Solid Freeform Fabrication 2017: Proceedings of the 28th Annual International Solid Freeform Fabrication Symposium – An Additive Manufacturing Conference
Reviewed Paper

STUDY ON STL-BASED SLICING PROCESS FOR 3D PRINTING

Jing Hu*

*Department of Mathematical and Statistical Sciences, University of Colorado Denver, Denver, CO 80204

Abstract

This paper presents a framework about layer contour reconstruction algorithms by STL-based slicing process for 3D printing. The experimental results by the traditional uniform slicing show the contour outline of each layer and comparison among the different slicing thickness of the cutting z-plane. We then proposed a simple but effective adaptive slicing method to work on the complicated model. Moreover, we discuss the future work to further study on addressing the accuracy of the boundaries with the adjustment of slicing thickness.
Keywords: 3D printing, STL, cutting plane, slicing thickness, contour, uniform, adaptive

1. Introduction

3D printing, becoming increasingly popular during the past few years, is used for a huge field of application. It is also known as additive manufacturing(AM), rapid prototyping(RP) and layered manufacturing(LM) by constructing a 3D object in a layer by layer fashion. Comparing to traditional manufacturing technologies, it can build objects with complex individualized features with little extra effort.

The 3D models must be first sliced into layers since the 3D printer prints out model layer by layer. The slicing algorithm plays a very important role in the 3D printing process. The most common technique for slicing is the produce contour data from STL files. An STL file approximates the surfaces of an object with many small planar triangular facets, a process called tessellation. STL file is easy to manipulate within the process-planning system[1].

Here, in section2, firstly we overview the STL file format including the calculation of coordinates of vertex
and normal vector in the triangle facet. Besides, we discuss briefly about the existing slicing algorithms and methods in the current research literature. Our experimental part is in section3. First the uniform slicing is used to get the data for all the layers and then compare the running time by the different interval of cutting z-plane. Besides, we investigate the output by our adaptive method for a human head model. Our future work will focus on the accuracy of locating the boundaries to change the slicing thickness during the process.

2. Previous work and background

2.1 STL files

The input of slicing is a standard STL, which is most commonly used to represent CAD models in 3D printing process planning due to its simplicity and ability to tessellation of almost all surfaces.

STL file with the surface triangulation includes a finite set of triangles satisfying the following conditions: (i) each edge is shared by at most two triangles. (ii) a vertex shared by any number of triangles. (iii) Connectivity: each triangle has at least one point in common with another triangle. (iv) knot-to-knot property: if a vertex shared by a second triangle, then it is also a vertex of the second triangle. (v) no piercing no overlapping: no triangle has an intersection with the interior of any other triangles[2].

An STL file consists of a list of triangle facet data which define the surface of a 3-dimensional object. Each facet is uniquely identified by a unit normal (i.e., a line perpendicular to the triangle and with a length of 1.0) and by three vertices (corners). The normal and each vertex are specified by three coordinates each, so there is a total of 12 numbers stored for each facet. Each facet is part of the boundary between the interior and the exterior of the object. STL file could be in ASCII or binary format, the table below is the structure of ASCII file format.

Orientation of a facet, shown in Fig.1, is determined by the direction of the unit normal and the order in which the vertices are listed, where p1, p2, p3 are three vertices of the triangle. First, the direction of the normal is outward. Second, the vertices are listed in counterclockwise order when looking at the object from the outside (right-hand rule).

Table1: The STL in ASCII file format

Solid
Facet normal (first facet) *Nx Ny Nz*
Outer loop
Vertex1 *V1x V1y V1z*
Vertex2 *V2x V2y V2z*
Vertex3 *V3x V3y V3z*
End loop
End facet
Facet normal (second facet)
Outer loop
.....
.....
End loop
End facet
.....
End Solid

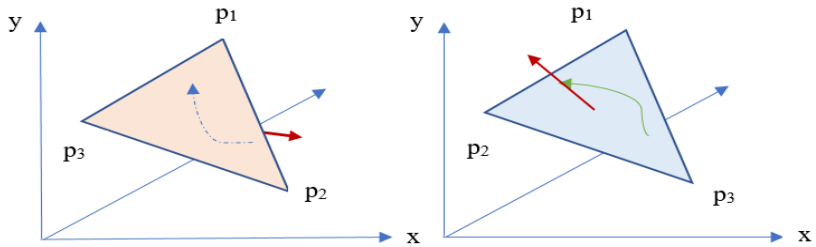


Fig.1: Direction of the face normal based on the right-hand rule

For each triangle, a surface normal can be calculated as the vector cross-product of two (non-parallel) edges of the polygon, see Eqn.(1) and (2).

If $V = p_2 - p_1$ and $W = p_3 - p_1$, and N is the surface normal, then:

$$\begin{cases} N_x = (V_y \times W_z) - (V_z \times W_y) \\ N_y = (V_z \times W_x) - (V_x \times W_z) \\ N_z = (V_x \times W_y) - (V_y \times W_x) \end{cases} \quad (1)$$

If A is the new vector whose components add up to 1, then,

$$\begin{cases} A_x = N_x/(|N_x| + |N_y| + |N_z|) \\ A_y = N_y/(|N_x| + |N_y| + |N_z|) \\ A_z = N_z/(|N_x| + |N_y| + |N_z|) \end{cases} \quad (2)$$

2.2 Slicing methods

The Fig.2 below shows the software pipeline between input STL file to output G-code file, i.e., convert a 3D model into printing instructions for the 3D printer. It cuts the model into horizontal layers, generates toolpaths to fill them and calculates the amount of material to be extruded.

In current 3D printing practice, the most common technique for slicing is to produce contour data from STL files. The STL model is then sliced by intersecting it with horizontal slicing planes, each of which gives piecewise linear contours of a slice.

According to the source data, there are two kinds of methods to slice the geometric model of a part into layers, i.e., the STL-based slicing and direct slicing based on different 3D CAD systems with different data formats[3].

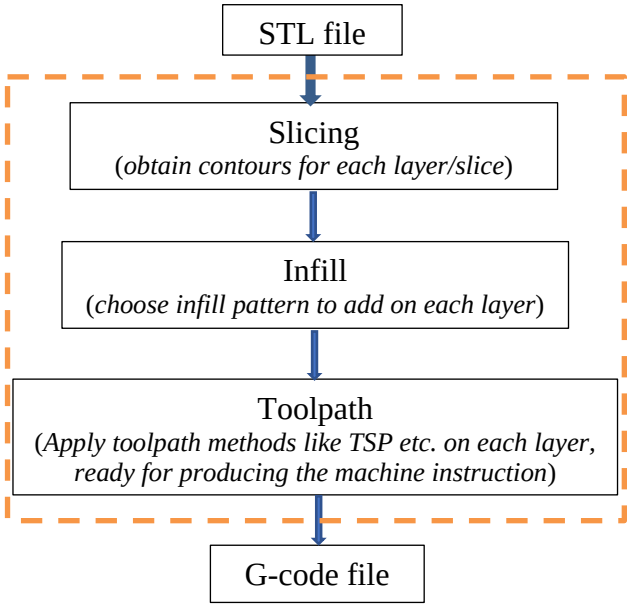


Fig.2. Software pipeline from input STL to output G-code

I. STL-based Slicing

Uniform slicing generates constant layer thickness slices. Adaptive slicing is a variant of the uniform slicing, where the spacing between the slices is not constant but determined by the geometry and machine capability[4]. Adaptive slicing mainly addresses the geometry issues and need a specific 3D printing system to achieve the desired results. However, there are still not 3D printers that fully support or able to take full advantages of adaptive slicing[3]. Therefore, uniform slicing seems more versatile.

Various slicing approaches in the current literature are discussed to aim in overcoming the constraints of computer memory and the problems of computation instability.

S.H.Choi et al. presented a memory efficient tolerant-slicing algorithm[5][6][10]. Instead of storing the whole model into the computer memory, it reads only the facets of the current layer, hence greatly reduces the amount of computer memory required and involves less computationally intensive searching operations. H.Ye et al. proposed a slicing process by reusing the topology information obtained from the template model for other customized products of the same category[8]. D.Ding et.al proposed the multi-direction slicing[7].

II. Direct Slicing

Although STL file format is widely used as a de facto industry standard in the 3D printing 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[9]. Direct slicing can generate precise slice contours from original 3D models and obviates the error-detection and repairing process of STL files.

However, a severe disadvantage of direct slicing is the capability among various 3D CAD systems. In other words, it can only be used for a specific set of software and machine, and is not applicable to any other 3D CAD combinations. As a matter of fact, STL-based slicing is still the commonly used method in processing the problem of layered 3D printing[4][7].

2.3 Slicing Procedure

The STL-based slicing problem is reduced to finding plane-plane intersections. The possibilities of intersection of a triangle facet with a cutting plane can be categorized into the five cases, as shown in Figure3. (a) one vertex of facet on the cutting plane and the other two on the same side, i.e., none of edges on the plane. (b) Two vertices of facet in the plane, i.e., one edge on the plane. (c) One vertex in the plane, one above and one under, i.e., plane cutting through one vertex and its opposite edge. (d) The triangle facet on the plane, i.e., three vertices all in the plane. (e) One vertex on one side and the other two on the other side, i.e., plane cutting through two adjacent edges[8].

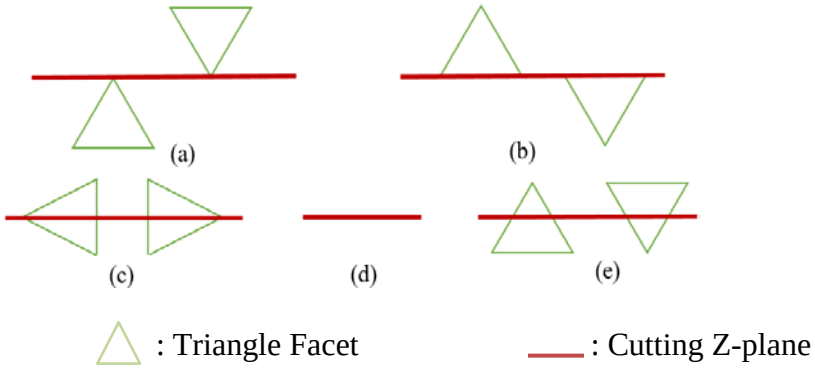


Fig.3 The possible cases of facet-plane slicing

The contours are closed polygons that do not intersect with each other. the scope of errorless contour is generating fully closed contour that all contour line segments are in one direction (i.e., one direction means head of one line lays on tail of previous line.) and store after each other consequently.

The calculation of end point coordinates for segment is straightforward, we traverse all triangle facets, and for each triangle facet we evaluate whether or not it intersects with current slicing z-plane Z_i. This can be done by simply checking the minimal vertical coordinate Z_{min} and maximal vertical coordinate Z_{max} of three vertices from the triangle.

$$\text{Slicing plane } Z_i = Z_{min} + [slice\ thickness]$$

Where the slice thickness denotes the spacing between two slices.
If $Z_{min} < Z_i$ and $Z_{max} > Z_i$, the triangle will intersect with current slicing plane [9]. Then each end point can be calculated by using Eqn.(3),

$$\begin{cases} X = X_1 + \frac{(Z_i - Z_1)(X_2 - X_1)}{Z_2 - Z_1} \\ Y = Y_1 + \frac{(Z_i - Z_1)(Y_2 - Y_1)}{Z_2 - Z_1} \\ Z = Z_i \end{cases} \qquad (3)$$

Where $V_1(X_1, Y_1, Z_1)$ and $V_2(X_2, Y_2, Z_2)$ are two end points of the intersected edge from the triangle.

3. Implementation

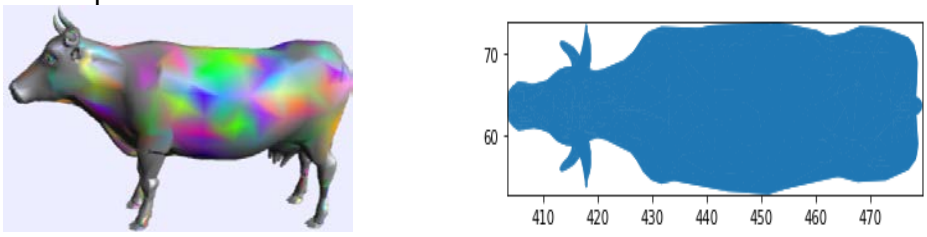
In our paper, our work is built on the assumption that STL file as input is correct without missing edges or vertices, hence it is not necessary to apply the direct slicing to get better outline information in every layer. Meanwhile, using uniform slicing eliminates most compatibility issues among different 3D printing processes since all 3D printers support uniform slicing. Therefore, the uniform approach is firstly applied in our simulation to reconstruct the contours for each layer.

The algorithm intersects all triangles with each z-plane, and then connects the resulting line segments by simple head-to-tail rule into closed polygons for each slice at one time. We could display the cow mesh surface from all the directions in our experiment, see Fig.4.

The first step after loading the input file (cow.stl: www.thingiverse.com) is the scaling, then we get

$x_{max} = 76.3260009765625$ *and* $x_{min} = 0.05$
$y_{max} = 21.202000427246094$ *and* $y_{min} = 0.05$
$z_{max} = 45.7639892578125$ *and* $z_{min} = 0.05$

We have 9964 triangle facets, 4950 vertices and 9895 normal in total. The slicing direction is from the bottom to the top.



(a) (b)

Fig.4 (a) one of all directions in the triangular mesh surface.
(b) The 2D image from the view of top

Set 0.1mm as the slice thickness. The outline for layer #39 and #159 shown in Fig.5, where the left column (a) and (c) are the contours by simply line up all the pairs of points (i.e., line segments), the right ones (b) and (d) are all the points of each layer. The results show the density of points representing the amount of triangle facets intersected by the cutting plane.

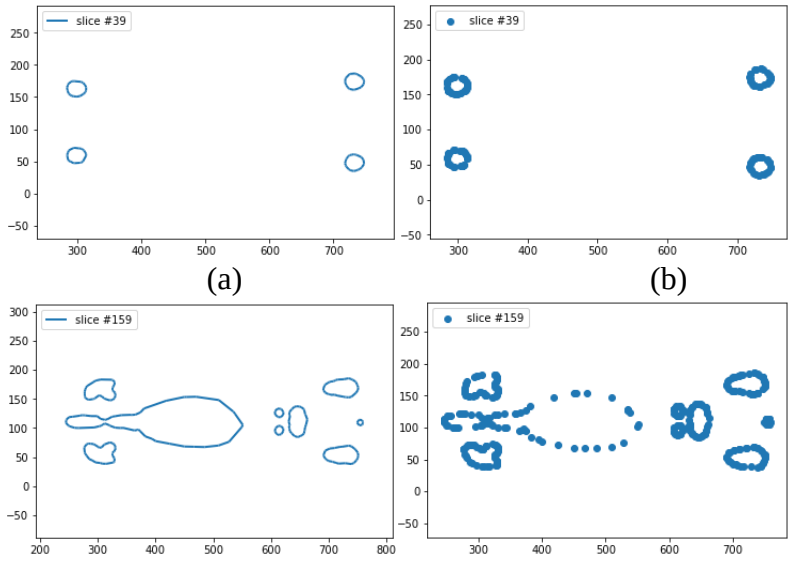


Fig.5 The results for layer #39 and #159

Here we change the interval of cutting planes, e.g., set the slice thickness is 1mm, 0.2mm and 0.3mm respectively. The Table2 shows the total slice number and running time for each case. As the slicing thickness goes bigger, the layer amount is less, the same for the running time.

Table2: comparison of running time for different slicing thickness.

Slicing thickness	Total layer number	Total running time
0.1mm	457	10.5789999962s
0.2mm	229	5.95200014114s
0.3mm	153	4.34500002861s

As shown in Fig.6, orange curve is for the slice spacing 0.3mm, green one for 0.2mm, blue is 0.1mm. We could see the three curves shapes are similar no matter what the spacing value is, however in each change, the lower slicing interval is, the more details is displayed. Therefore, the big interval distance by outputting less layers will result in the loss of the reconstruction accuracy of the 3D object, though the running time cost is decreased.

The STL model is the commonly format adopted by most commercial software for 3D printer input. The more triangles that are used in the representation of the model, the more accurate is the approximation. However, additional facets incur the expense of longer processing times and increased file sizes. Here our implement by the basic uniform slicing had no sign of time-consuming since we only used a simple structure STL file in the experiment. Besides the simulation was run on a PC with 7th generation core i7, thus the slicing speed is pretty fast. For bigger 3D models, this basic uniform slicing approach could fail to obtain the layer information within the tolerating time. Our next simulation experiments involve the more complicated model, here we use a human head one as the input STL file.

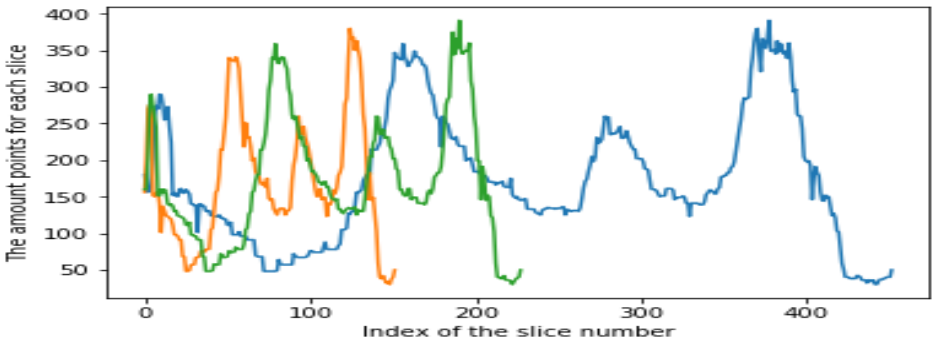


Fig.6. comparison graphs for different slicing thickness

Large thickness of slicing generally incorporates rough texture quality due to the staircase effect. Alternatively, utilizing very small thickness of layers makes the process expensive and time consuming. These two contradiction have led to the development of adaptive slicing technique, in which slice thickness varies according to the surface topology and machine constraints. Most of the available adaptive slicing techniques are based on local surface topology rather than considering the full parts.[15]

Currently there are all kinds of adaptive slicing methods in the research literature, e.g., see references [13]-[16] etc. Here we proposed a simple but effective approach on the complicated models by introducing the concept of "wavelet" in signal processing field. We draw some horizontal lines as the cutting z-planes with two different spacing on the head model, where the much smaller interval on the area with eyes, ears and nose etc. aims at obtaining more structure details, shown in Fig.7. The head model is from the source: (www.thingiverse.com). We plot the head mesh and draw those horizontal lines representing the cutting z-plane.

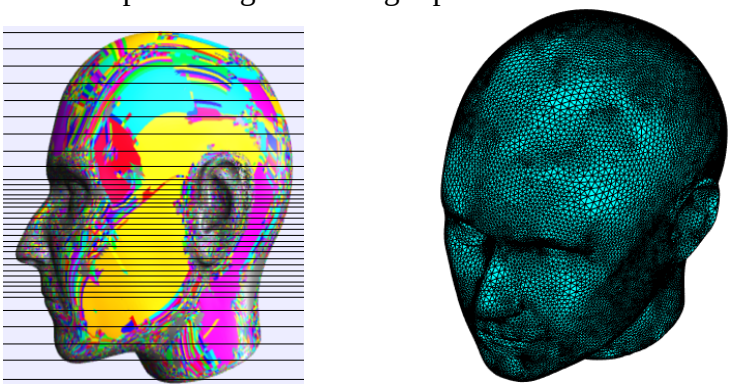


Fig.7. (a) variable slicing thickness (b) the triangle mesh of head model

Table3: comparison of running time for different slicing thickness.

Slice thickness (mm)	Total amount of layers	Running time (seconds)
0.1	2256	≈739.34
0.5	452	≈163.0
1	226	≈90.01
2	113	≈57.03

The general information in the input file is facets: 189450, vertices: 94727, normal: 180971, and the xyz coordinate value after scaling is as follows:

$$\begin{cases} 0.05 \le X \le 172.39577941894531 \\ 0.05 \le Y \le 226.15667419433594 \\ 0.05 \le Z \le 225.6246078491211 \end{cases}$$

In the beginning, we run the basic uniform slicing for the different thickness of slicing, shown in table3. Our goal is to reduce the running time without losing the geometry information details around the area with nose and eyes etc. We then plot all the contours of each layer as the slice thickness increases, see Fig.8.

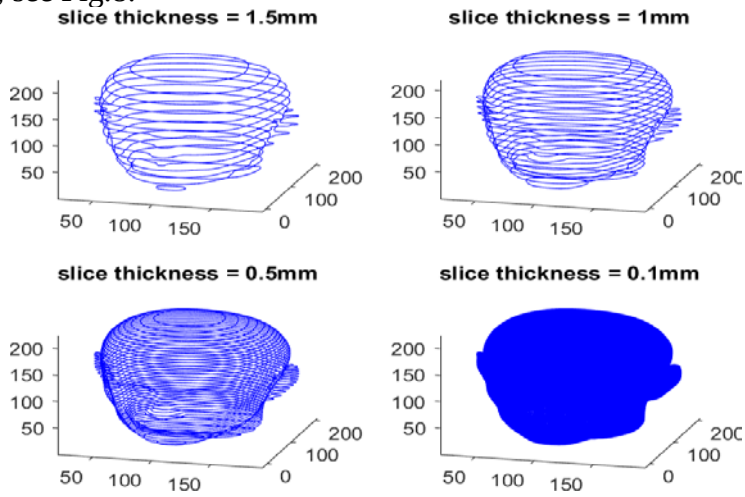


Fig. 8: 3D contour images for different slice thickness

Now, we start the procedure of our new method in the following steps:

Step1: let slicing thickness be 1mm, we got 226 layers.

Step2: based on the graph of points amount in each layer, we consider the interval of layer index [75, 135] since the layers among this interval have all over 1000 points (layer#75: 1048 points, #135: 1148 points), which means more triangle facets involved in each layer, and indicates the more complicated geometry surface details, that is the region of interest (ROI). The selected layer index interval [75, 135] is related to the head part $75\text{mm} \le Z \le 135\text{mm}$.

Step3: the window with the new thickness is set as 0.5mm works on [75mm, 135mm] of the head locally , totally result in 120 layers. According to the curve showing the amount of points per layer, we pick up ROI, i.e., the interval [82, 98] in the layer index graph. The list below is the set of points number of each layer between layer #82 to #98.

[1640,1636,1617,1612,1614,1632,1636,1626,1622,1622,1646,1620,1639,1647,1649,1720,1634]

Which implies the Z-coordinate interval $116\text{mm} \le Z \le 124\text{mm}$ inside the first interval $75\text{mm} \le Z \le 135\text{mm}$, the calculation is $75 + 0.5*82 = 116$ and $75 + 0.5*98 = 124$.

Step4: Now we consider 0.1mm as the final slicing spacing on the new local window on [116mm, 124mm], output is 80 layers.

Fig.9 shows the entire process, and Fig.10 shows the contours plotted from the two layers which are related to the endpoints of the selected intervals in the curve of the points amount index. Those low and upper bounds for each selected window indicate the location of the complicated part of the human head model. Fig.11 is the graphs for the three stages of slicing. Among these three

graphs, the interval [75,135] in the top graph is extended to the next one, similarly, the interval [82,98] in the middle graph is extended to the bottom one, the colors are relatively matched.

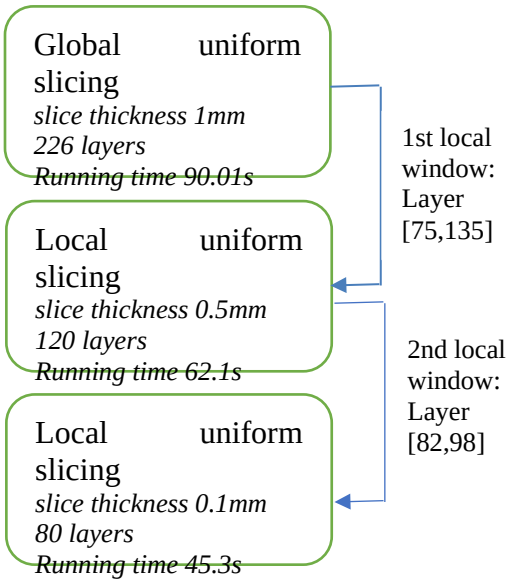


Fig.9. The full process of our method

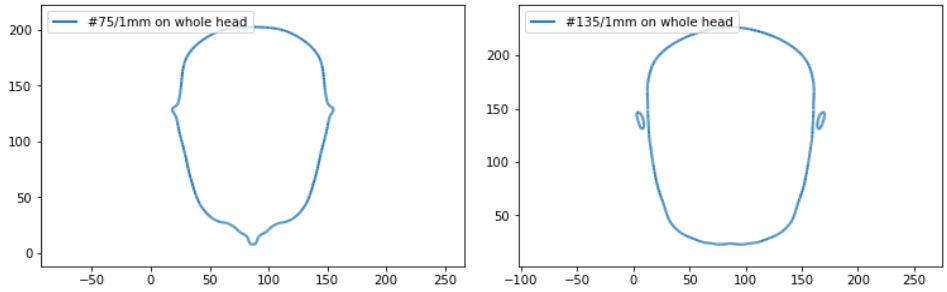


Fig.10. The contour plots for 4 layers related to the endpoints of the points amount of each layer

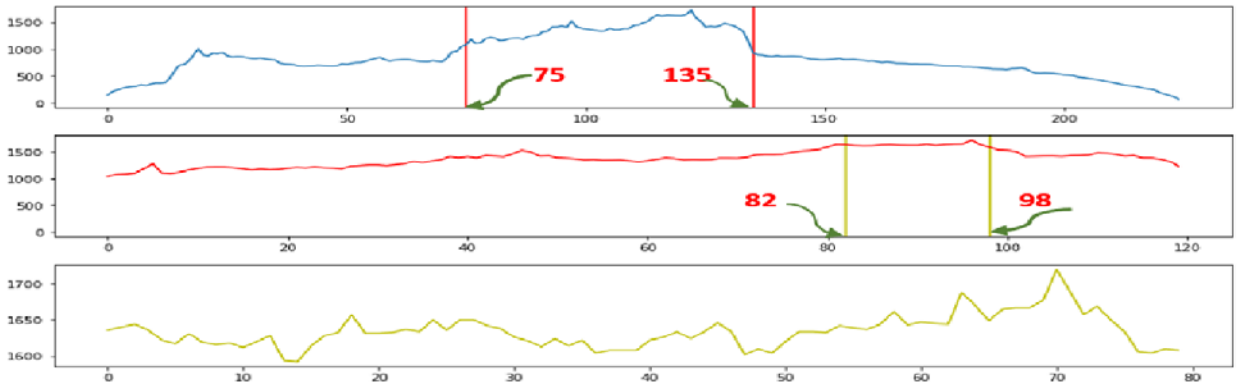


Fig.11: The graph of the points amount on the layer index

4. Conclusion and Discussion

The slicing procedure is an important element during the 3D printing steps, and the sliced contour is used in the generation of tool paths, for the layer-by-layer deposition required by the manufacturing process. The slicing method affects not only the precision of 3D printing parts but also production efficiency.

During our simulation experiments, we were able to get the comparison graphs for the points number on each layer, therefore we apply the adaptive slicing method according to the local window on the graph of points number along the layer index, i.e., set smaller slicing spacing on the high-density local area. However, there are still more work left for the further improvement. Our method aims at reducing the slicing running time by adjusting the slice thickness during slicing. Therefore, it is necessary to adopt the interpolation algorithms to obtain the layer information among the layers with big spacing in order to be ready for producing G-code file which is the terminal of the whole slicing process. Moreover, the adaptive approach here is still based on the concept stage, the future work will focus on how to accurately locate the boundaries by the mathematical algorithm support to adjust the slicing thickness during the process.

5. Acknowledgment

I would like to thank Prof. Billups at University of Colorado Denver for his advice and guidance.

6. References

[1] M.Y.Zhou, “STEP-based approach for direct slicing of CAD models for layered manufacturing”, *International journal of production research*, vol.43, no.15: pp.3273-3285, 2005.
[2] M.Szilvasi-Nagy, "Analysis of STL files", *Mathematical and computer modelling*, 38, pp.945-960, 2003
[3] H.P.Pan, T.R.Zhou, "Generation and optimization of slice profile data in rapid prototyping and manufacturing", *Journal of materials processing technology*, 187-188: pp.623-626, 2007.
[4] Zhengyan Zhang, Sanjay Joshi, "An improved slicing algorithm with efficient contour construction using STL files", *Int J Adv Manuf Technol*, 80: 1347-1362, 2015.
[5] S.H.Choi, K.T.Kwok, “A memory efficient slicing algorithm for large STL files", *Annual international solid freeform fabrication symposium*, pp.155- 162, 1999.
[6] S.H.Choi, K.T.Kwok, "Hierarchical slice contours for layered-manufacturing", *Computers in industry*, 48: pp.219-239, 2002.
[7] D.Ding, Z.Pan et.al, "Multi-direction slicing of STL models for robotic wire-feed additive manufacturing", *Annual international solid freeform fabrication symposium*, pp.1059-1069, 2015.
[8] M.Eragubi, "Slicing 3D CAD model in STL format and laser path generation", *International journal of innovation, management and technology*, Vol.4, No.4, pp.410-413, August 2013.
[9] H.Ye, C.Zhou et al.,"Mass customization: reuse of topology information to accelerate slicing process for continuous liquid interface production", *Proceeding of the 27th annual international solid freeform fabrication symposium*, pp.53-66, 2016.
[10] M.Vatani, A.R.Rahimi et al., "An enhanced slicing algorithm using nearest distance analysis for layer manufacturing", *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, Vol.3 No.1:pp.74-79, 2009.
[11] M.Vatani, A.R.Rahimi et al., "An enhanced slicing algorithm using nearest distance analysis for layer manufacturing", *International journal of mechanical, aerospace, industrial, mechatronic and manufacturing engineering*, vol 3 no.1, pp.74-79, 2009.

[12] J.B.Zhao, R.B.Xia et al., "A computing method for accurate slice contours based on an STL model", *Virtual and physical prototyping*, vol.4, no.1: pp 29-37, 2009.
[13] E.Sabourin, S.Houser et al., "Adaptive slicing using stepwise uniform refinement", *Rapid prototyping journal*, vol.2, No.4: pp.20-26, 1996.
[14] J.Tyberg, J.Behn, "Local adaptive slicing", *Rapid prototyping journal*, Vol.4, No.3, pp.118-127, 1998.
[15] S.Sikder, A.Barari et al., "Global adaptive slicing of NURBS based sculptured surface for minimum texture error in rapid prototyping", *Rapid prototyping journal*, Vol.6, No.6: pp.649-661, 2015.
[16] N.Siraskar, R.Paul et al., "Adaptive slicing in additive manufacturing process using a modified boundary octree data structure", *Journal of manufacturing science and engineering*, Vol.137, 2015.