Real-time slicing algorithm for Stereolithography (STL) CAD model applied in additive manufacturing industry

F A Adnan, F R M Romlay and M Shafiq
Faculty of Mechanical Engineering, Universiti Malaysia Pahang, 26600, Pekan, Pahang, Malaysia

E-mail: fb11002.ump@gmail.com

Abstract. Owing to the advent of the industrial revolution 4.0, the need for further evaluating processes applied in the additive manufacturing application particularly the computational process for slicing is non-trivial. This paper evaluates a real-time slicing algorithm for slicing an STL formatted computer-aided design (CAD). A line-plane intersection equation was applied to perform the slicing procedure at any given height. The application of this algorithm has found to provide a better computational time regardless the number of facet in the STL model. The performance of this algorithm is evaluated by comparing the results of the computational time for different geometry.

1. Introduction

Recently introduced Digital-Light-Processing (DLP) 3D printer utilizes mask projection by DLP projector onto photopolymer resin to cure the liquid form resin into solid model [1-2]. The mask projection is proven to be much faster and it provides uniform curing process due to each layer is being projected at once compared to SLA 3D Printer [3-4] . CAD file such as STereoLithography (STL) is considered as de facto in 3D printing. However, the STL file must be sliced into layers of 2D contours (as in Figure 1) before the contour projection process takes place [5-7]. In the past, researchers proposed several methods of this pre-processing stage of stereolithography such as uniform slicing and adaptive slicing methods [8]. All the mentioned methods are performed before the process of contour projections, and often affect the output of the printing process which is called cusp height or staircase effect due to differential slicing height.

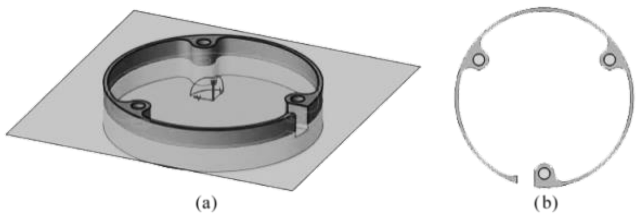


Figure 1. Contour generation process: (a) CAD model and a slicing plane located at certain slicing height (b) sliced contour.

This paper proposed a real-time slicing algorithm for contour mask generation with low computational time. The terms real-time in this context refers to the ability of the algorithm to compute a single layer of contour with respect to the changing slicing height.

2. STL Format

STL format is considered as a de facto in rapid prototyping. It was developed back in 1987 by 3D Systems for 3D CAD model conversion. The conversion involves tessellation process which forms multiple triangular facet of XYZ to represent the surface feature of the geometry. There are two types of STL format: binary STL and ASCII STL format. The study only focused on ASCII STL format. An ASCII STL format always begins with the *solid name* syntax where name is an optional and often omitted with spaces, then followed by *facet* syntax with its normal vector. The *outer loop* marks the beginning of vertices data which is used as P_1, P_2, and P_3 respectively in the algorithms. The *n* and *v* is a floating number with numbering format of *sign-mantissa-"e"-sign-exponent*, e.g. "2.999381e-002".

```
solid name
facet normal ni nj nk
  outer loop
    vertex v1x v1y v1z
    vertex v2x v2y v2z
    vertex v3x v3y v3z
  endloop
endfacet
endsolid name
```

The *endfacet* syntax marks the end of the facet. An STL file may consists of more than one facet, usually thousands, depending on the complexity of the geometry. If there is another facet, new *facet* syntax can be located after the previous *end facet* syntax. The STL file will have *endsolid name* syntax as the terminator of the file [9].

3. Algorithms

3.1 Facet Class

The proposed algorithm stored each facet in a list of facet class to be read by the slicing algorithm. The class stored information such as the facet vertices (P_1, P_2, P_3), maximum and minimum Z-height of the facet. The maximum/minimum Z value is later used to filter out other facets except the ones which intersect with the slicing plane by comparing $z_{min} \leq z_{slice} \leq z_{max}$ for each facet in the list.

3.2 Slicing Algorithm

The slicing algorithm is a method of converting each triangular facet into each respective line segment. These line segments can later be connected into contour lines by using contour generation algorithm [10]. Facets in STL format are arbitrary and in various orientations. Figure 2 and Table 1 describe every possible interaction between the slicing plane and the facets [5,11]. Each interaction must be handled properly since a line segment only requires 2 distinct points.

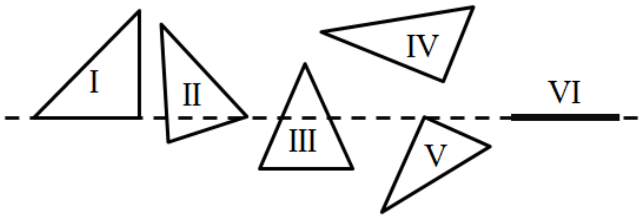


Figure 2. Possible orientation and position of facets.

Table 1. Definition of facet-plane intersection.

Case	Interaction of facet and plane	Possible Point
I	Line through one side of the facet	4
II	Line bisecting the facet through one vertex	3
III	Line bisecting the facet through two sides	2
IV	No intersection	0
V	Vertex intersection	2
VI	Parallel intersection	6

Table 1 classifies possible number of point that will be generated for each case of interaction mentioned above when sliced. Only Case I, II, and III will form the correct line segment. The rest of the cases are ignored. In Case V, the generated point will be the same and will not produce a line segment. For Case VI, this is considered as redundancy because in actual model, there will be another Case I facet at the exact position but in perpendicular (Case I) to the one in Case VI. Hence, Case VI must be ignored to eliminate overlapping line segment. Lastly, Case IV, there is no intersection and the algorithm will ignore it and proceed to search the next intersecting facet.

Intersections at vertex will produce two intersection points due to the facet edges that share the same vertex. This occurs in Case I, II, V, and VI. Hence, these cases must be handled properly. First, the proposed algorithm checks whether the edge is in parallel or it intersects with the plane to isolate Case VI and eliminates the coinciding edge in Case I. Case II is handled by eliminating one recurring intersection point. Case V can be identified when there exist only two intersection points which are similar.

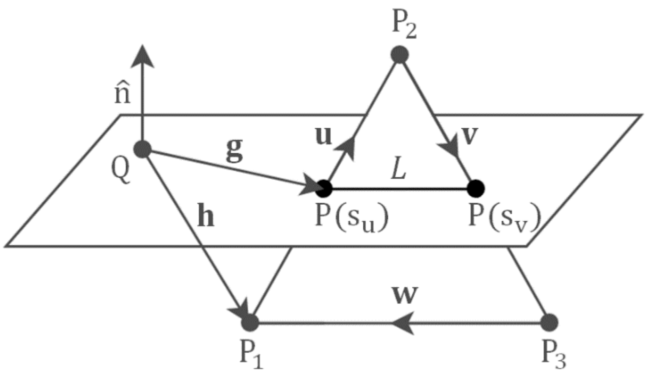


Figure 3. Facet-plane intersection diagram.

The proposed slicing algorithm is fundamentally based on line-plane intersection equation. Consider one side of the facet as line L that connects two vertices. In 3D, a line is either parallel to a plane or intersects it in a single point (see Figure 3). Parametric equation of a line can be written as:

$$P(s) = P_o + s(P_f - P_o) \tag{1}$$

and $Q = \langle x, y, z_{slice} \rangle$ is a point that exists on the plane where x and y can be any point on the plane (usually set at the origin), z_{slice} is the slicing height and the unit vector $\hat{n} = \langle 0, 0, 1 \rangle$ is the plane normal (for the case of slicing with respect to Z axis)**.** Changing the value of z_{slice} will affect the slicing height and resulted in different line segment. The algorithm first checks whether intersection exist between the lines and the plane by using the dot product criterion, $\hat{n} \cdot \mathbf{u} = 0$, $\hat{n} \cdot \mathbf{v} = 0$, $\hat{n} \cdot \mathbf{w} = 0$. These criterions will also handle both Case I and Case VI by eliminating the parallel line in the facet. As seen in the Fig.3, the $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ is the direction vector for each side of the facet in clockwise order (P_1, P_2, P_3, P_1). The dot product will become zero if there is no intersection which also means that the line is parallel to the plane and will not intersect with the plane. Should the line intersect with the plane such as the line between point P_1 and P_2 where $P_o = P_1$, $P_f = P_2$ and $s = s_u$, the $\hat{n} \cdot \mathbf{u} \neq 0$. At the intersection point, the direction vector from $P(s)$ to Q is $\mathbf{g} = \mathbf{h} + s\mathbf{u}$ where vector $\mathbf{h} = P_o - Q$. Since the vector $\mathbf{g} = P(s) - Q$ lies on the plane, previous criterion can be used, hence it is known that $\hat{n} \cdot \mathbf{g} = \hat{n} \cdot (\mathbf{h} + s\mathbf{u}) = 0$. Solving this will get:

$$s = \frac{-\hat{n} \cdot \mathbf{h}}{\hat{n} \cdot \mathbf{u}} = \frac{\hat{n} \cdot (Q - P_o)}{\hat{n} \cdot (P_f - P_o)} \tag{2}$$

By using Equation 3 and the s_u value obtained from Equation 4, the equation now becomes:

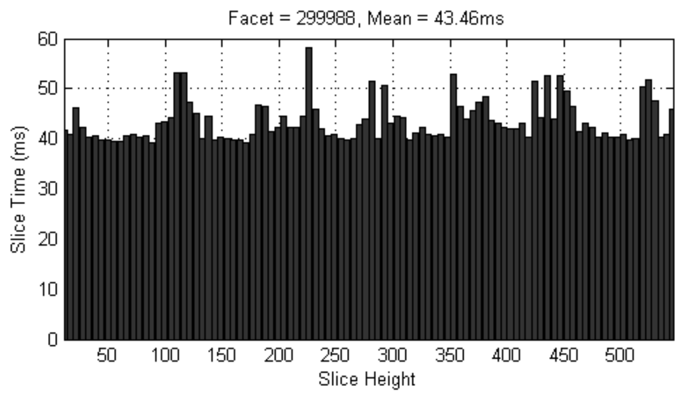
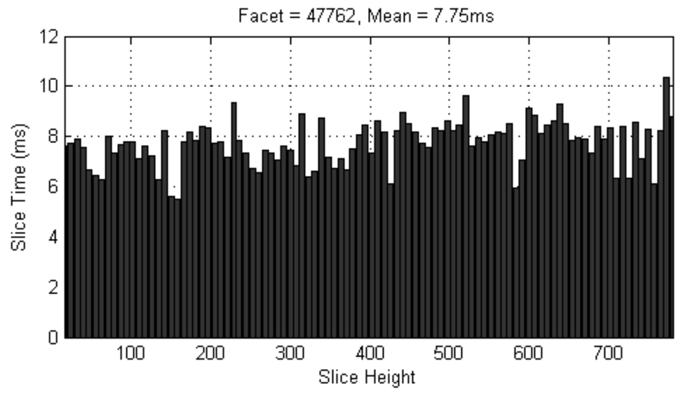
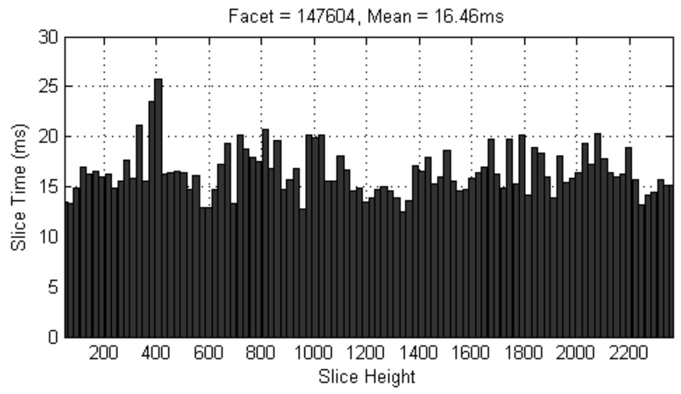
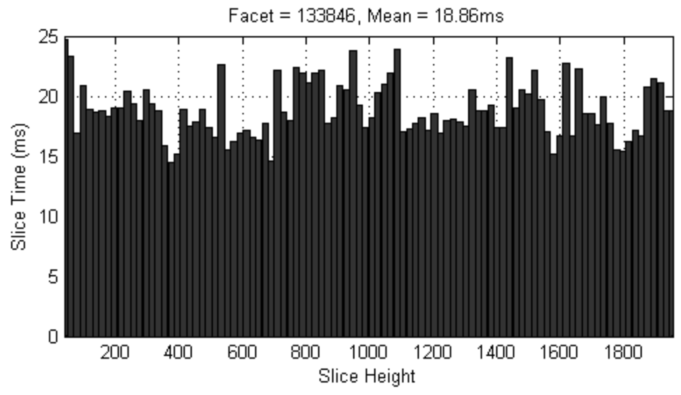
$$P(s_u) = P_1 + s_u(P_2 - P_1)$$

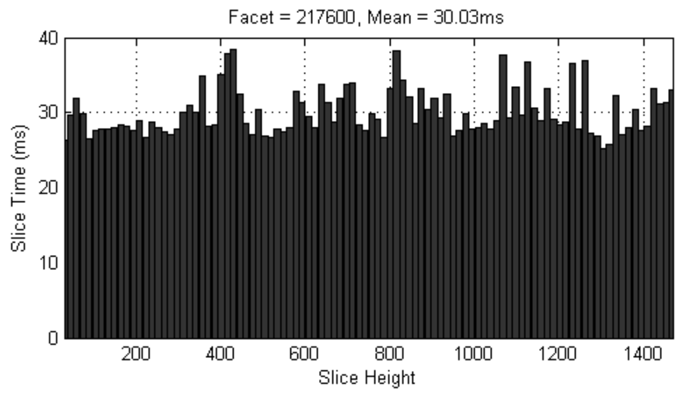
However, the value s_u must be verified to be within the range $0 \leq s \leq 1$ to ensure that the intersection point exists only within line. The same method is applied to the rest of the lines for which the criterion is fulfilled. In Figure 3 the $P(s_u)$ and $P(s_v)$ will generate a line segment L that will be used in the contour generation algorithm.

4. Experimental Results

The mentioned algorithm is developed in VB.NET programming language using Microsoft Visual Studio 2012 and executed on LENOVO IdeaPad i3-2350M CPU at 2.30GHz 6GB RAM and running on Windows 7 SP1 Home Premium (64-bit) operating system. The algorithm is tested with multiple STL model with different complexity and its results are shown in Table 2.

Table 2. Result of slicing algorithm.



As shown in Table 2, the results consist of slicing time at different slicing height starting from the bottom of the geometry until the top of each respective model. The consistency in the performances of each slicing height are noticeable. Hence, it can be represented as mean average slicing time as shown in the graph next to the 3D model. Each model contains different facet number indicating the complexity of the model. The graphs shown above are limited to 100 slices of contour. However, this can be adjusted regardless any value depending on the DLP 3D printing machine resolution. Based on each individual facet number and mean average, a graph of facet number versus mean can be plotted to analyse its correlation as shown in Figure 4 below.

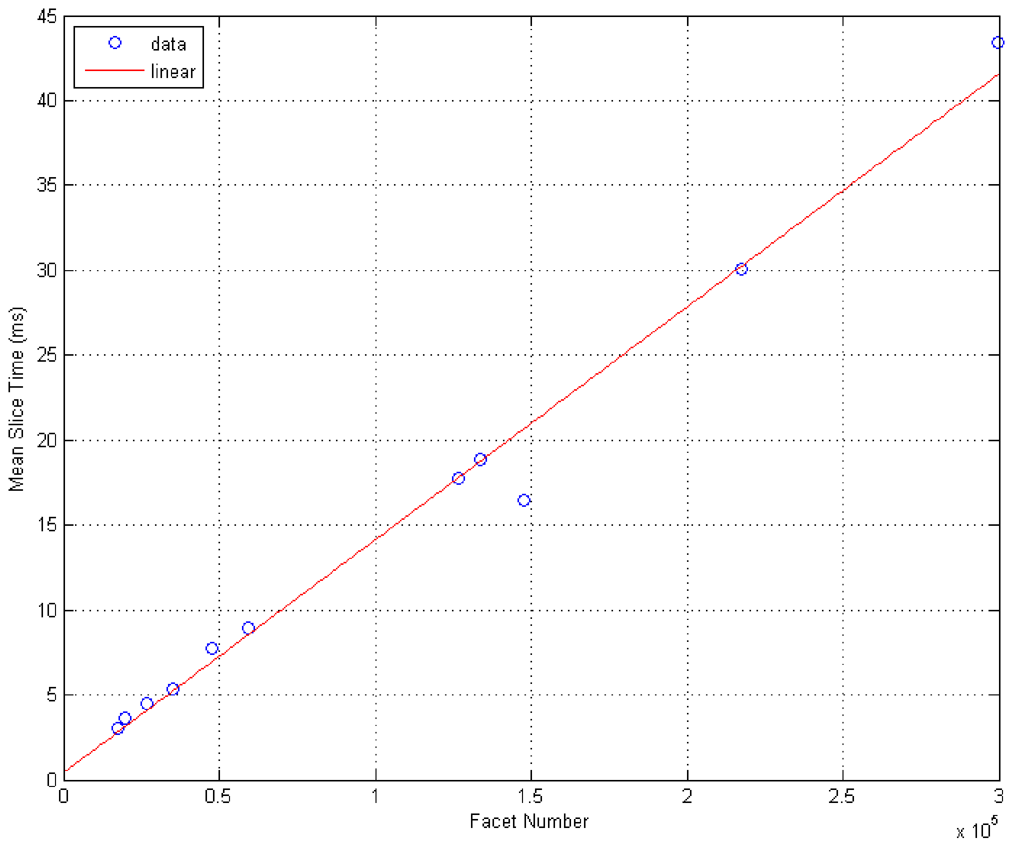


Figure 4. Graph of Facet Number vs. Mean.

As seen in Figure 4, mean slice time is linearly increase with respect to the facet number. More complex geometry will result in longer slicing time. According to Big-O notation which is considered as standard measures for algorithm complexity, the mentioned slicing algorithm runs on $O(n)$ that tells increasing number of elements will increase the algorithm execution time.

5. Conclusion

This paper presented real-time slicing algorithm which runs on $O(n)$ that able to slice an STL model at any given height directly with low computational time. The algorithm is fundamentally based on line-plane intersection equation and several case handlings as previously mentioned. The paper also focused on real-time evaluation of the slicing process so that the responses at each slicing heights are measured. Based on the results, it is found that, the slicing time of a geometry is consistent regardless the slicing height. However, for complex model, the slicing time will linearly increase.

Usually, a slicing algorithm is paired with a contour generation algorithm to connect each sliced line segments into single/multiple closed contour loop. This can be the extended work for the proposed slicing algorithm.

Acknowledgement

This research was funded by Universiti Malaysia Pahang (UMP) research grant number RDU160387. Credits to everyone whom directly and indirectly involved in helping us to achieve all the specified objectives of this research.

References

[1] Limaye A S and Rosen D W 2007 Process planning method for mask projection micro-stereolithography *Rapid Prototyp. J.* **13** 76–84

[2] Katal G, Tyagi N and Joshi A 2013 Digital Light Processing and its Future Applications *Int. J. Sci. Res.* **3** 1–16

[3] Chia H N and Wu B M 2015 Recent advances in 3D printing of biomaterials *J. Biol. Eng.* **9** 4

[4] Zhao X 2009 Process planning for thick-film mask projection micro stereolithography

[5] Topçu O, Taşcıoğlu Y and Ünver H Ö 2011 A Method for Slicing CAD Models in Binary STL Format *6th Int. Adv. Technol. Symp.* 141–8

[6] Minetto R, Volpato N, Stolfi J, Gregori R M M H and da Silva M V G 2017 An optimal algorithm for 3D triangle mesh slicing *Comput. Des.* **92** 1–10

[7] Pan X, Chen K and Chen D 2014 Development of rapid prototyping slicing software based on STL model *Proc. 2014 IEEE 18th Int. Conf. Comput. Support. Coop. Work Des. CSCWD 2014* 191–5

[8] Zhou M Y, Xi J T and Yan J Q 2004 Adaptive direct slicing with non-uniform cusp heights for rapid prototyping *Int. J. Adv. Manuf. Technol.* **23** 20–7

[9] Huang S H, Zhang L C and Han M 2002 An effective error-tolerance slicing algorithm for STL files *Int. J. Adv. Manuf. Technol.* **20** 363–7

[10] Zhang Z and Joshi S 2015 An improved slicing algorithm with efficient contour construction

using STL files *Int. J. Adv. Manuf. Technol.* **80** 1347–62

[11] Brown A C and De Beer D 2013 Development of a stereolithography (STL) slicing and G-code generation algorithm for an entry level 3-D printer *IEEE AFRICON Conf.*