International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:3, No:1, 2009

An Enhanced Slicing Algorithm Using Nearest Distance Analysis for Layer Manufacturing

M. Vatani, A. R. Rahimi, F. Brazandeh, and A. Sanati nezhad

***Abstract*—**Although the STL (stereo lithography) file format is widely used as a de facto industry standard in the rapid prototyping industry due to its simplicity and ability to tessellation of almost all surfaces, but there are always some defects and shortcoming in their usage, which many of them are difficult to correct manually. In processing the complex models, size of the file and its defects grow extremely, therefore, correcting STL files become difficult. In this paper through optimizing the exiting algorithms, size of the files and memory usage of computers to process them will be reduced. In spite of type and extent of the errors in STL files, the tail-to-head searching method and analysis of the nearest distance between tails and heads techniques were used. As a result STL models sliced rapidly, and fully closed contours produced effectively and errorless.

***Keywords*—**Layer manufacturing, STL files, slicing algorithm, nearest distance analysis.

I. Introduction

RAPID layer-by-layer fabrication methods now play an important role in time reduction and improving flexibility of product development, and also facilitate product visualization and functional testing, which effectively shorten the product development cycle. One of the important areas of research and development that contributes to the success of rapid prototyping, is the interface standard, which is the vital link between the computers aided graphic design (CAGD) modeling and layer fabrication equipment [14].

Several existing interface standards have been used, such as stereo lithography interface specification (STL), initial graphics exchange Specification (IGES) [1], standard for the exchange of product model data (STEP) [2], Hewlett-Packard graphics language (HP/GL) [3], computerized tomography (CT) [4], rapid prototyping interface (RPI) [5, 6], common layer Interface (CLI) [7], layer exchange ASCII Format (LEAF)[8], surface triangles hinted formats (STH) [9], etc.

STL file format because of its simple topology and powerful nature in tessellation of almost all surfaces is widely accepted and supported by most commercial CAGD software and layer fabrication equipment [14]. STL format is composed of only one type of element, a triangular facet, which is defined by its normal and three vertices [14]. All the triangular facets described in a STL format file constitute a triangular mesh to approximate Modeling surfaces. In fact, STL file format is a polyhedral representation of a surface model with triangular facets which facets must obey the vertex to vertex rule and facet orientation rule [10, 11]. Since calculations involved in the generation and slicing of STL triangular facet are easy, fast and accurate enough to satisfy the requirements of the rapid prototyping industry, STL is reasonably suitable to be the interface between object modeling and layer-by layer fabrication. Many CAD systems would generate incorrect STL files that disobey the above two rules when the CAD models are very complex [12, 13]. In this case, flaws may appear in the process of creating triangular facets. There are four types of flaws that may appear in STL files. First type is inconsistency problem such as incorrect normal vector and inconsistent normal vector [14, 15]. Second type is malformation problem, for instance cracks and holes that exist in the STL models are two types of malformation flaws. Third type of flows in STL models is illegal over laps and final type of flaws in STL models is non manifold facets. [15].

Redundant depiction of geometric elements in STL format is another disadvantage of its format, i.e., each vertex of a triangular facet is recorded at least four times brings extra computational memory occupation and time consumption. Another shortcoming is that STL file size is incommensurate with its approximation accuracy. Especially when the required approximation accuracy of an object surfaces increases, or when there is a complex surface, the size of the generated STL file is dramatically enlarged [14]. Flaws whose appear in the process of creating triangular facets can be checked and corrected afterwards [14-12]. Most of current STL file repairing programs can repair only simple defects automatically [15] and flaws such as Complex cracks, and non-manifold facets are difficult for current technology to repair. In addition algorithms [16], which their aims are to reduce the amount of computer memory required for processing of STL file, can't repair complex flows.

As the purpose of RP systems is to produce parts layer by layer from a CAD model, the useful information for

F. M. Vatani is Msc Student of Manufacturing (e-mail: vatani@aut.ac.ir).
S. A. Rahimi is Assistant Professor at manufacturing Department (e-mail: rahimi@aut.ac.ir).
T. F. Brazandeh is Assistant Professor at manufacturing Department (e-mail: f.brazandeh@aut.ac.ir).
S. A. Sanati Nezhad is Msc Student of Manufacturing (e-mail: amirsanatinezhad@gmail.com).

manufacturing is correct contours of the sliced layers of the CAD model instead of the STL file itself. In this paper a complete algorithm is proposed which slices the bad STL model directly in spite of amount and type of the flows in it. It can also repairs the slicing contours effectively at a 2D level to obtain the fully closed and errorless contours.

II. Optimized Slicing Algorithm

In conventional slicing algorithms all the facets store into the computer memory (Fig.1a), which limit the size and complexity of design and thus lead to breakdown of the LM process [16]. In addition in most slicing algorithms to reduce memory usage, only one adjacent facet reference is stored for each edge when building the topological structure. If an STL file contains non-manifold facets, which contain any over-adjacent edges, some adjacency information for non-manifold facets will be lost [15]. To overcome the shortcomings of these algorithms, as shown in Figure 1b, the proposed optimized slicing algorithm store all facets that intersect with cutting plane in computer memory. Then subsequent analysis applies only to these facets, which cause alleviation of memory and also prevent deletion of adjacency information for Non-manifold facets.

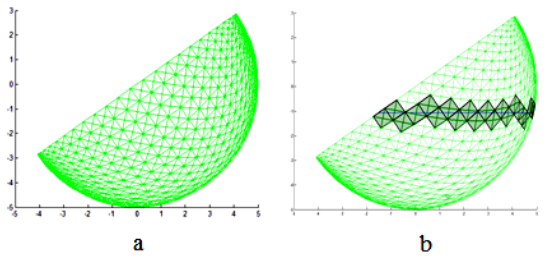

Fig. 1 Proposed algorithm, reads facets which intersect with cutting plane and then processes them instead of process all facets

III. Contours Line Generation

For Implementation of enhanced algorithms the ASCII format of STL file is used because it is simple, readable and has ability to show the advantage of enhanced algorithm. The structure of text STL format is shown in Fig. 2a. In the first step vertex coordinates of each facet, as shown in Figure 2b, store to an n×3D array. In order to save memory and alleviate the subsequent analysis processing, the data of unit normal vectors were ignored while storing them in the Facet Vertex Data Matrix as shown in Fig. 2b. Therefore the size of data to be treated will be reduced.

To facilitate subsequent contours construction, the slicing module calculates the intersection line segment, instead of point coordinates of the facets with cutting plane, then as shown in Fig. 2c, calculated segment lines store in an m×6D array. This procedure eases sorting and finding the direction of contours. Also correction of contours errors with combining the tail to head search [16] and the nearest distance analysis techniques will be more convenient.

The slicing steps can summarize to extract facets data, comparing them with cutting plane, and if there is any intersection, store the intersection lines to an m×6D array. The flowchart of this module is shown in Fig. 3.

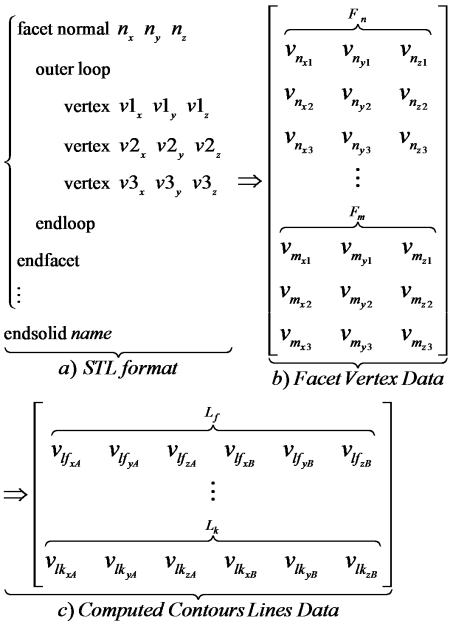

Fig. 2 Data structure for facets and Computed contours lines

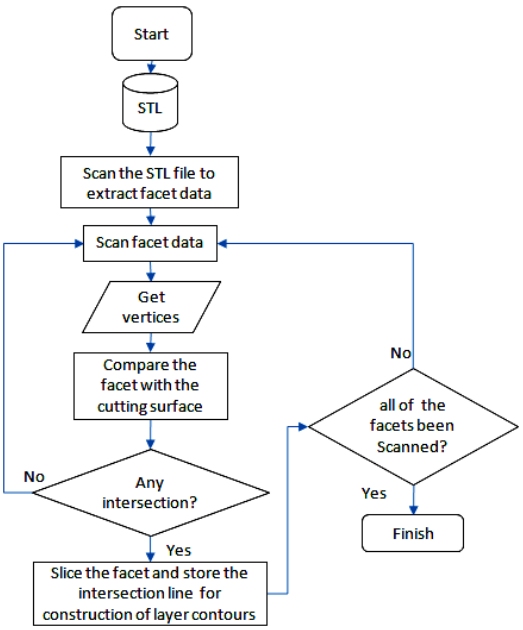

Fig. 3 Slicing Algorithm

The possibilities of intersection of a facet with a cutting plane can be categorized into the following four groups, as shown in Figure 4. The first group is shown as case 1 in Figure 4, which is the normal case where a facet intersects with the cutting plane with all the vertices away from cutting plane. The intersection results of cutting plan with facet edges

will be obtained from the equation 1.

$$\frac{x-x_a}{x_b-x_a}=\frac{y-y_a}{y_b-y_a}=\frac{z-z_a}{z_b-z_a} \quad (1)$$

In this equation (x_a, y_a, z_a), (x_b, y_b, z_b) are end points of facet edge that must satisfy the equation 2.

$$z_{v1} \le z_{cutting\ plane} \le z_{v2} \quad (2)$$

In this equation Z_v is z-height of facet vertex.

The second group will satisfy the conditions of equation 2, but the intersection results are three points where two points are equal and one of them store in the line matrix.

Third group occurs if one vertex lies in the cutting plane and remaining vertices lie in same region as shown in Fig. 4 as case 3. Zhang has ignored this case and has obtained line segment from neighboring facets [15]. Since there is no acquaintance about validity of neighboring facets; the intersection results, which is two equal points, must store in line matrix. Therefore the efficiency of proposed algorithm to create valid contours will be enhanced.

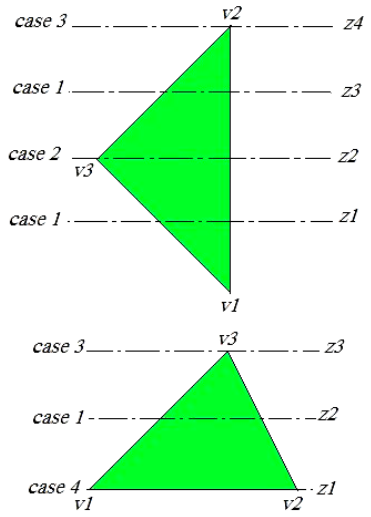

Fig. 4 Probability of intersection of a facet with a cutting plane

Fourth group occurs if two vertices lie in the cutting plane. In this case equation 1 is not satisfied and the result of intersection is the end point of the line that could be obtained from two other edges.

IV. Errorless Contour Construction with Nearest Distance Searching

The scope of errorless contour is generating fully closed contour that all contour line segments are in one direction (one direction means head of one line lays on tail of previous line) and store after each other consequently. Normally, there are three types of error in generated line segment contours. First, line segments are stored in line matrix randomly [15, 16]. Second, due to inconsistency problem in the STL file, line segments may be generated in different directions. Last error is caused by the malformation problem or rounding-off error [15], where the head and tail of neighboring lines may not lie on each other. The proposed algorithm tries to solve these errors automatically.

V. Implementation of Contour Construction Algorithm

In calculated line matrix there may exist equivalent lines in opposite direction because of incorrect and inconsistent normal vector, or equivalent lines in same direction, though one of equivalent lines must be ignored as shown in Fig. 5.

$$\begin{bmatrix} x_A & y_A & z_A & x_B & y_B & z_B \\ x_B & y_B & z_B & x_A & y_A & z_A \\ x_{A'} & y_{A'} & z_{A'} & x_{B'} & y_{B'} & z_{B'} \\ x_{A'} & y_{A'} & z_{A'} & x_{B'} & y_{B'} & z_{B'} \end{bmatrix} \Rightarrow \begin{bmatrix} x_A & y_A & z_A & x_B & y_B & z_B \\ x_{A'} & y_{A'} & z_{A'} & x_{B'} & y_{B'} & z_{B'} \end{bmatrix}$$

Fig. 5 Elimination of equivalent lines which are 1-in same direction and 2- in opposite direction

VI. Nearest Distance Analysis

In first stage, line matrix must be sorted, in order to sort line matrix, proposed algorithm selects the tail of first line and looks for nearest point (NP) of other lines, regardless of being tail or head, which selects the related line (RL) to that line. The proposed algorithm uses equation 3.

$$D=\sqrt{(x-x_t)^2+(y-y_t)^2} \quad (3)$$

Here x_t, y_t are coordinate of tail of line. If NP is related to head of RL, then this line stores in new line matrix after previous line. But if NP is related to tail of RL, then algorithm changes the situation of tail with head, then new line stores in new line matrix after previous line. This operation will be done until all lines are sorted and stored in new line matrix. The flowchart of sorting module is shown in Figure 6.

After sorting all lines, algorithm must looks for third groups of line which are generated by intersection of cutting plane with facets. In the third group of generated lines, heads and tails are laying on each other. If they lay on head of subsequent line and tail of previous line (state in which there is no flaws in surrounding area), it will be ignored from line matrix. If they lay either on head of subsequent line, or tail of previous line (state in which there is flaws in surrounding area) it will be ignored too, as shown in Figure 7. So with this operation all topological and geometrical errors will be corrected. Where there is no match between this line and tail and head of previous and subsequent lines, and the contour is open, it will be stored for further processing.

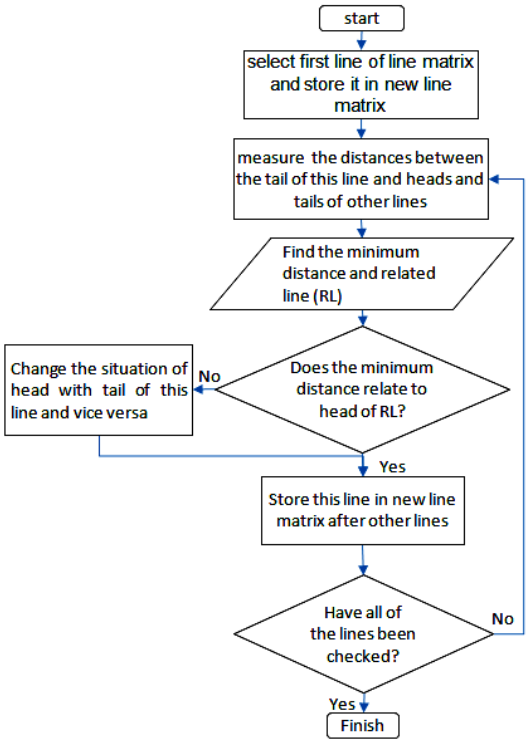

Fig. 6 Arranging and sorting algorithm

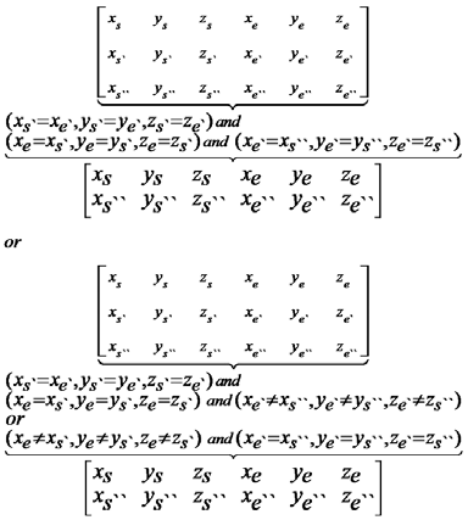

Fig. 7 Elimination of topological and geometrical errors from line matrix

Final stage of proposed algorithm is constructing fully closed contours from line segments. In this stage algorithm looks for a line where its end is open (OL), and then goes through the following operation:

1-Measure distance between the tail of OL and heads of subsequent line, then gets minimum distance (D1) and selects related line (RL) to it.

2- Measure distance between the tail of OL and head of starting line of contour (D2). Starting line of a contour is a line which contour begins from and ends to it.

If D1 is less than D2 then algorithm creates a line between tail of open line and head of related line. But if D2 is less than D1 then algorithm creates a line between tail of open line and head of starting line, then the next line after open line is considered as the starting line for next contour. The flowchart of this module is shown in Fig. 9.

In order to have a correct closed contour not overlaying on itself, the algorithm checks the created line does not lie on previous line, and also does not pass through previous lines.

VII. Verification of Proposed Slicing Algorithm

The enhanced slicing algorithm is implemented with Matlab 7.4.0. The program has been tested extensively with STL files with various defects reported in earlier researches and found that output contour layer is the correct slices of the model.

In order to find maximum ability of proposed algorithm on crack and non-manifold edge an STL, model of assembled fans is chosen. It contain of 4 parts as shown in Fig. 8.

It contains many minute features for testing the stability of the proposed slicing algorithm. There are 119746 facets in ASCII format. There are 2 non-manifold edge and 1264 non manifold vertices and a lot of free edge and cracks in STL file. The other specifications of this model are list in Table I.

TABLE I
STL Model Specification

Name	Fan assembly
Nb. cells	2
Nb. points	89202
Nb. activated points	89202
Nb. Triangles	119746
Nb. boundaries	293
Min extremity	-80.398 mm
	166.869 mm
	-19 mm
Max extremity	53.916 mm
	78.964 mm
	19 mm
Dimensions	134.314 mm *
	245.832 mm *
	38 mm
	(0.001 m3)

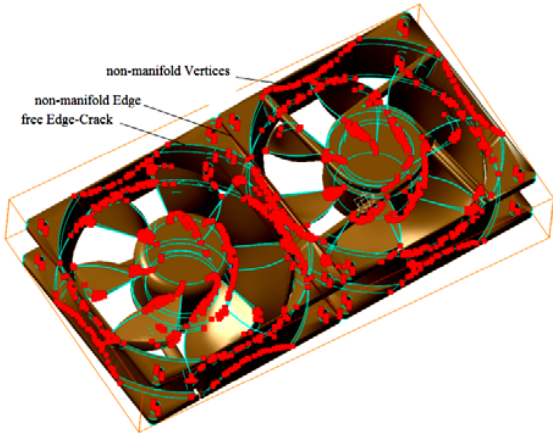

Fig. 8 Bad STL model

Fig. 10 shows the output slice of the model. It was hatched to show the features more obvious.

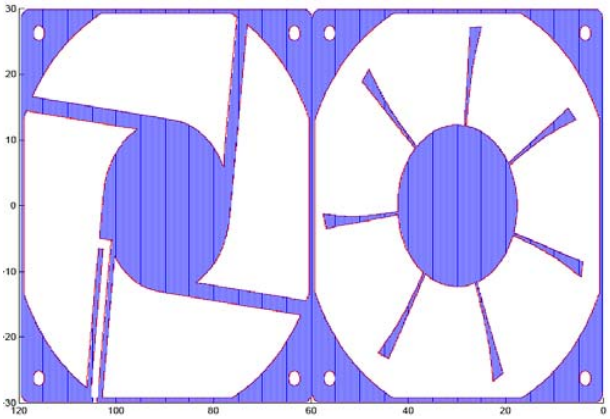

Fig. 10 The slice Layer which have created with proposed algorithm from bad STL model

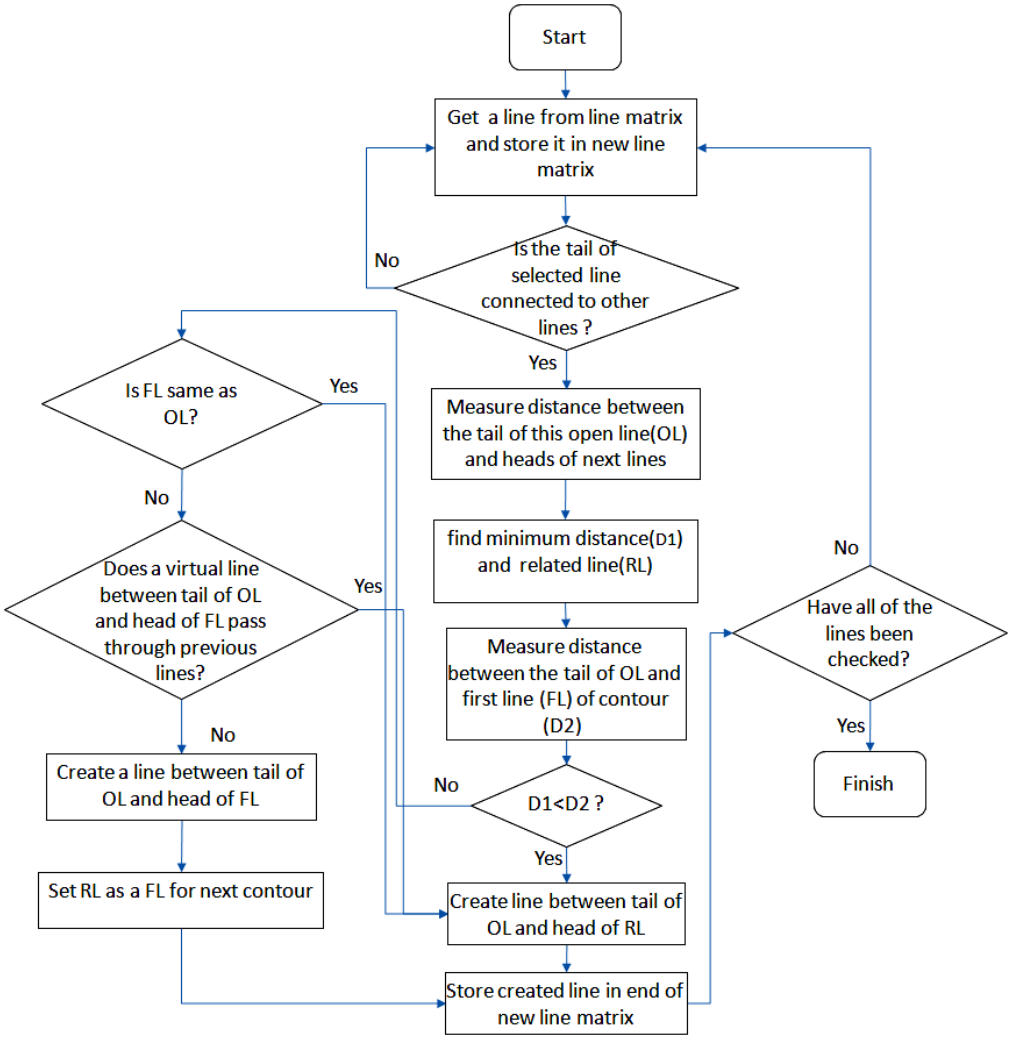

Fig. 9 Closing contours algorithm

VIII. Conclusion

Layer Manufacturing (LM) processes produce models layer by layer. Hence, the models must be first sliced into layers. Therefore, the slicing algorithm plays a very important role in LM system.

The proposed slicing algorithm has very good ability for processing and slicing almost every STL file, regardless of the complexity of models and type of defects. Therefore, no further efforts would be required to correct the bad STL files, which are usually very time consuming, if conventional slicing algorithm is used. In addition, due to ignoring the normal vectors, the size of STL file for processing are reduced and the constraints of computer memory and computational instability are overcome. As a result, the proposed algorithm is able to produce quick accurate closed contour.

References

[1] Reed K, Harvd D, Conroy W "Initial graphics exchange specification (IGES) version 5.0". CAD-CAM Data Exchange Technical Center, Fairfax, VA,1990
[2] Owen J "STEP: an introduction. Information Geometers", Winchester, UK,1990
[3] Hewlett-Packard Company Staff " The HP-GL/2 reference guide: a handbook for program developers", Addison- Wesley, Reading, MA,1990
[4] Swaelens B, Kruth JP "Medical application of rapid prototyping techniques". The 4th international conference on rapid prototyping, Dayton, Ohio, pp 107–120, 14–17 June 1993
[5] Rock SJ, Wozny MJ "A flexible file format for solid freeform fabrication". In: Marcus HL et al (eds) Proceedings of solid freeform fabrication symposium, The University of Texas at Austin , pp 1–12, 12–14 August 1991
[6] Wozny MJ "Systems issues in solid freeform fabrication". In: Marcus HL et al (eds) Proceedings of solid freeform fabrication symposium, The University of Texas at Austin, pp 1–15, 12–14 August 1992
[7] BRITE/Euram "Common layer interface CLI version 1.31". Brite Euram project BE2578 RPT—development and integration of rapid prototyping techniques for the automotive industry, Brite/Euram Industrial and Materials Technologies. 1994
[8] Dolenc A, Malela I "Leaf: a data exchange format for LMT processes". The 3rd international conference on rapid prototyping, Dayton, Ohio, pp 4–12, 1992
[9] Burns M Automated fabrication. Prentice-Hall, Englewood Cliffs, NJ, 1992
[10] 3D Systems Inc, Stereolithography Interface Specification, 1988.
[11] Y. H. Chen, C. T. Ng and Y. Z. Wang, “Data reduction in integrated reverse engineering and rapid prototyping”, International Journal Computer Integrated Manufacturing, 12(2), pp. 97–103, February 1999.
[12] K. F. Leong. C. K. Chua and Y. M. Ng, “A study of stereolithography files errors and repair. Part 1: Generic solution”, International Journal of Advanced Manufacturing Technology, 12, pp. 407– 414, 1996.
[13] K. F. Leong. C. K. Chua and Y. M. Ng, “A study of stereolithography files errors and repair. Part 2: Special cases”, International Journal of Advanced Manufacturing Technology, 12, pp. 415–422, 1996.
[14] Tong 7Wu. Edmund H. M. Cheung,’’ Enhanced STL’’, International Journal of Advanced Manufacturing Technology,29,pp 1143-1150,2006
[15] L.-C. Zhang, M. Han and S.-H. Huang,” An Effective Error-Tolerance Slicing Algorithm for STL Files”, International Journal of Advanced Manufacturing Technology , 20,pp 363–367,2002
[16] S.H. Choi and K.T. Kwok ,” A tolerant slicing algorithm for layered manufacturing”, Rapid Prototyping Journal, Volume 8 • Number 3 • pp. 161–179,2002