Proceedings of the ASME 2016 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2016
August 21-24, 2016, Charlotte, NC, USA

DETC2016/CIE-60245

IMAGE BASED SLICING AND TOOL PATH PLANNING FOR HYBRID STEREOLITHOGRAPHY ADDITIVE MANUFACTURING

Hang Ye[1], **Chi Zhou**[1]*, **and Wenyao Xu**[2]

[1]Department of Industrial and Systems Engineering
[2]Department of Computer Science and Engineering
University at Buffalo, The State University of New York, Buffalo, NY 14260

Email: hye2@buffalo.edu, chizhou@buffalo.edu, wenyaoxu@buffalo.edu

ABSTRACT

Hybrid stereolithograpgy (SLA) process synthesizes the laser scanning based SLA system and mask projection based SLA system. It adopts laser as the energy source for scanning the border of a 2D pattern, whereas a mask image is used to solidify the interior area. By integrating the merits of the two subsystems, the hybrid SLA process can achieve relatively high surface quality without sacrificing the productivity. For the hybrid system, closed polygon contours are required to direct laser scanning, and a binary image is also needed for mask projection. We proposed a novel image based slicing method. This method can convert the 3D model into a series of binary images directly, and each image is corresponding to the cross-session of the model at a specific height. Based on the resultant binary image, we use image processing method to gradually shrink the image. The contours of shrunk image are traced and then restored as polygons to direct the laser spot movement. The final shrunk image will serve as the input for mask projection. The experimental result of several test cases demonstrate that proposed method is substantially more time-efficient than traditional approaches.

Keywords: *Additive Manufacturing, Stereolighography, Slicing, Contour Tracing, Image Shrinking.*

*Corresponding author. Phone: (716) 645-4706 Fax: (716) 645-3302

1 Introduction

Among various additive manufacturing (AM) processes, stereolithography apparatus (SLA) was the first commercialized one, and it is one of the most popular AM technologies nowadays. There are two typical stereolithography configurations [1]. One is to use laser to illuminate the liquid resin, and the laser will scan every point within the cross-sectional area. The advantage of this configuration is that it will generate comparatively high accuracy, but it's time-consuming as the laser routing has to pass all points of the cross-section. The other configuration is the application of digital light projection (DLP) to illuminate the liquid resin, and it can dramatically reduce the production time, as whole layer is cured once by exposing through a designed digital mask. But its accuracy is restricted by the resolution of DLP device. Extensive research work has been done to improve the efficiency and/or accuracy of SLA from different perspectives [2–7].

However, most researchers in this field have been focusing on improving either one of the two processes, and it would always result in making tradeoff between efficiency and accuracy. This inherent tradeoff has been identified as a major technology barrier existing across many AM processes [8]. And most recently, Zhou *et al*. proposed a hybrid SLA process to combine the laser scanning with projection such that one subsystem can complement the other to improve overall performance [9]. The proposed hybrid system is the combination of laser scanning based and mask projection based SLA. It adopts laser as the en-

ergy source for the border of a 2D pattern, whereas a mask image is used to solidify the interior area. The hybrid SLA system can take advantages from both sides, i.e., achieving relatively high surface quality (by laser) without sacrificing the productivity (by mask image).

Although traditional slicing and tool path planning approaches can be applied to hybrid SLA system seamlessly, to maximize its advantage from productivity and surface quality, a configuration-specific method is desirable. In this paper, we will introduce a novel slicing and tool path planning method which can well suit the hybrid system. The remainder of this paper is organized as follows: Section 2 will review how the existing slicing and tool path planning approaches can be adopted by the hybrid system, and a new data flow paradigm will be proposed. Section 3 will introduce an image based slicing method which can convert the STL format 3D model into binary images directly. Section 4 will describe an image erosion method to shrink the original binary image based on image processing technology. Efficiency analysis and experimental result will be presented in Section 5. Conclusion and future work will be discussed in Section 6.

2 Data Flow for Hybrid Stereolithography System

2.1 Flow Chart of Slicing and Tool Path Planning

The hybrid system synthesizes laser scanning based SLA system and mask projection based SLA system. Therefore, input files from both sides are required for this system. To be specific, two input files are necessary: 1) contours for laser scanning; 2) binary image for mask projection. We will review how these two required input data are generated using traditional method in this section.

AM process usually takes an STL (STereoLithography) file as raw data, which is the triangulated surface representation of a 3D object. To build an object in a layer by layer fashion, the shape of cross-section at each layer is required. The process to obtain the geometric property of cross-section area for each layer is usually called "slicing". It can be seen as using a plane to slice the model along a given direction, so all resultant slices are parallel to each other. Common practice for slicing is to calculate the intersections between the plane and triangular meshes from STL file, and then all intersections are re-organized to obtain a contour (simple polygon) according to topological information. For laser scanning based SLA process, a series of polygons are generated after slicing by offsetting, and the offset distance t is always negative, i.e., inward offsetting for outside contour and outward offsetting for inside contour. The magnitude of the offset distance is set as multiple of laser spot diameter, thus the distance between any two consecutive laser paths is equal to the diameter of laser spot. All these resultant polygons except for the last one will serve as the path for laser scanning in the hybrid system. For mask projection based SLA process, the input is a series of binary (or gray scale) images. The liquid photopolymer will be solidified according to the patterns shown in these images. Each geometric pattern is defined by the area enclosed by a contour (or by nested outside and inside contours), and pixel value for pixels within this area will be set as 1 for binary image. A scanline rendering algorithm can be adopted to generate these binary images by utilizing the contour information. In the hybrid system, the area for mask projection is corresponding to the innermost polygon generated from the outside contour and the outermost polygon generated from the inside contour. The liquid photopolymer at other areas on the cross-section will be cured by laser scanning. The flow chart of the slicing and tool path planning at a specific layer height is shown in FIGURE 1(a).

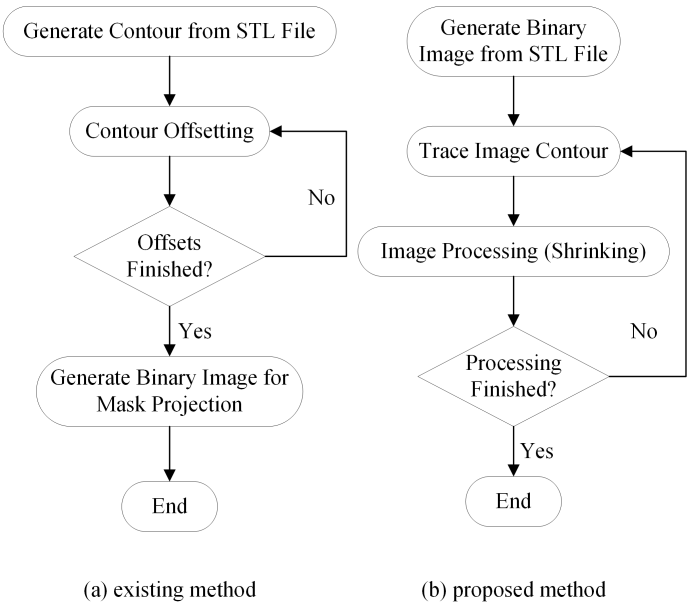


FIGURE 1. Flow Chart of Slicing and Tool Path Planning for Hybrid SLA.

In the upper part of FIGURE 2, a bunny model is used to illustrate the traditional slicing and tool path planning method for hybrid system. FIGURE 2(a) shows the original 3D model. FIGURE 2(b) shows the sliced contour at a specific height. FIGURE 2(c) shows the offset contour series, and these polygons will serve as the laser path. FIGURE 2(d) shows the final binary image which will be used for mask projection.

The existing slicing and tool path planning method reviewed above can fulfill the requirement of the hybrid SLA system, but it also suffers from several drawbacks. For example, during the slicing process, the connectivity among intersection points has to be restored. Compared with the intersection points calculation, retrieving this topological information takes much more

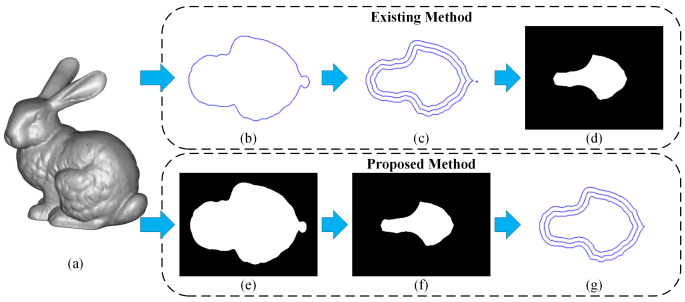


FIGURE 2. Bunny Model: (a) original 3D model, (b) original contour, (c) offset contours, (d) mask image, (e) original image, (f) shrunk image, (g) reconstructed contours.

computational time. If a given contour contains n segments, the intersection calculation takes $O(n)$ time, but ordering these intersections will take $O(n^2)$ time if "closest point" method [1]) is adopted. Furthermore, in order to conduct the offsetting operation accurately, the contour orientation has to be consistent, e.g., all outside contours are oriented counter-clockwise, whereas all inside contours are oriented clockwise. Extra effort is required to maintain this consistency. Besides the aforementioned disadvantages, the most challenging part stems from the offsetting operation itself. Because offsetting is not a topology preserving operation, it is possible that the original object and the offset object are not homeomorphic. This feature of topological inconsistency makes polygon offsetting problem extremely difficult to tackle. Tremendous effort has been made to maintain the robustness of offsetting operation [10–12], however, such effort increases the algorithm complexity dramatically.

2.2 Image based Method

Method to solve an engineering problem can be categorized as either analytical method or numerical method. The method reviewed in previous section adopts analytical method to generate the exact contour of the cross-section area by calculating the intersection points between a triangle and a given plane. In the later polygon offsetting operation, the offset polygonal contours have been generated also by the analytical method. Analytical method holds a significant advantage that it provides the exact solution to the original problem without losing any accuracy, whereas numerical method can only provide an approximate solution. However, in the real world every engineering setup has its finite resolution, and any feature beyond this cannot be handled by that system. Under this circumstance, obtaining an exact solution by analytical method for an engineering problem might not be necessary. For example, in the mask projection SLA process, if the size of a feature is smaller than the pixel size of the image, the pursuit of the exact representation for this feature is not necessary. Furthermore, the time complexity of analytical method is usually higher than numerical method in order to preserve the accuracy. In other words, to solve the same problem numerically can achieve higher efficiency with the tradeoff of accuracy. Motivated by this rational, in this paper we will derive an image based approach which is essentially numerical. This approach can provide higher efficiency than the existing analytical method while maintain sufficient accuracy. The flow chart of proposed approach is shown in FIGURE 1(b).

The proposed approach generates the binary image directly from the STL file. This binary image represents the cross-section area of a 3D object at a specific height, thus it can be used in mask projection based SLA process directly. But for the hybrid system, this image needs to be further processed to create: 1) contour path to direct laser scanning and 2) shrunk image for mask projection. We first extract the boundary pixels from the image by contour tracing algorithm, and then carry out image shrinking by morphological image processing and Boolean operation based on these boundary pixels. The extracted image contours will be converted into the laser scanning path, and the final resultant binary image will be used for mask projection. In the lower part of FIGURE 2, the same bunny model is used to demonstrate the proposed approach. FIGURE 2(e) shows the original image at a specific height. FIGURE 2(f) shows the final shrunk image. FIGURE 2(g) shows the restored contour set to direct laser scanning.

3 Binary Image Generation

3.1 Existing Slicing Approach

Additive manufacturing builds 3D object incrementally in a layer by layer fashion. Therefore, the 3D model needs to be cut into a set of slices in order to obtain geometric information for each slice. Most existing slicing methods fall into one of the two categories: non-topology based and topology based. For both methods, it is necessary to compute the intersection between a given line segment (defined by two vertices $V_1(X_1,Y_1,Z_1)$ and $V_2(X_2,Y_2,Z_2)$) and a slicing plane ($Z = Z_k$), and it can be easily calculated by linear interpolation [9].

For non-topology based method, each triangle facet from STL model needs to be traversed to check whether it intersects with current slicing plane or not. If n triangles intersect with current slicing plane, $2n$ end points from n line segments will be computed. Because these n line segments are connected end to end to form one or multiple closed polygons, there are only n distinct points in total. The connectivity can be determined by "closest point" method [1].

Topology-based method is also called "marching" algorithm. The key of this method is the half-edge data structure (also referred as "doubly-connected edge list" in [13]). All edges from STL model need to be pre-processed in order to preserve the topological information. With these information, we can march from one triangle to the next until we get back to the starting triangle to close the polygonal contour [14].In this algorithm, if n

triangles intersect with current slicing plane, only n intersection points need to be computed, and the connectivity is exactly the same as the sequence of marching.

Non-topology based method is easy to understand and implement, but ordering the line segments is time-consuming. Although the time complexity of topology based method is lower than non-topology based method, a sophisticated data structure has to be adopted, and the construction of this data structure is also time-consuming. In addition, this method is not as robust as non-topology based method, i.e., models with For example, if the order of vertices from a triangle doesn't follow the right-hand rule, this method cannot generate the desired result.

3.2 Image based Slicing Method

In order to extract the geometric property of the cross-section A which is a 2D planar area, the existing method adopts a boundary representation, i.e. using its contours to represent the 2D area. Here, the segment contour, denoted as ∂A, is the topological boundary of A, and it is an orientable, continuous, and closed 1D manifold embedded in planar R^2. In this section, an image based method is presented. It generates a binary image with finite resolution γA directly from the STL file to represent A. Major steps of image based slicing algorithm can be described as: 1) convert STL file into sampling point cloud; 2) generate a series of images according to the sampling points from bottom to top in an accumulative manner.

3.2.1 Sampling Point Conversion For a solid S in R^3, its STL file can be seen as a triangulated surface representation of S. In the context of geometric modeling, surface is defined as "an orientable continuous 2D manifold embedded in R^3" [15], denoted as ∂S. Here, ∂S is an infinite set whose elements are orientable points. It also can be seen as the union of all orientable triangle facets T in STL file, and each triangle facet is comprised of infinite points from its closure. The sampling points are some points we intentionally select from ∂S in order to approximately represent the surface. In other words, the set of sampling points I is a proper subset of ∂S.

Suppose the binary image to be generated is positioned at a rectangle area defined by its four vertices (0, 0, 0), (W, 0, 0), (W, H, 0), and (0, H, 0), and the 3D model will also be placed in this planar domain. If the resolution of the image is $m \times n$, each pixel is a grid with the width $d = W/m$ (or H/n). Among all elements in ∂S, the points with the following X- and Y- coordinates are selected as sampling points.

$$\begin{cases} X_{ij} = (i-0.5)d, \, for \, i = 1,2,...m \\ Y_{ij} = (j-0.5)d, \, for \, j = 1,2,...n \end{cases} \tag{1}$$

It can be imagined as a series of rays radiate vertically (Z-direction) from the pixel centers, and if the ray hits a triangle facet on ∂S, a sampling point will be generated. By doing this ∂S has been uniformly discretized in both X- and Y- direction. Each triangle facet is a 2D object in R^3, and can be defined by its three vertices $V_1(X_1,Y_1,Z_1)$, $V_2(X_2,Y_2,Z_2)$, and $V_3(X_3,Y_3,Z_3)$. The plane in which the triangle facet lies can be represented as Eqn.(2). If the ray intersects with the triangle facet, the intersection can be computed by substituting Eqn.(1) into Eqn.(2).

$$\begin{vmatrix} X-X_1 & Y-Y_1 & Z-Z_1 \\ X_2-X_1 & Y_2-Y_1 & Z_2-Z_1 \\ X_3-X_1 & Y_3-Y_1 & Z_3-Z_1 \end{vmatrix} = 0 \tag{2}$$

It is noteworthy that Eqn.(2) is the equation for the plane that the triangle facet lies in, so it is possible that the point obtained by solving Eqn.(1) and (2) simultaneously lies in the exterior of the triangle facet. To tackle this, the incident pixels for each triangle facet need to be identified before the substitution. The projection of triangle facet T_i on X-Y plane, denoted as T_i', is obtained by simply omitting the Z-coordinate of each vertex. The incident pixels of T_i can be identified by conducting scanline triangle rendering on T_i' [16]. Once the incident pixels of T_i are identified, their positions can be substituted into Eqn.(2) to compute the Z-coordinates for sampling points.

In order to distinguish the interior from the exterior of the solid S, ∂S is orientable. An outward pointing normal vector $\vec{n}$ is used to represent the orientation of the triangle facet. Similarly, all sampling points need to be orientated for the same consideration. Since these sampling points I will be converted into γA later, we need to know whether each point in I is an entering point or an exit point along a given direction. In the context of geometric modeling, the entering point of solid S can be defined as the first point of a local feature we met along a given direction, and the exit point is the last point we met along the same direction. If we select the positive Z-axis as the working direction, all sampling points on the triangle facet with negative Z-component normal vectors are entering points, and those with positive Z-component normal vectors are exit points. For the triangle facet with zero Z- component normal vector, the facet is perpendicular to X-Y plane, so it won't generate any incident pixels.

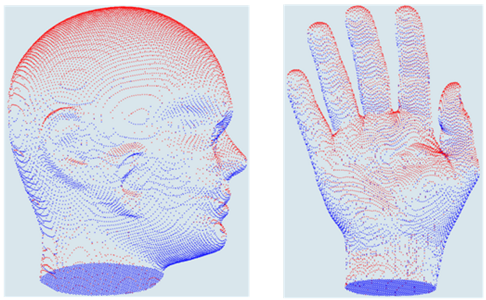
The pseudo-code for converting STL file into sampling points is shown as below. Algorithm ScanlineRendering($T[i]$, d) is the scanline triangle rendering, and algorithm Calculate$Z$(X, Y) is used to compute Z-coordinate according to Eqn.(2). Two examples of sampling point conversion are shown in FIGURE 3. The points in blue are entering points, and the points in red are exit points.

Algorithm 1 ConvertToSamplingPoints

Input: triangle facet T, pixel width d

Output: sampling point cloud I

```
I ← φ
for each triangle facet T[i] in T do
    if n[i].Z <0 then
        entering ← true
    else if n[i].Z <0 then
        entering ← false
    end if
    incidentPixel ← ScanlineRendering(T[i], d)
    for each pixel incidentPixel[j] in incidentPixel do
        S.col ← incidentPixel[j].column
        S.row ← incidentPixel[j].row
        X ← (S.column-0.5) × d
        Y ← (S.row-0.5) × d
        S.Z ← CalculateZ(X, Y)
        S.entering ← entering
        I ← I ∪ S
    end for
end for
return I
```



(a) head model (b) hand model

FIGURE 3. Sampling Point Conversion.

3.2.2 Binary Image Generation In previous section, the STL file has been converted into orientable sampling points. However, the sampling points are only the intermediate product, and they cannot be used as the input for layer based manufacturing process. In this section, those sampling points will be converted into a series of image in an accumulative manner, and each image is corresponding to the cross-section area for a given slicing plane ($Z = i \times d\_LT$, for $i > 0$, and d_LT is the layer thickness). The basic idea of this accumulative method is that the image for the $(i+1)^{\text{th}}$ layer, P_{i+1}, is generated by manipulating the pixel values of the image for the i^{th} layer, P_i, according to the sampling points between these two consecutive layers.

For the $(i+1)^{\text{th}}$ layer, the binary image P_{i+1} with finite resolution $m \times n$ can be defined by a set of pixel values $g_{i+1,j}$ ($1 \leq j \leq m \times n$), where j is corresponding to the pixel position. Initially $g_{i+1,j}$ is set as the same as $g_{i,j}$. At position j, for each sampling point Q_k between the i^{th} layer and the $(i+1)^{\text{th}}$ layer, $g_{i+1,j}$ has to be updated once. These updates represent the local feature change along the building direction. If Q_k is an entering point, $g_{i+1,j} = g_{i,j} + 1$, otherwise, $g_{i+1,j} = g_{i,j} - 1$. Initially, P_0 is created by setting all pixel values $g_{0,j}$ as 0 for $1 \leq j \leq m \times n$.

For a closed 2D manifold, these two types of sampling points must occur one after the other, and at each pixel position the total number of sampling points must be even. Therefore, the relationship between $g_{i,j}$ and $g_{i+1,j}$ can be summarized as Eqn.(3).

$$g_{i+1,j} = \begin{cases} g_{i,j} & \text{if number of } Q_k \text{ is even} \\ (g_{i,j}+1) \mod 2 & \text{if number of } Q_k \text{ is odd} \end{cases} \quad (3)$$

The pseudo-code for binary image generation is shown as below. The image is generated from the bottom to the top, and each sampling point only takes effect once. Therefore, sorting all sampling points according to their Z-coordinates can make this algorithm more efficient, and only the sampling points that fall between two consecutive layers need to be visited to generate an image at a specific height. In FIGURE 4, an example object is used to demonstrate how binary images with 16×12 resolution are generated from sampling point cloud.

3.2.3 Sampling Accuracy The proposed image based slicing algorithm selects finite number of points from ∂S as the sampling points, therefore, some features of ∂S will be missing. However, it is necessary to guarantee the topological equivalence between a 3D object S and the reconstruction from its digital images. In other words, S and its reconstruction should be homeomorphic. The importance of homeomorphism has been briefly discussed in [17]. Intuitively, the denser the sampling point cloud is, the more possible a 3D object S and its reconstruction are homeomorphic. But if the sampling point cloud is too dense, its generation and processing will be very time-consuming. In addition, each projector has its own finite resolution, so it will be pointless to use images with too much higher resolution. In this section, we will briefly discuss the sufficient condition to guarantee the homeomorphism between S and its reconstruction.

Algorithm 2 BinaryImageGeneration

Input: sampling point cloud I, layer thickness d_LT

Output: binary image set P

01. $P \leftarrow \phi$
02. $I' \leftarrow$ Sort I according to Z-coordinate
03. **for** $j \leftarrow 1$ **to** $m \times n$ **do**
04. $\quad g[j] \leftarrow 0$
05. **end for**
06. $i \leftarrow 1$
07. $k \leftarrow 1$
08. $size \leftarrow$ size of I'
09. **while** $k \leq size$ **do**
10. $\quad$ **if** $I'[k].Z \leq I'[1].Z + i \times d\_LT$ **then**
11. $\quad\quad j \leftarrow (I'[k].row\text{-}1) \times n + I'[k].column$
12. $\quad\quad$ **if** $I'[k].entering$ = true **then**
13. $\quad\quad\quad g[j] \leftarrow g[j] + 1$
14. $\quad\quad$ **else**
15. $\quad\quad\quad g[j] \leftarrow g[j] - 1$
16. $\quad\quad$ **end if**
17. $\quad\quad k \leftarrow k + 1$
18. $\quad$ **else**
19. $\quad\quad P[i] \leftarrow \phi$
20. $\quad\quad$ **for** $j \leftarrow 1$ **to** $m \times n$ **do**
21. $\quad\quad\quad P[i] \leftarrow P[i] \cup g[j]$
22. $\quad\quad$ **end for**
23. $\quad\quad P \leftarrow P \cup P[i]$
24. $\quad\quad i \leftarrow i + 1$
25. $\quad$ **end if**
26. **end while**
27. **return** P

The following definitions and theorem are given in [18], and Definition 1 and Theorem 1 are Definition 1 and Theorem 16 in [18], respectively.

Definition 1: A set $S \subset R^3$ is called *r-regular* if, for each point $x \in \partial S$, there exist two osculating open balls of radius r to ∂S at x such that one lies entirely in S and the other lies entirely in the complementary set of S.

Definition 2: Any set G which is a translated and rotated version of the set $\frac{2r'}{\sqrt{3}}Z^3$ is called a *cubic r'-grid* and its elements are called *grid points*.

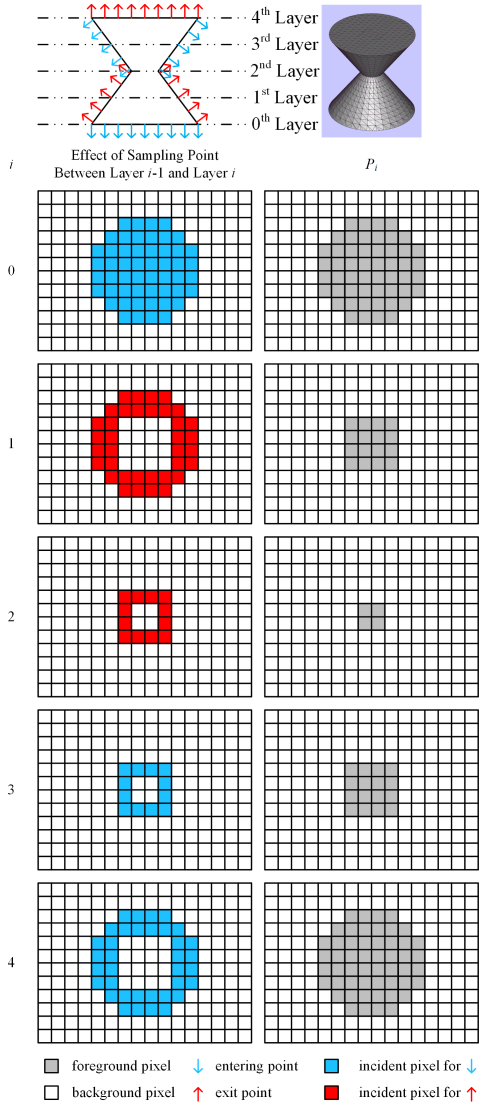


FIGURE 4. 3D Model Used for Image Generation Demonstration.

Theorem 1: Let S be an *r-regular* object and G be a *cubic r'-grid* with $2r' < r$. Then the result of a topology preserving reconstruction method is r-homeomorphic to S.

From Definition 2, the edge length of a *cubic r'-grid* $d' = \frac{2r'}{\sqrt{3}}$, and this relationship can be seen in FIGURE 5. According to Theorem 1, we can derive the following conclusion: For an *r-regular* object S and *cubic r'-grid* G, if the edge length of G, d', satisfies $d' < \frac{\sqrt{3}}{3} r$, the result of a topology preserving reconstruction method is r-homeomorphic to S.

In the context of additive manufacturing, in X-Y plane, d' is represented by the pixel width d. And in building direction (Z-

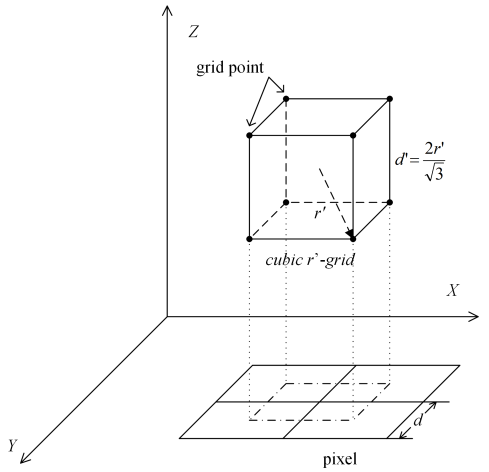


FIGURE 5. Cubic *r*'-Grid.

axis), *d*' is represented by the layer thickness *d_LT*. Therefore, we can conclude that for an *r-regular* object *S*, if both the pixel width *d* and the layer thickness *d_LT* are less than $\frac{\sqrt{3}}{3}r$, the homeomorphism between *S* and its reconstruction can be guaranteed.

It is not necessary that all 3D models are *r-regular*. For instance, a polyhedron does not fall in this category. However, for the model to be built by AM process, the magnitude of the parameter *r* can be interpreted as the size of the minimal spatial feature the model is going to present. This is because any AM setup has a finite resolution, and the objects fabricated by this specific machine can only present features whose sizes are greater than this defined resolution. This resolution is comprised of two independent components, lateral resolution (*X*- and *Y*-axis direction) and vertical resolution (*Z*-axis direction). For SLA process, the lateral resolution is defined by the diameter of laser spot (for laser scanning based SLA process) or the physical size of the pixel (for mask projection based SLA process). The vertical resolution is defined by the minimal step length of elevator and the penetration depth of the resin. If the minimal feature from an object is beyond these thresholds, it cannot be built as desired. Therefore, in order to properly build an object with minimal spatial feature size *r*, both the pixel size and the layer thickness have to be less than $\frac{\sqrt{3}}{3}r$.

4 Image based Tool Path Planning

4.1 Existing Tool Path Planning Method

As shown in FIGURE 1(a), existing tool path planning for SLA process takes polygonal contour information as input, and this contour information is obtained from slicing. Each polygonal contour L_0 encloses an area A_0, and successively negative offsetting will be conducted on this area with the offset distance t_i. By doing this, a series of shrunk areas A_i for i=1,2,3...n will be generated, and their corresponding polygonal contours L_i for i=1,2,3...$n-1$ will serve as the laser path. This can be summarized as Eqn.(4), where d_{ls} is the diameter of the laser spot, $\downarrow^*$ is the negative regularized offset (shrinkage) operator, and ∂A_i is the boundary of A_i. The area enclosed by the last polygonal contour will be converted into a binary image to serve as the input for mask projection.

$$\begin{aligned} &A_i = A_{i-1} \downarrow^* t_i, for\ i = 1,2,3\ldots n \\ &L_i = \partial A_i, \qquad for\ i = 0,1,2\ldots n \\ &t_i = \begin{cases} \frac{1}{2}d_{ls}, & for\ i = 1,n \\ d_{ls}, & for\ i = 2,3\ldots n-1 \end{cases} \end{aligned} \tag{4}$$

The offsetting operation here should be solid offsetting [19], which can be defined as follows [20]:

Definition 3: The regularized solid offset of a regular and non-empty set A by a positive distance t (expanding) is defined as $A \uparrow^* t = \{p : d(p,A) \le t\}$, where $d(p,A) = \inf_{q \in A} || p-q ||$ and inf denotes the greatest lower bound.

The negative offset of a regular and non-empty set A can be viewed as the complement of the positive offset of the complement of A, denoted as c^*A, and it can be defined as follows:

Definition 4: The regularized solid offset of a regular and non-empty set A by a negative distance t (shrinking) is defined as $A \downarrow^* t = \{p : d(p,c^*A) > t\}$, where $d(p,c^*A) = \inf_{q \notin A} || p-q ||$.

Here, the planar area A_i is an infinite set, therefore, it is impossible to implement solid offsetting operation according to its definition. The common practice for offsetting a polygonal area is to move each segment of the polygon along its normal by the offset distance t, and this operation is named as normal offsetting [19]. Since each polygonal area A_i can be defined by its contour L_i, each contour can be defined by finite number of segments, and each segment can be defined by its two end points, this approach is practicable. But the resultant polygon might not be simple any more, and identification and removal for the self-intersections are very difficult.

According to Definition 4, if $p \in A$, then $d(p,c^*A) = d(p,\partial A)$, and if $p \notin A$, then $d(p,c^*A) = 0$. Therefore, the negative offset can be viewed as shifting the center of a disc with radius t along ∂A, removing all the area swept by the disc, and the remainder is $A \downarrow^* t$. The image based tool path planning method, which will be introduced later in this session, is inspired by this idea. The proposed method converts A, ∂A, and the disc with radius t into binary images. This kind of conversion may lead to losing certain level of accuracy due to the finite resolution from the image, but it enables the computerized realization of aforementioned "sweep" and "removal" processes. More importantly, as a field which has been extensively studied, image processing

can equip us with a lot of classic operations and algorithms to realize the desired purpose.

The proposed tool path planning method extracts contour pixels from the image generated by image based slicing method. Then based on these pixels, mathematical morphological and Boolean operations are applied on the image to generate the shrunk image. The centers of contour pixels from intermediate shrunk images will be used to define the laser path, and the final shrunk image will serve as the input for mask projection.

4.2 Image Contour Tracing

According to proposed image based slicing algorithm, a binary image will be generated from STL file for a specific layer. This binary image may contain multiple isolated objects, and each object may have multiple contours, e.g., object with holes. If we want to use boundary pixel set to represent the given object, either inner boundary or outer boundary can be adopted [21]. The inner boundary itself is part of the object, and each of its elements has at least one neighbor which is not an element of the given object. The outer boundary is defined in the opposite way. It is part of the complement of the object, and each of its elements has at least one neighbor which is an element of the given object. In this section, we will use 8-connected [21] inner boundary as the image contour. By saying 8-connected, we mean that for a given pixel p, any pixel either shares an edge or a corner with p is an 8-connected neighbor of p. Similarly, 4-connected neighbor is defined as any pixel that shares an edge with the given pixel. FIGURE 6(a) shows pixels which are 8-connected with a given pixel p, whereas FIGURE 6(b) shows pixels which are 4-connected neighbors of a given pixel p. The position numbering of a neighborhood pixel relative to a given pixel p is shown as FIGURE 6(c).

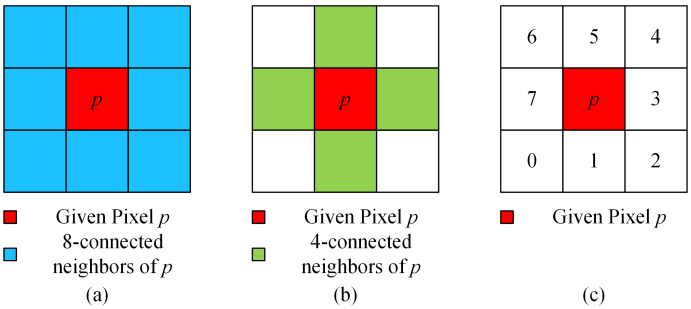


FIGURE 6. Neighbor Numbering for A Given Pixel.

As each object has non-ambiguous contours, the connectivity of contour pixels is unique. In other words, following a given direction, if we know the i^{th} contour pixel p_i, the $(i{+}1)^{\text{th}}$ contour pixel p_{i+1} is unique. In order to trace the contour, an initial contour pixel p_1 is necessary to serve as the starting point. One observation is that an 8-connected contour pixel (except the pixel on the image border) is a foreground pixel which has at least one 4-connected background neighbors. We can search the image from the northwest corner until a foreground pixel is found. This pixel must be a contour pixel, as its left neighbor, if exists, is a background pixel. If its left neighbor doesn't exist, it's on the border of the image (the 1st column), and it's also a contour pixel.

Then we can search the neighborhood of a known contour pixel p_i to find the next contour pixel p_{i+1}. This searching will be performed in a given direction, e.g., counter-clockwise (corresponding to the ascending order of the neighbor numbering in FIGURE 6(c)). This process can be assumed as rotating an imaginary line segment about p_i, so it is named as radial sweep. To properly trace the object contour, the selection of searching starting pixel, denoted as q_i, is critical, and the following two observations should be taken into consideration:

1. If the searching starts from a background pixel, the first foreground pixel swept must be a contour pixel;
2. If there are more than one contour pixels 8-connected with p_i for $i > 1$, one of them must be p_{i-1} which has already been identified. We have to guarantee that in the given direction, the unvisited contour pixel must be swept earlier than p_{i-1}. Otherwise, the searching will make a sharp turn (180°) while certain feature has not been fully explored.

Here, we define a variable *pos* to store the position of pixel p_i relative to p_{i-1}. For the first contour pixel p_1, we can start the searching for p_2 from the left neighbor of p_1, because its left neighbor, if exists, must be a background pixel. Hence, the initial value of *pos* is set as 7. For $i \geq 1$, the searching for p_{i+1} will start at p_i's neighbor whose number is $(pos+5) \mod 8$. It can be proven that this pixel will always be a background pixel, and this selection will also make p_{i-1} be the last foreground pixel to be swept.

As the contour of a given object is a closed loop (a unit pixel width open curve can be viewed as a special case of closed loop), the contour propagation should stop if the next contour pixel found would be the same as p_1. Therefore, for each contour, the terminating criteria is set as: $p_i = p_1$. If p_1 is incident to more than one contours, e.g., two contours both have p_1 in common (can be seen as one self-intersected contour), it will be treated as a multi-contour case.

To tackle the multi-contour case, we first scan the whole image from the northwest corner to check the left neighbor for each foreground pixel. If a foreground pixel has a background left neighbor, it will be selected as a seed pixel. By doing this, we can ensure that each contour has at least one pixel being selected as seed. Apparently, if all the seeds have been propagated, no contours will be omitted. When all the pixels from the image have been checked, the seed pixels will be stored in a set. Then

the first seed pixel will serve as p_1 to trace a contour which contains p_1. During the contour tracing process, the set of seed pixel will be updated, and all seed pixels on this contour will be removed from the seed set. When a contour is finished, we will check if the set of seeds is empty. If yes, then all contours on the image have been explored. Otherwise, we will pick up another seed in the set to trace a new contour. This process continues until the set of seeds becomes empty.

As we always search neighborhood in a consistent direction, i.e., CCW, it is noteworthy that the generated contours will be orientation consistent. The outsider contour will always be CCW, whereas the inside contour is CW. The reason is as the following: To search contour pixel p_{i+1}, we start searching at a background neighbor of contour pixel p_i. If we find p_{i+1} at position n, at least position $(n-1) \mod 8$ is a background pixel. Neighbor $(n-1) \mod 8$ must be on the right hand side of $p_i \rightarrow p_{i+1}$. In other words, the right side of the contour should always be background (hollow), whereas the left hand side of the contour is foreground (solid). Therefore, outside contour traced is CCW, and inside contour traced is CW.

4.3 Image based Contour Offset

By the algorithm described in the previous session, all contours from an image can be traced, and these contours are single pixel width. Therefore, simply connecting the centers of contour pixel according to the connectivity preserved will provide the path for laser scanning. However, a shrunk binary image is still necessary for mask projection. In traditional method, when the segment contour is derived, we will conduct a negative offset by distance t in order to obtain a contracted area. In this session, we will generate the contracted binary image by the combining mathematical morphological and Boolean operation.

Assume B_0 is the original image generated by image based slicing algorithm, and its image contour C_0 can be traced by the algorithm described in previous session. Here, C_0 is the set of pixel indices for all the contour pixels, and it can also be represented by a binary image by simply setting all pixels in C_0 as foreground pixels and others as background. As we want to shrink the image B_0 by some amount, saying t_i (similar to offsetting by a negative distance t_i), a structuring element E_i will be created which is corresponding to t_i in Eqn.(4). Then we conduct a dilation on the contour image C_0 by the structuring element E_i. The shrunk image B_i can be calculated as the Boolean difference between B_0 and dilated C_0. This relationship can be represented as Eqn.(5), where "$\oplus$" denotes dilation, and "$-$" denotes Boolean difference.

$$B_i = B_0 - (C_0 \oplus E_i) \tag{5}$$

To conclude, in order to generate $n-1$ laser paths, n contractions have to be conducted. The final shrunk image B_n will serve as the input for mask projection, and the contours traced, $C_1, C_2 \ldots C_{n-1}$, will be used to direct the laser scanning. The pseud-code of image shrinking is shown as the following:

Algorithm 3 ImageShrinking

Input: a binary image B, structuring element E, integer n

Output: final shrunk image Bn, contour set D

01. $C \leftarrow$ ImageContourTracing(B)
02. $Cn \leftarrow \phi$
03. $Bn \leftarrow \phi$
04. $D \leftarrow \phi$
05. **for** $i \leftarrow 1$ **to** n **do**
06. $\quad Bn \leftarrow B - (C \oplus E[i])$
07. $\quad Cn \leftarrow$ ImageContourTracing(Bn)
08. $\quad D \leftarrow D \cup Cn$
09. **end for**
10. **return** Bn, D

The structuring element E_i is constructed according to the desired shrinkage amount t_i. If we use two integers u and v to represent the horizontal (column) and vertical (row) positions of a pixel, and assume the origin of the structuring element is located at the center of pixel (0,0), then all the pixels that satisfy $\sqrt{u^2+v^2} \leq \frac{t_i}{d}$ will be selected as a member of structuring element, where d is the pixel width. FIGURE 7 shows several examples of structuring elements. Obviously, the shape represented by the structuring element is an approximation of a circular area with radius t_i.

In this application, all structuring elements generated are symmetric about the origin, hence E_i and its reflection $\hat{E}_i$ are identical. Therefore, based on the definition of dilation [22], the operation represented by Eqn.(5) is equivalent to placing a circular disc with radius t_i on each inner contour pixel (center to center), and setting all pixels whose centers are covered by these discs as background. The resultant image will be the shrunk image B_i.

5 Efficiency Analysis and Implementation

The efficiency of the proposed algorithms for hybrid SLA system has been analyzed from time and space complexity perspectives, respectively. Then the proposed algorithm was implemented using the C++ programming language with Microsoft® Visual C++ compiler, and four 3D models were tested. The test

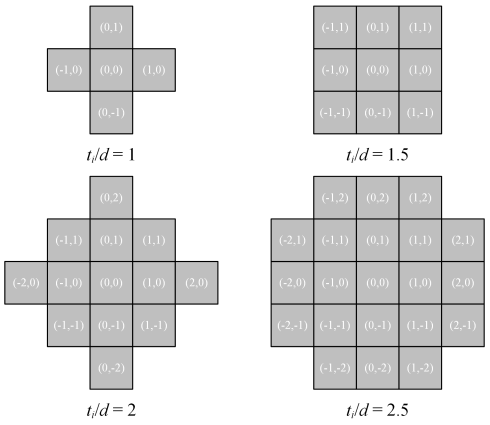


FIGURE 7. Examples of Structuring Element.

results were compared with those generated by traditional approach.

5.1 Efficiency Analysis

5.1.1 Time Complexity We first start to analyze the time complexity of image based slicing algorithm. The first step of proposed slicing algorithm is to convert STL file into point cloud. Assume the STL file of a 3D model has t triangle facets, reading vertex coordinates and normal vectors for all facets takes $O(t)$ time. For an arbitrary triangle facet, suppose that its projection in the plane Z=0 covers i incident pixels on average, so calculating and saving the point cloud for the model takes $O(t \times i)$ time. Therefore, the time complexity for converting an STL file into point cloud takes $O(t \times i)$ time. Since the total number of sampling points is $t \times i$, sorting all sampling points according to their Z-coordinates takes $O((t \times i)\log(t \times i))$ time. To generate the binary image set for the model from the point cloud, each sampling point needs to be traversed, so it also takes $O(t \times i)$ time. Here, we assume the amount of sampling points $t \times i$ is much greater than the required resolution and the number of total number of layers. Therefore, the overall time complexity for the proposed slicing algorithm is $O((t \times i)\log(t \times i))$.

The path planning process for a specific layer starts at identifying the seed pixel from a given image. Seed pixel identification needs to check each pixel, so it takes $O(m \times n)$ time. For a given contour pixel p_i, it needs to check all 8 neighbors at the most to find the next contour pixel p_{i+1}. If we assume that for a given image, there exists c contour pixels, then locating all contour pixels takes $O(c)$ time. Apparently, the total amount of contour pixels c is less than image resolution $m \times n$, so the entire contour tracing process takes $O(m \times n)$ time. Let $e = \lfloor \frac{t_i}{d} \rfloor$, constructing the structuring element takes $O(e^2)$ time. Therefore, generating one laser path and obtaining the resultant image take $O(c \times e^2)$ time. As both c and e^2 are less than $m \times n$, conducting one round image shrinking takes $O((m \times n)^2)$ time.

5.1.2 Space Complexity For the image based slicing algorithm, all $t \times i$ sampling points need to be stored, so the space complexity for proposed slicing algorithm is $O(t \times i)$. The path planning process has to save the pixel value for each pixel, hence, its space complexity is $O(m \times n)$.

5.2 Implementation and Experimental Result

Both the proposed method and traditional approach are implemented, and Clipper, which is an open source library for clipping and offsetting lines and polygons, was adopted to implement the traditional offsetting method. The same set of the 3D models (STL files) has been tested, and for each layer the output is an image for mask projection and a series of contours which serve as the laser path. The test environment is as follows:

64 bit Windows 10 Pro system laptop with Intel(R) Core(TM) i7-4600U, CPU @ 2.10GHz 2.69 GHz and 8GB RAM.

Table 1 shows model information, process parameters and processing time for all four models. The output comparison for a specific layer is shown in Table 2.

6 Conclusion and Future Work

In this paper we proposed a novel image based slicing and tool path planning method for the hybrid stereolithography system. It is essentially using the numeric method to solve an engineering problem. The details for two major components are discussed, and the proposed image shrinking approach can be viewed as a heuristic method for solid offsetting. Through theoretical analysis and experiment on test cases, efficiency of this method has been proven.

From Table 2 we can see that perceptible differences exist between the results generated by traditional approach and the proposed method. For example, for the Eiffel Tower and the ring knots, the 2nd laser paths generated by traditional approached have more loops than the laser paths obtained by the proposed method. By intuition these differences stem from using binary image with finite resolution to represent a 2D shape. If we can quantitatively correlate these differences with image resolution, this image based method will be more reliable.

REFERENCES

[1] Gibson, I., Rosen, D. W., and Stucker, B., 2010. *Additive Manufacturing Technologies: Rapid Prototyping to Direct Digital Manufacturing*. Springer, New York.

[2] Nakamoto, T., and Yamaguchi, K., 1996. "Consideration on the producing of high aspect ratio micro parts using uv

TABLE 1. Processing Time for Tested Models

Model	Number of Triangles	Size (L×W×H, mm)	Layer Thickness (mm)	Number of Layers	Image Resolution (L×W)	Image Size (L×W, mm)	Number of Offsets	Offset Distance (mm)	Time of Traditional Approach (s)	Time of Proposed Method (s)
Bell	210,500	$26.42 \times 26.11 \times 40.76$	0.1	407	1024×768	80×60	4	−0.078125	122.225	9.644
Eiffel Tower	138,530	$54.66 \times 54.66 \times 120.95$	0.5	241	1024×768	80×60	4	−0.078125	137.878	7.924
Hearing Aid	32,750	$49.99 \times 42.94 \times 39.96$	0.1	399	1024×768	80×60	4	−0.078125	46.682	9.666
Ring Knots	33,730	$22.58 \times 14.33 \times 23.15$	0.1	231	1024×768	80×60	4	−0.078125	42.536	4.540

TABLE 2. Result comparison for Tested Models

Model	2nd Laser Path by Traditional Approach	2nd Laser path by Proposed Method	Final Image for Mask Projection by Tradition Approach	Final Image for Mask Projection by Proposed Method
Bell	150th Layer	150th Layer	150th Layer	150th Layer
Eiffel Tower	36th Layer	36th Layer	36th Layer	36th Layer
Hearing Aid	160th Layer	160th Layer	160th Layer	160th Layer
Ring Knots	210th Layer	210th Layer	210th Layer	210th Layer

sensitive photopolymer". In Proceedings of 7th International Symposium on Micro Machine and Human Science, IEEE, pp. 53–58.

[3] Monneret, S., Loubere, V., and Corbel, S., 1999. "Microstereolithography using a dynamic mask generator and a non-coherent visible light source". In Proceedings of SPIE 3680 on Design, Test, and Microfabrication of MEMS and MOEMS, pp. 553–561.

[4] Stampfl, J., Fouad, H., Seidler, S., Liska, R., Schwager, F., Woesz, A., and Fratzl, P., 2004. "Fabrication and moulding of cellular materials by rapid prototyping". *International Journal of Materials and Product Technology,* ***21***(4), pp. 285–296.

[5] Xu, G., Zhao, W., Tang, Y., and Lu, B., 2011. "Novel stereolithography system for small size objects". *Rapid Prototyping Journal,* ***12***(1), pp. 12–17.

[6] Jiang, C.-P., 2010. "Accelerating fabrication speed in two-laser beam stereolithography system using adaptive crosshatch technique.". *International Journal of Advanced Manufacturing Technology,* ***50***(9-12), pp. 1003–1011.

[7] Kang, H.-W., Park, J. H., and Cho, D.-W., 2012. "A pixel based solidification model for projection based stereolithography technology". *Sensors and Actuators A: Physical,* ***178***, pp. 223–229.

[8] Bourell, D. L., Leu, M. C., and Rosen, D. W., 2009. Roadmap for Additive Manufacturing: Identifying the Future of Freeform Processing. Tech. rep., The University of Texas at Austin, Austin, TX.

[9] Zhou, C., Ye, H., and Zhang, F., 2015. "A novel low-cost stereolithography process based on vector scanning and mask projection for high-accuracy, high-speed, high-throughput, and large-area fabrication". *Journal of computing and information science in engineering,* ***15***(1), pp. 011003–1–011003–8.

[10] Choi, B. K., and Park, S. C., 1999. "A pair-wise offset algorithm for 2d point-sequence curve". *Computer-Aided Design,* ***31***(12), pp. 735–745.

[11] Park, S. C., and Shin, H., 2002. "Polygonal chain intersection". *Computers & Graphics,* ***26***(2), pp. 341–350.

[12] Chen, X., and McMains, S., 2005. "Polygon offsetting by computing winding numbers". In Proceedings of International Design Engineering Technical Conferences & Computers and Information Conference (IDETC/CIE 2005), ASME, pp. 1–11. Paper number DETC2005-85513.

[13] de Berg, M., Cheong, O., van Kreveld, M., and Overmars, M., 2008. *Computational Geometry: Algorithms and Application*. Springer, Berlin.

[14] Rock, S. J., and Wozny, M. J., 1991. "Utilizing topological information to increase scan vector generation efficiency". In Proceedings of International Solid Freeform Fabrication Symposium, pp. 1–9.

[15] Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., and Lévy, B., 2010. *Polygon Mesh Processing*. A K Peters, Natick, MA.

[16] Hughes, J. F., van Dam, A., McGuire, M., Sklar, D. F., Foley, J. D., Feiner, S. K., and Akeley, K., 2013. *Computer Graphics: Priciples and Practices*. Addison-Wesley, Upper Saddle River, NJ.

[17] Huang, P., Wang, C. C. L., and Chen, Y., 2013. "Intersection-free and topologically faithful slicing of implicit solid". *Journal of computing and information science in engineering,* ***13***(2), pp. 021009–1–021009–13.

[18] Stelldinger, P., Latecki, L. J., and Siqueira, M., 2007. "Topological equivalence between a 3d object and the reconstruction of its digital image". *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* ***29***(1), pp. 126–140.

[19] Rossignac, J. R., and Requicha, A. A. G., 1986. "Offsetting operations in solid modelling". *Computer Aided Geometric Design,* ***3***(2), pp. 129–148.

[20] Nadler, S. B., 1978. *Hyperspaces of Sets: A Text with Research Questions*. Marcel Dekker, New York.

[21] Sonka, M., Hlavac, V., and Boyle, R., 2008. *Image Processing, Analysis, and Machine vision*. Thomson Learning, Toronto, Ont.

[22] Gonzalez, R. C., and Woods, R. E., 2007. *Digital Image Processing*. Prentice Hall, Upper Saddle River, NJ.

[23] Engedy, I., and Horváth, G., 2009. "A global, camera-based mobile robot localization". In Proceedings of 10th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics, pp. 217–228.

[24] Huang, P., Wang, C. C. L., and Chen, Y., 2014. *Algorithms for Layered Manufacturing in Image Space*, Vol. 1 of *Advances in Computers and Information in Engineering Research*. ASME, Chap. 15.

[25] C. Sun, N.Fang, D. M. W., and Zhang, X., 2005. "Projection micro-stereolithography using digital micro-mirror dynamic mask". *Sensors and Actuators A: Physical,* ***121***(1), pp. 113–120.

[26] Vatani, M., Rahimi, A. R., Brazandeh, F., and Nezhad, A. S., 2009. "An enhanced slicing algorithm using nearest distance analysis for layer manufacturing". In Proceedings of World Academy of Science, Engineering and Technology, Vol. 37, pp. 721–726.