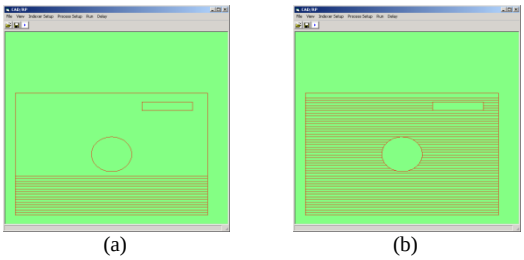
The algorithm orders the contour in this step to construct a continuous tool path. It is very clear to see the importance of this step, especially when circular contour is constructed see Fig. 10.



(a) 20% of contour (b) 50% of contour

Fig. 10. Construction of contour as found in STL file.

D. Filled Pattern



(a) (b)

Fig. 11. Raster scan pattern filling.

The algorithm fills the cross section area with a raster filling style, this can be seen in Fig. 11 where the cavities inside the geometry were defined and only the solid areas were filled.

REFERENCES

[1] V. Kumar and D. Dutta, "An assessment of data formats for layered manufacturing," *Advances in Engineering Software*, vol. 28, no. 3, pp. 151-164, April 1997.

[2] I. Stroud and P. C. Xirouchakis, "STL and extensions," *Advances in Engineering Software*, Elsevier Science, vol. 31, no. 2, pp. 83-95 February 2000.

[3] G. Jacob, C. kai, and T. Mei, "Development of a new rapid prototyping interface," *Computer in Industry*, Elsevier Science, vol. 39, no. 1, pp. 61-70, June 1999.

[4] C. Machover, *The CAD/CAM hand book*, The McGraw-Hill Companies, Inc. 1996.

[5] X. Yan and P. Gu, "A review of rapid prototyping technologies and systems," *Computer-Aided Design*, Elsevier Science, vol. 28, no. 4, pp. 307-318, 1996.

[6] S. H. Choi and K. T. Kwok, "Hierarchiacal slice contours for layered-manufacturing," *Computers in Industry*, Elsevier Science, vol. 48, no. 30 , pp. 219-239, August 2002.

[7] Y. F. Wu, Y. S. Wong, H. T. Loh, and Y. F. Zhang, "Modelling cloud data using an adaptive slicing approach," *Computer-Aided Design, Elsevier Science*, vol. 36, no. 3, pp. 231-240, March 2004.

[8] J. H. Kingston, "Algorithms and data structures," *Addison Wesley Longman Limited*, 1997, ISBN 0-201-40374-9, pp. 221.

[9] M. Eragubi, "Integrating CAD/CAM with rapid prototyping," M. Eng. thesis, Dept. Mechanical. Eng. Dublin City University, Dublin, Ireland, 2004.

Munir Eragubi was born on December 06, 1969, in Tripoli, Libya. He got master degree in "Integrating CAD/CAM with Rapid prototyping", in Dublin City University, Dublin, Ireland, 2004, and Bachelor of Mechanical Engineering, Tripoli University, Tripoli, Libya. His major of study is Mechanical Engineering.

He is a LECTURER in the College of Engineering. Technology/ Department of Mech. Eng., Janzor, Libya.