Contents

1

Algorithms for Layered Manufacturing in Image Space

Pu Huang

Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong

Charlie C. L. Wang

Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong

Yong Chen

Epstein Department of Industrial and Systems Engineering, University of Southern California, USA

1.1 Introduction

Layered manufacturing plays an important role in industry. It fabricates an input 3-D model by adding material in the layer-by-layer pattern. Layered manufacturing is widely used in applications such as biomedical engineering, aerospace industry, and automotive industry. Most layered manufacturing processes require the input model to be represented in STereoLithography (STL) format, which defines the object as a raw unstructured triangulated surface by the unit normals and vertices (ordered by the right-hand rule) of the triangles using a 3-D Cartesian coordinate system. A set of parallel planes are used to intersect with the triangulated surface of the object as the slicing strategy, and the intersection contours are traced on each slice. Non-manifold features like self-intersection, degenerated triangles, or gaps will always lead to problematic contours. The existing commercial software packages implement heuristic rules to deal with such problematic contours. For example, cutting the singular point at which the contour has a self-intersection (as shown in Figure.1.1). However, these heuristic rules do not fundamentally solve the problem. Consequently, incorrect part region classification will lead to incorrectly fabricated layers, either with unwanted gaps (see Figure.1.2 from [11]) or membranes. The models shown in Figure.1.2 are fabricated by fused deposition modeling (FDM).

In this chapter, we investigate robust and efficient approaches for layered manufacturing process planning directly applied on an implicit solid, which is reconstructed from point cloud or volumetric images in the reverse engineering context. Approaches are developed in image space for the two

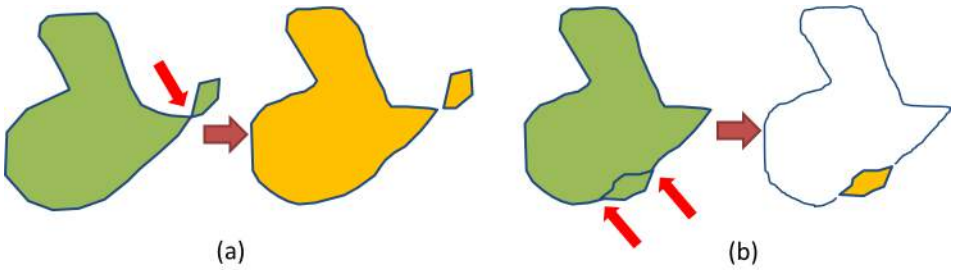

FIGURE 1.1: Heuristic used by commercial software to deal with problematic contour. The self-intersected contour will be cut at the intersection point to form separated contour(s) as a result. Only the regions surrounded by closed contours are classified as parts inside the input model – i.e., the yellow regions in this figure. However, this heuristic rule does not succeed in all cases: (a) a successful case and (b) a failed case that will lead to a layer with most of its part material missing.

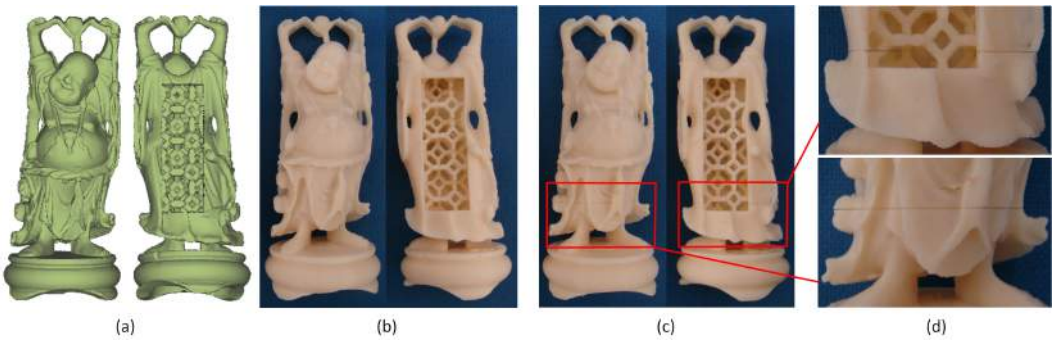

FIGURE 1.2: Incorrect contours generated by slicing a given model will produce a model with unwanted gaps (and/or membranes) in layered manufacturing: (a) a given Buddha model in implicit representation [actually layered depth-normal images (LDNI) [4]], (b) the correct model fabricated from contours generated by presented approach, (c) the problematic object fabricated by slicing the locally self-intersected polygonal model extracted from (a) using a variation [29] of dual-contouring [13], and (d) a magnified view of the incorrect layer.

main steps in process planning: support generation and slicing. The effectiveness of the presented approach is demonstrated on the actual fabricated parts by layered manufacturing. In particular, the chapter makes the following contributions:

- For a given implicit solid and its slicing data $\{l_i\}$, we generate a reliable general support region on a binary image, and optionally, its boundary contour. The self-support area of the part is excluded as much as possible to save support material. Proofs of these properties are given (see Section 1.4).

- Compared with conventional polygon-based region subtraction, the presented approach is based on a binary set and uses binary Boolean operations and integer arithmetic. This makes it more robust and easy to implement (see Section 1.4 for details).

- For an r-regular solid represented by the implicit indicator function (i.e., "$-$" for inside and "$+$" for outside), a method is presented to generate contours that are topologically faithful, self-intersection free, and with the shape approximation error controlled. As a direct slicing

approach, it is efficient in computation and memory usage. Only the information on a particular slicing plane is involved. This is different from those techniques which first polygonize an implicit solid into B-reps and then generate contours from B-rep ([6]).

A flow diagram of sections in this chapter is given in Fig.1.3. We assume the input model H satisfies the r-regular property. The sampling resolution is chosen based on the rapid prototype machine specification in order to fabricate a topologically correct part. After the binary image sampling, all the subsequent processes are independently for each slicing image l_i. Region subtraction is used to calculate the support structure region if necessary. Note that we use S_i to represent the general support region directly produced by the region subtraction method. It can be further processed to produce specific support region for specific layered manufacturing processes like FDM and Stereolithography (SLA). At last, topologically faithful contours can be generated for both part region and support structure region if needed. The rest of this chapter is organized as follows. Section 1.2 provides a literature review of the related work in layered manufacturing. Section 1.3 describes how to choose the binary image sampling resolution based on machine specification. Support structure generation based on region subtraction is discussed in Section 1.4 and several test results are provided specifically for support generation. Section 1.5 describes the topologically faithful contour generation for any region on a binary image. Several results are shown to highlight the advantage of the proposed contour generation method. Section 1.6 provides the conclusion and also discusses the future work.

1.2 Literature Review

This section reviews the related work in layered manufacturing.

1.2.1 Slicing-based Support Generation

Most layered manufacturing processes generate a support structure based on the respective STL model [20]. The most common method for generating a support structure is to determine whether the angle between an overhang facet and horizontal plane is big enough to withstand the weight of material. This can be easily performed by computing the dot product of facet normal vector and horizontal vector ([15, 30, 25, 12]). Chalasani et al. [3] use 2-D slicing data to determine the support region in each layer for FDM. This method calculates the shadow region between adjacent layers, and checks whether the overhang in the shadow is big enough for adding support. Recently, Qian et al. present an slicing-based support generation for SLA in [21]. Their approach calculates the difference between two adjacent layers as shadow region. After that, they detect the self-support region by offsetting the lower slicing contour for a distance that is a material-dependent threshold, and subtract the resultant contour from the shadow region. However, this approach may cause some area missing support, which will be explained and solved in Section 1.4.2.

We recently published a paper which use binary image to perform implicit solid slicing for layered manufacturing [11]. The binary image has the advantage of robustness for 2-D solid modeling operations like Boolean, Offsetting, and Minkowski Sum. Moreover, the morphological operations on binary image makes it more useful to perform two-dimensional geometric processing [26, 10]. Couprie and Bertrand [7] introduced a morphology-based sequential filter for smoothing 2-D and 3-D objects. In their paper, they use the distance map computed by *Euclidean distance mapping* (UDM) ([8]) to control the radius of the element structure for morphological operations. This inspired us in developing the conservative growing-swallow method described in Section 1.4.2.

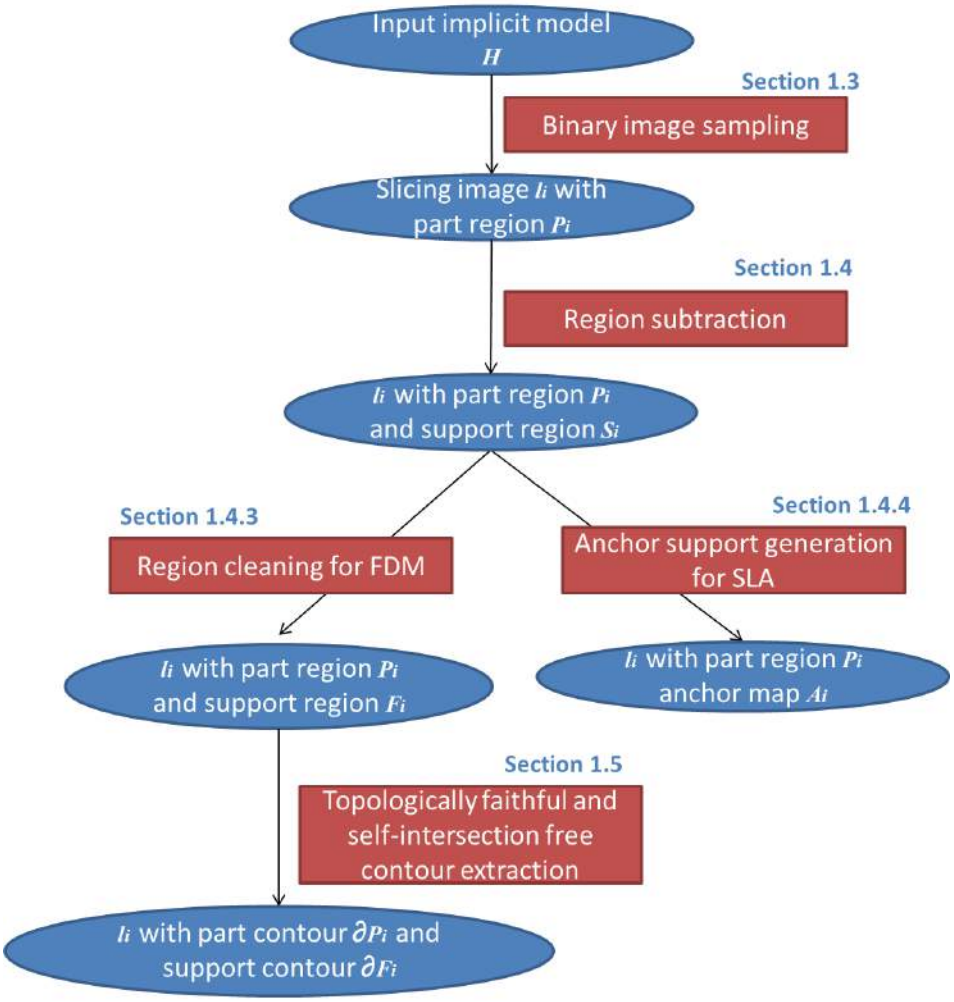

FIGURE 1.3: Flow chart of the chapter.

1.2.2 Direct Slicing on Implicit Solid

The problem of directly slicing an implicit solid involves the computation of the intersection between the solid and a plane. According to the review in [19], all available methods can be classified into two categories: analytical and numerical. Analytical methods find precise intersection points by solving polynomial equations derived from implicit surface representation (e.g., [9]). However, these methods can only be applied to algebraic surfaces and the computing speed in general is slow. Numerical strategies like subdivision [17] and marching [2, 1] do not require precise analytical representation of surface boundary. Subdivision methods intersect a tessellated piecewise linear approximation of the given implicit surface with a plane, which have the problem that some small intersection loops will be missed if the subdivision stops at an improper level. Marching-based methods (e.g. [31]) always start from an initial point, and then proceed to march along a curve, but they suffer from robustness problem at critical points of contours (i.e., the point where two loops join into one). Tracing-based contour generation algorithms like [31] may also miss some small loops. In a follow-up work in [22], the authors divide the *moving least-square* (MLS) surfaces into

several slabs with each slab having the same topology, then use a tracing strategy to generate contours for each slab separately. However, unlike the presented approach, their method is specialized for MLS surface and they do not provide rigorous proof of the self-intersection free property and the topological faithfulness as the presented recent approach in [11].

1.3 Sampling and Accuracy

The appropriate sampling rate to guarantee the extraction of topologically faithful contours is analyzed by briefly reviewing the relevant definitions and theorems given in [24].

Definition 1. A solid $H \subset \Re^3$ is called r-regular if, for each point $\mathbf{p} \in \partial H$, there exist two osculating open balls of radius r to ∂H at $\mathbf{p}$ such that one lies entirely in H and the other lies entirely outside.

Theorem 1. For an r-regular solid H, the boundary surface, M, generated by a *topology preserving method* on cubic grids is r-homeomorphic to the exact surface boundary, ∂H, if the cube width r' of grids satisfies $\sqrt{3}r' < r$.

The above definition and theorem are derived from Definition 1 and Theorem 16 of [24], which is the foundation of binary image sampling and topologically faithful contouring. The term r-homeomorphic means that the reconstructed boundary surface M can be deformed into ∂H within distance r by morphology operation without changing topology. Note that r' here is different from r' used in [24]. Details about the *topology preserving methods* can also be found in [24].

For a given implicit solid $H = \{\mathbf{p} | f(\mathbf{p}) \leq 0, \forall \mathbf{p} \in \Re^3\}$ and a slicing plane P, a contour $C = M \cap P$ is defined as a *topologically faithful* contour when M is a surface r-homeomorphic to the exact surface boundary, ∂H, of H. For an implicit solid H and a set of slicing data $\{l_i\}$ which are binary images with pixel width r', support regions S_i are generated for layer l_i. Each slicing layer l_i contains part region P_i on the binary image. A support region S_i for l_i is defined as a *reliable* support region when $P_i \bigcup S_i \bigcup \Delta(P_{i+1}, P_i, d) \supseteq S_{i+1} \bigcup P_{i+1}$ where P_{i+1}, S_{i+1} are the part region and support region, respectively, on l_i's above adjacent layer l_{i+1} and $\Delta(P_{i+1}, P_i, d)$ represents the region in P_{i+1} that does not need support due to the self-support effect. d is a given threshold for self-support region detection for any two adjacent layers. Essentially, S_i can be considered as reliable if the support region S_i plus the part region P_i on l_i is larger than the region needed to be supported on l_{i+1}, which is the support region S_{i+1} plus the part region P_{i+1} minus the self-supported region $\Delta(P_{i+1}, P_i, d)$. For example, for the case shown in Fig.1.8(a), the yellow region represents the self-support region $\Delta(P_{i+1}, P_i, d)$ in P_{i+1}. Assume there is no support region S_{i+1} on layer l_{i+1}, the support region S_i needs to satisfy that $S_i \bigcup P_i$ is larger than $P_{i+1} - \Delta(P_{i+1}, P_i, d)$ in order to be reliable. The relationship can be represented as $P_i \bigcup S_i \supseteq S_{i+1} \bigcup P_{i+1} - \Delta(P_{i+1}, P_i, d)$ and after simple manipulation, we can get $P_i \bigcup S_i \bigcup \Delta(P_{i+1}, P_i, d) \supseteq S_{i+1} \bigcup P_{i+1}$. The support region F_i generated specifically for FDM will be shown to be reliable in Section 1.4.3. On binary image, the regions like P_i are discretized into pixels and their corresponding boundaries are represented by contours consisting of line segments. In this chapter, a method is developed to compute region S_i on a binary image which satisfy (1) the region S_i is reliable and (2) the area of S_i is reduced as much as possible by excluding the self-support area to save supporting material.

1.3.1 Sampling

To generate topologically faithful contours so that a model homeomorphic to the exact boundary of H can be fabricated from them, the 2D solid, $H \cap P$, is sampled into a binary image I and

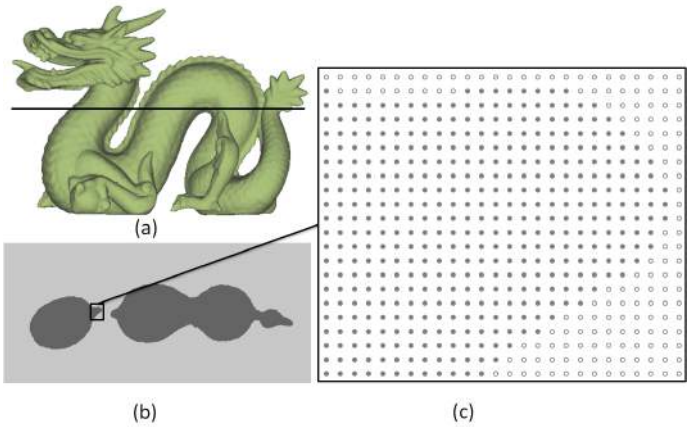

FIGURE 1.4: A binary image sampled from a Dragon solid: (a) the solid H is intersected by a slicing plane P, (b) the region of $H \cap P$ on the binary image, and (c) a zoom-view of the binary image, where the nodes inside $H \cap P$ are displayed in solid dots while outside nodes are shown in hollow.

then generates a contour $\tilde{C}$ from I. The following propositions are derived to guarantee that $\tilde{C}$ is topologically faithful.

Proposition 1. For a r-regular solid H, the contour generated by a *topology preserving method* on square grids is topologically faithful if the width r' of the squares satisfies $\sqrt{3}r' < r$.

Following this proposition, a binary image I is sampled from $H \cap P$ on the slicing plane P with the pixel distance r'. The first step is to rotate the given solid H into the coordinate system with x-o-y plane parallel to the slicing plane P. Then, the dimension of the binary image can be determined by the intersection between P and the bounding box of H. The resolution is defined according to the value of r'. Figure.1.4 shows an example of the binary image sampled from a solid of Dragon model. Note that the sampling of binary image is not limited to the planes perpendicular to the major axes. As long as the in/out membership tests can be efficiently conducted on the given solid H, the binary image can be generated on a slicing plane in any orientation.

After obtaining a binary image, the marching square method introduced in [18] is used to generate an approximate contour $\tilde{C}^0$ that is topologically faithful. Defining the edge on a square grid with different in/out status on its two endpoints as a *stick*, the contour $\tilde{C}^0$ can be formed by the line segments linking the middle points of sticks in all grids. Note that, in the rest of this section, endpoints are not included when we refer to a *stick* (i.e., it is defined on an open interval). Figure.1.5 shows the lookup table we used in marching square method to construct contour edges.

Proposition 2. The contour generated by using the lookup table shown in Fig.1.5 is topologically faithful.

The lookup table is derived from the topology preserving contouring method (e.g., Ball Union) in [24] by considering the boundary of the 3-D grid in [24] as the planar square grid (defined in binary image) here. Contour edges E are built up based on the in/out classification of the four grid nodes (pixels on a binary image).

1.3.2 r-Regularity and Accuracy in Layered Manufacturing

The contour generated by the above method is ensured to be on a surface homeomorphic to the boundary of a r-regular solid H. Although not all implicit solids are r-regular, this assumption

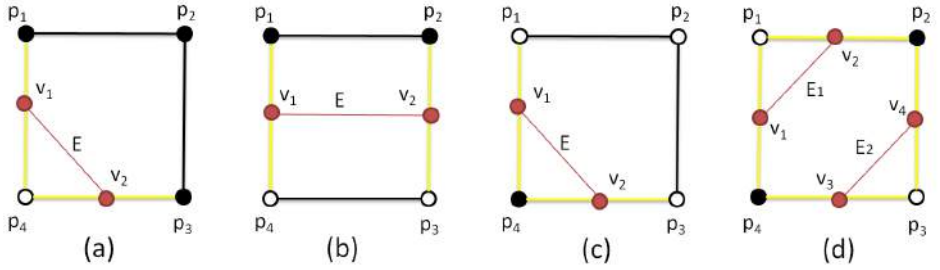

FIGURE 1.5: Lookup table for the marching square method with topology preserved. Sticks are in yellow. Sampling nodes inside the solid H are shown in black while the outside nodes are displayed in white. The contour edges linking sticks are labeled as E. Taking into consideration the rotational symmetry, the in/out configurations of the four corner nodes for a grid containing sticks can be classified into the listed four cases shown in (a), (b), (c), and (d). For each grid with sticks, its corners' in/out configuration can be found among the four cases and the contour edge can be created accordingly.

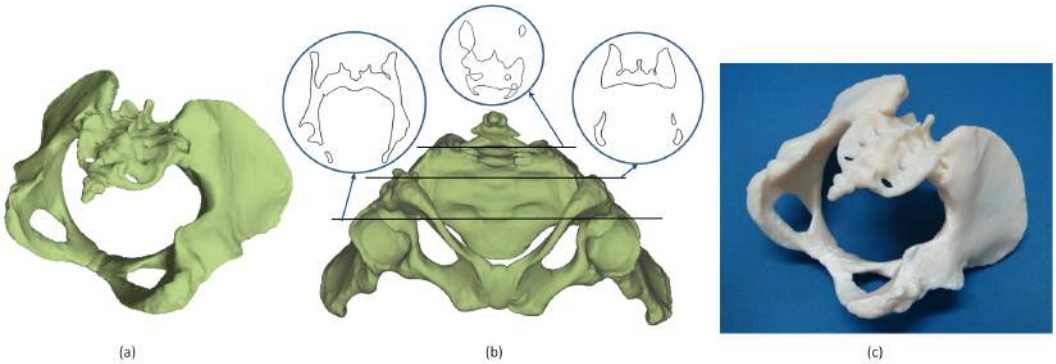

FIGURE 1.6: An example model fabricated from the contours generated by the proposed method with $r' = 0.101$ mm: (a) the given Donna model in the LDNI representation, (b) the sliced contours of respective layers at 58.42, 76.2, and 92.96 mm heights, and (c) the resultant model fabricated by FDM.

is reasonable for the models to be fabricated by layered manufacturing. The value of r actually relates to the smallest component that can be fabricated by an rapid prototyping machine (e.g., the diameter of plastic filaments on an FDM machine as well as the finest position that can be provided by motion controller). Most commercial x-y positioning systems used in rapid prototyping can achieve the precision of 10^{-2} mm. The diameter of plastic filaments is usually greater than 10^{-1} mm. Moreover, for an implicit solid represented by LDNI [4, 29], the accuracy of the solid is also limited by the resolution of LDNI. It is meaningless to make the grid width of binary image smaller than that of LDNI. As an example, for the case of the Donna model shown in Fig. 1.6, we choose $r' = 0.101$ mm to generate the binary images, slightly larger than that of LDNI (0.099 mm). Usually, selecting r' with a value no larger than 0.125 mm is good enough for models fabricated by FDM. In addition, all solid models can be processed to r-regular using the techniques like morphological operators.

1.4 Reliable and Robust Region Subtraction for Support Generation

The support structure can be used to support overhangs, keep stability for the part to prevent from tipping over during the fabrication, support large flat walls, and prevent excessive shrinkage. Depending on the layered manufacturing process and corresponding process planning method, the support structure can be either generated directly on the STL model before slicing or calculated on each layer from part slices. In the proposed approach, generating the support structure from part slices is preferred because directly using the slicing data eliminates the handling of the flaws in the STL model. With the road-width, material, and other parameters selected, the self-support feature can also be detected on each layer in order to save support material and cleaning time.

For two adjacent layers, region subtraction represents the subtracting of the lower layer part region from the upper layer regions (including part region and support region) that need support. The output of region subtraction is the support region for the lower layer. Conventional approaches use polygonal offsetting and Boolean operations to implement region subtraction. These polygonal operators are based on numerical computation, which is not as robust as binary discrete operation, especially for some extreme cases like tangential contact regions. The robustness problem of polygonal operators will lead to slow processing, incorrect support region generation, or even program crash. On the other hand, in order to detect the self-supported regions, one needs to offset the polygon of part slice, which may introduce self-intersection. For polygonal operations, this method usually suffers from numerical pruning in algorithms. This leads to non-robust and tedious implementation. Although the presented approach has a layer-wise context, it starts from the discrete slicing data which are actually a set of binary images. Figure 1.7 shows two physical parts processed using the presented direct slicing and layer-wise support generation method, and the physical models are fabricated by FDM and SLA, respectively.

Recall that for a given implicit solid H and a set of slices $\{l_i\}$ which are binary images with pixel widths r', support regions S_i are generated on l_i satisfying the reliability property, which is defined in Section 1.3. The region subtraction approach consists of three steps. Suppose l_i is going to be processed, first, using boolean operation subtraction to compute $\Psi_i = P_{i+1} - P_i$ in which Ψ_i is called *shadow for layer* l_i. Ψ_i is the potential region that will be processed to generate the support region. Next, an intermediate region $\widetilde{S_i}$ is computed by a growing-swallow method which conservatively excludes the *self-support region* $\Delta(P_{i+1}, P_i, d)$ from Ψ_i. The self-support region is a part of the region in P_{i+1} on layer l_{i+1} that does not need support on layer l_i due to the self-support effect (i.e., the lower layer part region P_i will provide support effect around its boundary for the upper layer part region P_{i+1}). It is defined as the area of shadow region which satisfies two conditions. The first is to fall in the region $\Gamma(P_i, d) - P_i$. The second is to have a connection path to $P_i \cap P_{i+1}$ and the path entirely lies in the region $\Gamma(P_i, d) - P_i$ (see Figure.1.8). $\Gamma(P_i, d)$ indicates the outward offset region of P_i for distance d. Note that in [21], the self-support region is excluded by subtracting $\Gamma(P_i, d) - P_i$ from Ψ_i. This will cause some non-self-support region classified as self-support region [the green region in Figure.1.8(b)], and thus not supported at all. The growing-swallow method introduced in this chapter will not suffer from this problem and can detect any self-support region correctly. Finally, the support region S_{i+1} is projected from the above layer to current layer and computes $S_{i+1} \bigcup \widetilde{S_i} - P_i$ as S_i. Optionally, a self-intersection free and error-bounded contour extraction method is employed to generate contour from S_i. The region subtraction is general for the layered manufacturing process which generates the support structure based on slicing layers. In this section, we also introduce the support generation method employing the region subtraction technique for both FDM and SLA.

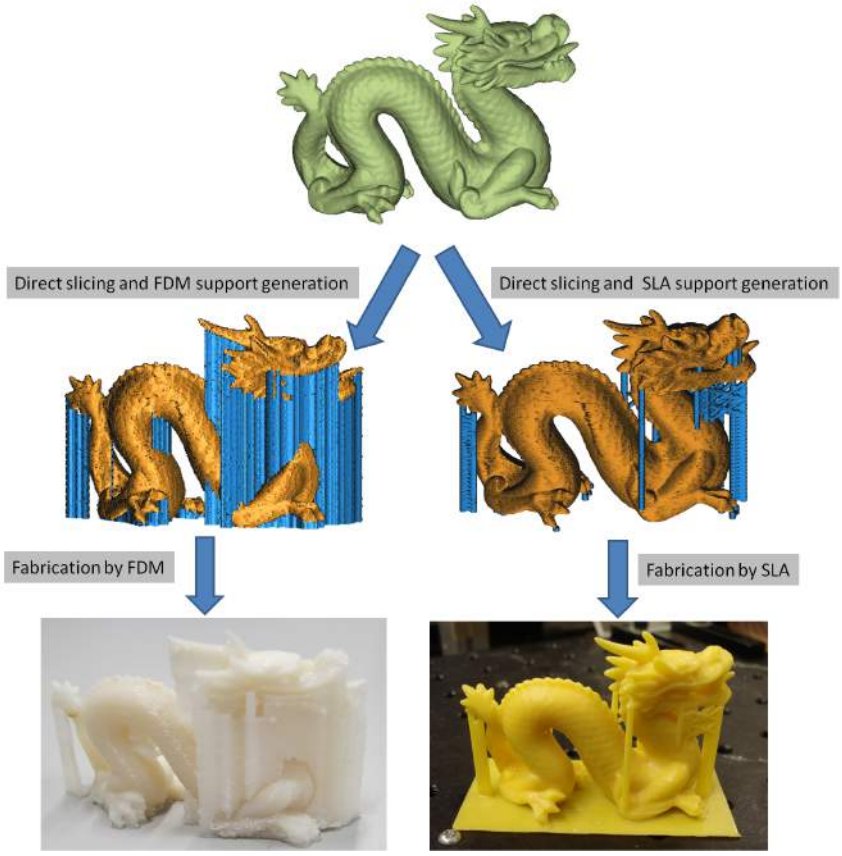

FIGURE 1.7: Physical parts demonstration for slicing and support generation using FDM, and SLA. The input dragon model is represented by LDNI ($1024 \times 1024 \times 1024$) and with size $50.80 \times 22.86 \times 35.81$ mm^3 and $76.2 \times 34.29 \times 53.85$ mm^3 for FDM and SLA respectively. The grid width of binary image and the layer thickness is set as 0.135 mm and 0.101 mm for SLA, and 0.076 inch and 0.177 mm for FDM. The values of these parameters are selected according to the specification of RP machines.

1.4.1 Preliminary

We review some basic notations of mathematical representation on a binary image, which will be used in this chapter. We denote the set of relative integers by $\mathbb{Z}$ and the discrete 2-D space by $\mathbb{Z}^2$. A point $x \in \mathbb{Z}^2$ is represented by (x_1, x_2) with $x_1, x_2 \in \mathbb{Z}$. For $x \in \mathbb{Z}^2, r \in \mathbb{R}$, we use $\mathscr{C}_d(x)$ to represent a circle with radius d and centered at x, with the definition $\mathscr{C}_d(x) = \{y \in \mathbb{Z}^2, Dist(x,y) \leq d\}$, where $Dist(x,y)$ is the distance between two points in $\mathbb{Z}^2$ and is defined as $Dist(x,y) = \sqrt{(x_1-y_1)^2+(x_2-y_2)^2}$. The morphological operation *dilation* by $\mathscr{C}_d(x)$ is defined as $\gamma_d(X) = \bigcup_{x \in X} \mathscr{C}_d(x)$. X is a subset in $\mathbb{Z}^2$ on which the dilation operation γ_d is applied. The circle $\mathscr{C}_d(x)$ is called the *structure element* of the dilation (see Fig.1.9). For a morphological operator α, its dual operator $*\alpha$ can be defined by: $\forall X \in \mathbb{Z}^2, *\alpha(X) = \overline{\alpha(\overline{X})}$ in which $\overline{X}$ represents the complementary set of X in $\mathbb{Z}^2$. The dual operator for dilation is called *erosion*, it is defined as $\varepsilon_d = *\gamma_d$. The *opening* operation is defined as an erosion followed by a dilation and can be denoted by $\zeta_d = \gamma_d \bigcirc \varepsilon_d$. Its dual operator is *closing* and can be denoted by $\eta_d = \varepsilon_d \bigcirc \gamma_d$. For any given d, the opening operation ζ_d satisfies the following three important properties:

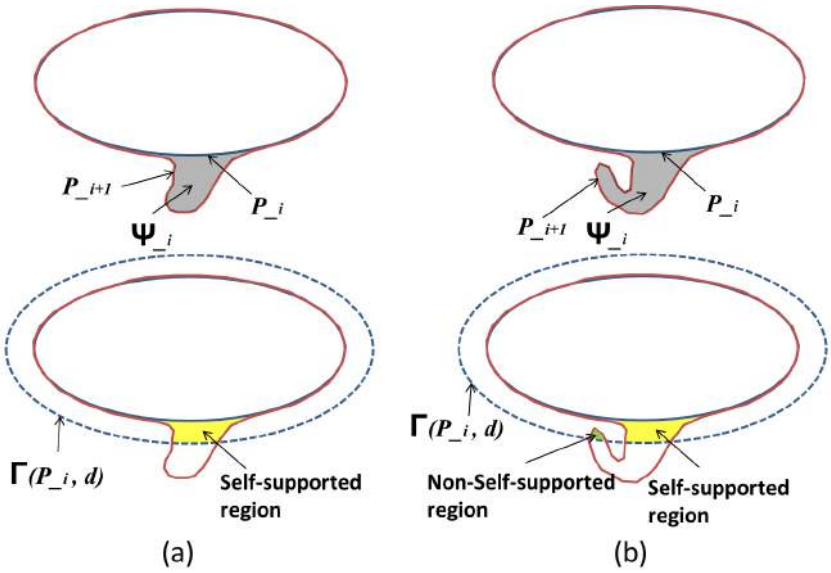

FIGURE 1.8: The demonstration of shadow region (gray) and self-support region (yellow) for two different cases: (a) shadow region and self-support region for two adjacent layers, case 1 (b) shadow region and self-support region for two adjacent layers, Case 2.

1. *increasing*, $\forall X, Y$ subset of $\mathbb{P}$, $X \subseteq Y \Rightarrow \zeta_d(X) \subseteq \zeta_d(Y)$,

2. *anti-extensive*, $\forall X \subseteq \mathbb{P}, \zeta_d(X) \subseteq X$ and,

3. *idempotent*, $\forall X \subseteq \mathbb{P}, \zeta_d(\zeta_d(X)) = \zeta_d(X)$.

Similarly, the closing operation η_d satisfies increasing, extensive ($\forall X \subseteq \mathbb{Z}^2, \eta_d(X) \supseteq X$) ,and idempotent.

1.4.2 Region Subtraction

We perform region subtraction for each layer in a top-down manner starting from the second layer on top (since the first layer does not need any support). For a specific layer l_i, a conservative growing-swallow method is adopted based on the error-bounded UDM to implement region subtraction.

Binary Image Grid-width and Self-support Feature Threshold

Since the self-support area detection is performed on a binary image, the image grid width r' should be smaller than the self-support feature threshold t in order to obtain enough accuracy. The part that can be fabricated by typical rapid prototyping has a precision limitation that depends on the smallest component that can be achieved by the system. Based on the calibration result reported in [5], for FDM, the road width of part material has the magnitude of 10^{-1} mm. This implies that the self-support feature threshold t should be in the same magnitude. According to [11], the binary image grid width r' should be set no larger than 0.125 mm in order to generate topologically faithful slicing contours. This indicates that for FDM, the value of r' is usually smaller than t. For SLA, the threshold t can be determined by physical experiments and the value of r' can be adjusted accordingly to achieve accurate detection results.

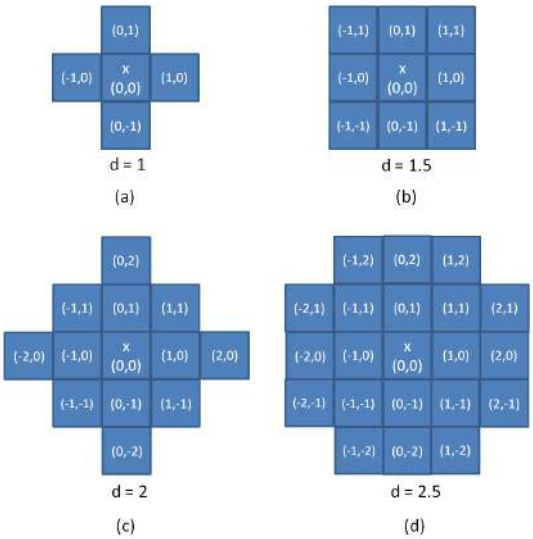

FIGURE 1.9: Different element structure $\mathscr{C}_d(x)$ for different values of d.

Conservative Growing-swallow Method

Based on the self-support region $\Delta(P_{i+1},P_i,d)$ defined above, a conservative growing-swallow method is developed to find $\Delta(P_{i+1},P_i,d)$. Starting from a shadow region Ψ_i, an intersection region $P_i \cap P_{i+1}$ and a pre-computed outward offset region $\Gamma(P_i,d)-P_i$, we initialize a region A as $P_i \cap P_{i+1}$ for temporary use. Two steps are applied iteratively to the image. The first step involves performing a dilation on A and labeling the newly grown pixels by the dilation. The second step involves integrating all the pixels satisfying three conditions: belonging to newly grown pixels, belonging to region $\Gamma(P_i,d)-P_i$, and belonging to the shadow region Ψ_i. The iteration does not stop until there is no more pixel satisfying all the above three conditions after dilation in step 1. The remaining support region $\widetilde{S_i}$ is the result and $\Delta(P_{i+1},P_i,d)$ can be easily calculated as $\Psi_i - \widetilde{S_i}$. Figure 1.10 provides an illustration of the working principle of growing-swallow algorithm. Note that in the example shown in Fig.1.10, there is no pixel belonging to $D \leftarrow \{x \in \mathbb{Z}^2, |\ Distmap(x)\ | \leq t\}$, i.e., $C \neq \emptyset$ after three iterations.

The presented growing-swallow method adopts the error-bounded UDM introduced in [8] to calculate the outward offset region $\Gamma(P_i,d)-P_i$. For a given image with black set $B_i \subseteq \mathbb{Z}^2$ (in this section, let's assume the black set is the region to be filled with material in layer manufacturing and the white set is empty), we use the provided four-point sequential UDM algorithm ($4SED(B_i)$) to compute a distance map $Distmap$ with respect to black set B_i. $Distmap$ satisfies $Distmap(x) = (u,v)$ with u and v being the projective distance value on u and v axes, respectively, to the closest pixel belonging to B_i. For $x \in B_i$, $(u,v) = (0,0)$. We also define $|Distmap(x)| = \sqrt{u^2+v^2}$. The psuedo-code of the growing-swallow method is as follows.

Note that the $4SED$ algorithm generates a distance map which is error-free except for very sparsely scattered pixels, and guarantees an error bound of 0.29r'. The formal error analysis can be found in [8]. Because error can only manifest from over-estimating the real distance, excluding the self-support region makes the presented growing-swallow method conservative. In other words, those regions being excluded can be guaranteed as self-support regions, but not all the self-support regions are guaranteed to be excluded. For the dilation operation $\gamma_{1.5r'}(A)$, $1.5r'$ is used as the radius of structure element, and this gives us the structure element shown in Fig.1.9(b). Based on the physical property of layer manufacturing that the self-support effect can occur all around a piece of

Algorithm 1 GrowingSwallow

Require: shadow region Ψ_i, part region P_i and P_{i+1}, self-support feature threshold t
Ensure: support region $\widetilde{S}_i$

```
1: S̃_i ← Ψ_i ;
2: Initialize A ← P_i ∩ P_{i+1}, B ← A, C ← B;
3: Distmap ← 4SED(P_i);
4: Initialize D ← {x ∈ Z^2, | Distmap(x) | ≤ t};
5: while C ≠ ∅ do
6:     B ← γ_1.5(A);
7:     C ← (B − A) ∩ D ∩ S̃_i;
8:     A ← C ∪ A;
9:     S̃_i ← S̃_i − C;
10: end while
11: return S̃_i;
```

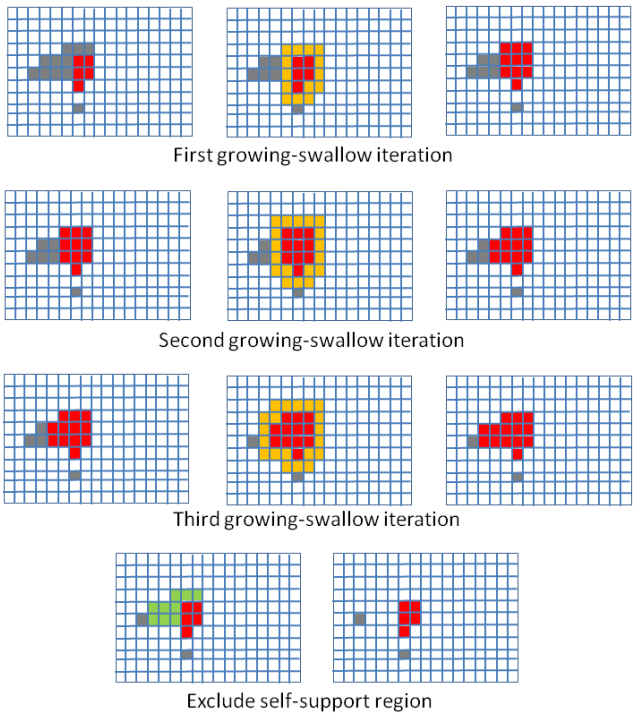

FIGURE 1.10: Growing-swallow algorithm illustration. Note that in this example, we assumed that there is no pixel belonging to $D \leftarrow \{x \in \mathbb{Z}^2, | \, Distmap(x) \, | \leq t\}$, i.e., $C \neq \emptyset$ after three iterations. The red pixels indicate intersection region $P_i \cap P_{i+1}$ in the above algorithm, gray pixels indicate $\widetilde{S}_i$, and green pixels indicate self-support region.

material, the element structure of radius $1.5r'$ provides better approximation for self-support effect than that of radius r'.

The following Algorithm 2 shows the proposed region subtraction technique.

All the operations like Boolean, morphological, and image scanning required in the proposed region subtraction technique are applied on binary sets and using Boolean or integer arithmetic, this makes the presented approach very robust, highly parallelizable, and easy to implement.

Algorithm 2 RegionSubtraction

Require: image l_i with part region P_i for current layer, image l_{i+1} with part region P_{i+1} and support region S_{i+1}, self-support feature threshold t

Ensure: support region S_i on image l_i

1: $\Psi_i \leftarrow P_{i+1} - P_i$;
2: $\widetilde{S}_i \leftarrow GrowingSwallow(\Psi_i, P_i, P_{i+1}, t)$;
3: $S_i \leftarrow \widetilde{S}_i \cup S_{i+1} - P_i$;
4: **return** S_i;

Proposition 3. The support region S_i generated by algorithm *RegionSubtraction* satisfies the reliable property with respect to the above layer l_{i+1}.

Proof. According to algorithm *RegionSubtraction*, pixels in P_{i+1} can always be supported by at most three kinds of supporting layout. The first kind is to be supported by P_i. For those that cannot be supported by P_i (pixels in shadow region Ψ_i), they are either supported by $\Delta(P_{i+1}, P_i, d)$ in self-support manner [pixels excluded by $GrowingSwallow(\Psi_i, P_i, P_{i+1}, d)$] or supported by $\widetilde{S}_i$. Because $\widetilde{S}_i \subseteq S_i$, P_{i+1} can be considered as being supported by $P_i \bigcup S_i \bigcup \Delta(P_{i+1}, P_i, d)$, i.e., $P_{i+1} \subseteq P_i \bigcup S_i \bigcup \Delta(P_{i+1}, P_i, d)$. For pixel in S_{i+1}, because of $S_i \leftarrow \widetilde{S}_i \cup S_{i+1} - P_i$, $S_{i+1} \subseteq S_i \cup P_i$. Hence, we have $P_i \bigcup S_i \bigcup \Delta(P_{i+1}, P_i, d) \supseteq S_{i+1} \bigcup P_{i+1}$, and the reliable property is satisfied.

□

1.4.3 Region Cleaning Technique for FDM

The proposed region subtraction technique can compute a reliable support region S_i, which serves as a core technique for most layer manufacturing support generation. FDM process may generate its own support region F_i based on S_i by considering its own system specification. In FDM, after calculating the support region, a region boundary extraction and toolpath generation process needs to be applied. These subsequent processes with benefit if the boundary ∂F_i contains less contour length and less topological complexity. Moreover, because a sparse pattern is always employed in the toolpath scanning for support generation, the topology complexity of the support region should be reduced in order to make good use of the sparse pattern. However, the region subtraction technique does not consider the topology of the resultant support region S_i. In fact, due to the growing-swallow technique, there will always be some gaps between S_i and P_i. These gaps will propagate into the lower layers and cause very complex topology on the resultant support region (see Fig.1.11). In this section, a region cleaning technique is introduced which can allow the user to generate F_i with much lower topology complexity from S_i.

We adopt the closing operation η_d for the region cleaning technique. It can fill the gaps and hence reduce the topology complexity. The pseudo-code is as follows.

Note that the selection of the radius d of element structure for $\eta_d(S_i)$ provides flexible trade-off between topology complexity and region cleaning effectiveness. The support region S_i computed by region subtraction definitely has a smaller area than F_i. On the other side, F_i has shorter boundary length and less topology complexity compared with S_i. Figure 1.11(c),(d) show the difference between resultant F_i with different d. Based on our experiments, $d = 2t$ can always give satisfactory result for FDM.

According to our investigation, provided that S_i is reliable, F_i satisfies $F_i \supseteq S_i$ and $F_{i+1} \subseteq F_i \cup P_i$, it can be proved that the reliable property also holds for F_i.

Proposition 4. If F_i satisfies $F_i \supseteq S_i$ and $F_{i+1} \subseteq F_i \cup P_i$, and the reliable property holds for S_i, then the reliable property also holds for F_i.

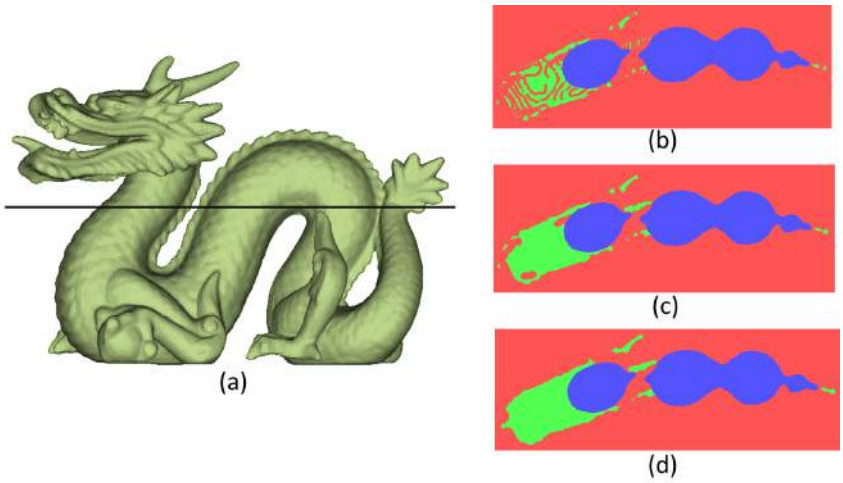

FIGURE 1.11: A demonstration of region cleaning technique: (a) region subtraction resultant support region (green) without region cleaning, (b) region cleaning result with $d = t$, (c) region cleaning result with $d = 2t$.

Algorithm 3 RegionSubtractionFDM

Require: image l_i with part region P_i for current layer, image l_{i+1} with part region P_{i+1}, support region S_{i+1} and final support region F_{i+1}, self-support feature threshold t

Ensure: final support region F_i on image l_i

1: $\Psi_i \leftarrow P_{i+1} - P_i$, $\widetilde{S}_i \leftarrow GrowingSwallow(\Psi_i, P_i, P_{i+1}, t)$;
2: $S_i \leftarrow RegionSubtraction(P_i, P_{i+1}, S_{i+1}, t)$;
3: Initialize $\Delta(P_{i+1}, P_i, t) \leftarrow \Psi_i - \widetilde{S}_i$;
4: $F_i = F_{i+1} \cup \eta_d(S_i) - \Delta(P_{i+1}, P_i, t) - P_i$;
5: **return** F_i;

Proof. First, for P_{i+1}, because $F_i \supseteq S_i$ and $P_{i+1} \subseteq P_i \bigcup S_i \bigcup \Delta(P_{i+1}, P_i, d)$, F_i must satisfy $P_{i+1} \subseteq P_i \bigcup F_i \bigcup \Delta(P_{i+1}, P_i, d)$. Second, for F_{i+1}, $F_{i+1} \subseteq F_i \cup P_i$ is already satisfied. We have

$$P_i \bigcup F_i \bigcup \Delta(P_{i+1}, P_i, d) \supseteq F_{i+1} \bigcup P_{i+1}.$$

Hence, the reliability property also holds for F_i. □

Proposition 5. the final support region F_i for FDM generated by algorithm *RegionSubtractionFDM* satisfies $F_i \supseteq S_i$ and $F_{i+1} \subseteq F_i \cup P_i$.

Proof. Because of $F_i = F_{i+1} \cup \eta_d(S_i) - \Delta(P_{i+1}, P_i, t) - P_i$, S_i cannot include $\Delta(P_{i+1}, P_i, t)$ and P_i, and the extensive property of closing operation, i.e., $\eta_d(S_i) \supseteq S_i$, we have $S_i \subseteq \eta_d(S_i) - \Delta(P_{i+1}, P_i, t) - P_i \subseteq F_{i+1} \cup \eta_d(S_i) - \Delta(P_{i+1}, P_i, t) - P_i = F_i$. Hence, we get $F_i \supseteq S_i$.

Because $\Delta(P_{i+1}, P_i, t)$ is the self-support region on P_{i+1}, i.e., $\Delta(P_{i+1}, P_i, t)$ is a part of P_{i+1}, F_{i+1} cannot include $\Delta(P_{i+1}, P_i, t)$. We have $F_{i+1} \cup \eta_d(S_i) - \Delta(P_{i+1}, P_i, t) \supseteq F_{i+1} \Longrightarrow (F_{i+1} \cup \eta_d(S_i) - \Delta(P_{i+1}, P_i, t) - P_i) \cup P_i \supseteq F_{i+1} \Longrightarrow F_i \cup P_i \supseteq P_{i+1}$. Hence, we get $F_{i+1} \subseteq F_i \cup P_i$.

□

From Proposition 4 and 5, final support region F_i can be generated by algorithm *RegionSubtractionFDM* and also has the reliability property.

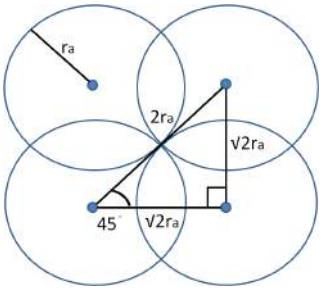

FIGURE 1.12: Grid length illustration in first step of anchor map generation.

1.4.4 Anchor Support Generation for SLA

For an image-based layer manufacturing process like SLA, the support region is not expected to have low topology complexity. In SLA, anchor support (vertical cylinders) is used instead of region filled with supporting material. Hence, there is no need for region cleaning technique. Anchor support generation approach is developed based on region subtraction technique. The algorithm is actually the same as for *RegionSubtraction* except for using anchor map A_i to cover the support region S_i. The anchor map A_i represents the support region to be cured in SLA, which consists of several isolated pixels indicating the position of the anchors needed. Since each anchor has its effective region, the union of all the effective regions from anchors indicated on A_i should cover the general support region S_i. Each time calculating $\widetilde{S}_i$, instead of directly counting it as the support region that needs to be filled with support material, an anchor map $\widetilde{A}_i$ is used to cover $\widetilde{S}_i$. Meanwhile, instead of S_{i+1}, the anchor map A_{i+1} is projected from the above layer into the current layer as PA_{i+1}. Finally, PA_{i+1} is integrated into anchor map A_i for the current layer. The associated pseudo-code follows.

Algorithm 4 RegionSubtractionSLA

Require: image l_i with part region P_i for current layer, image l_{i+1} with part region P_{i+1} and anchor support map A_{i+1}, self-support feature threshold t, anchor effective region radius t_a

Ensure: final anchor support map A_i on image l_i

1: $\Psi_i \leftarrow P_{i+1} - P_i$;
2: $PA_{i+1} \leftarrow A_{i+1} - P_i$;
3: $\widetilde{S}_i \leftarrow GrowingSwallow(GrowingSwallow(\Psi_i, P_i, P_{i+1}, t), PA_{i+1}, PA_{i+1}, t_a)$;
4: $\widetilde{A}_i \leftarrow AnchorMapGen(\widetilde{S}_i, t_a)$;
5: $A_i \leftarrow \widetilde{A}_i \cup PA_{i+1}$;
6: **return** A_i;

Note that the algorithm *AnchorMapGen* in the above method requires the definition of two parameters. The first one is $\widetilde{S}_i$, the region that needs to be covered by resultant anchor map, and the second one is the anchor effective region radius t_a. The anchor map indicates the position of anchors using pixels. For each anchor (a pixel x_a on the resultant map $\widetilde{A}_i$), $GrowingSwallow(\widetilde{S}_i, x_a, x_a, t_a)$ is used to remove its supporting region from $\widetilde{S}_i$. New anchors will be added until the whole region $\widetilde{S}_i$ can be supported by anchors in $\widetilde{A}_i$. An alternate pattern can be used for adding anchors on A_i. In our prototyping system, for simplicity, we use a heuristic method to implement $AnchorMapGen(\widetilde{S}_i, t_a)$. Figure 1.13 provides a demonstration for our implementation of *AnchorMapGen*. In the first step, a uniform grid is used to sample the region $\widetilde{S}_i$ and record the intersection points between $\widetilde{A}_i$ and grid nodes. Then we execute *GrowingSwallow* to exclude the supported region from $\widetilde{S}_i$ (see

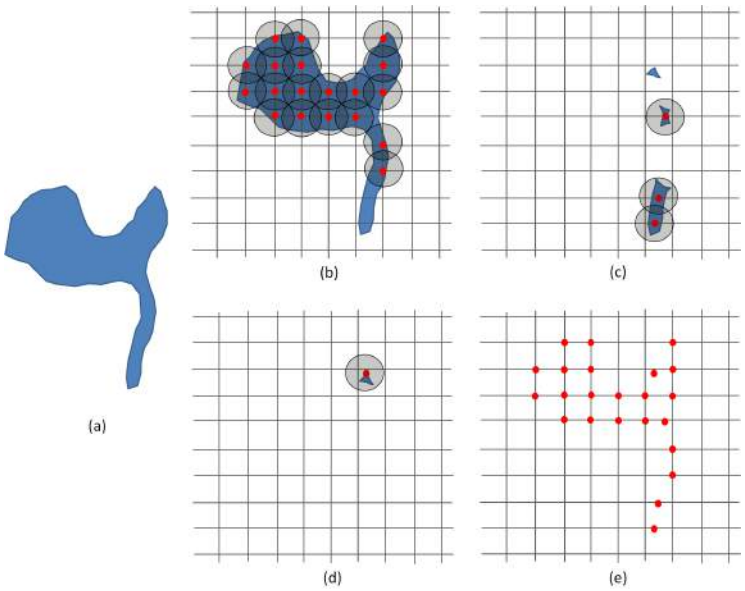

FIGURE 1.13: An illustration of anchor map generation algorithm: (a) region that needs to be covered by anchor map, (b) first step: use uniform grid node to intersect with the region and exclude the anchor-support region with growing-swallow technique, (c) second step: use orthogonal rays of the uniform grid to scan for remaining region, add anchors and further exclude anchor-support region, (d) third step: scan the whole image for remaining region, add anchors and exclude anchor-support region until there is no remainder, and (e) the resultant anchor map.

Figure.1.13(b)). Note that the grid width of the uniform grid used in $AnchorMapGen(\widetilde{S}_i, t_a)$ should be no bigger than $\sqrt{2}t_a$ in order to form a seamless region to cover $\widetilde{S}_i$ (see Fig.1.12). In the second step, a scan is performed with the orthogonal rays forming the uniform grid to detect whether there is any remainder of $\widetilde{S}_i$. If a ray intersects with any remaining region of $\widetilde{S}_i$, then add the centroid point of intersection line as anchor into $\widetilde{A}_i$. Then perform *GrowingSwallow* to further exclude the region from $\widetilde{S}_i$ [see Figure.1.13(c)]. In the last step, the whole image is scanned pixel by pixel to see whether there is any remainder of $\widetilde{S}_i$ and add anchors for them (see Fig.1.13(c)). Note that in the second and third step, each time adding an anchor, a *GrowingSwallow* is performed to exclude its supporting region from $\widetilde{S}_i$ until all regions in $\widetilde{S}_i$ are excluded. As introduced in [21], the pattern for adding new anchors can be optimized using the *centroidal voronoi tessellation* (CVT) method in order to use minimal number of anchors to cover $\widetilde{S}_i$. After generating anchor map A_i for all layers, the position and length for each anchor can easily be determined. Usually there should be some connection among these anchors for the purpose of providing rigidity.

We now introduce two approaches to generate connection graphs that indicate which pairs of anchors need to be connected. One approach is based on a *minimal spanning tree* (MST) which tends to minimize the number of connected anchor graphs and the total edge length of these graphs. The other is based on a local strategy which simply connects the two closest neighbors for each anchor. Before applying the MST algorithm, first generate input graphs for the MST algorithm. Then evaluate each pair of anchors to see whether they are overlapped in the building direction. When this is true, put an edge between these two anchors with the edge length being the distance between them. Otherwise neglect this pair of anchors (see Fig.1.15). In such a way, one or several graphs $\bar{G}_i$ are generated among all the anchors. Then use the Prim's algorithm [16] to generate minimal spanning tree with $\bar{G}_i$ as input graphs. For each $\bar{G}_i$, we will have a resultant graph G_i,

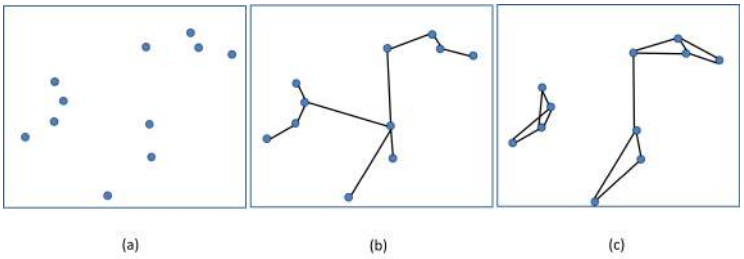

FIGURE 1.14: Anchor connection graphs generation illustration: (a) The top view of input anchors; note that all the anchors overlap in the building direction in this example, (b) the resultant anchor connection graph by MST-based approach, (c) the resultant anchor connection graphs by closest-neighbor-based approach. Note that (b) has only one graph while (c) has two graphs.

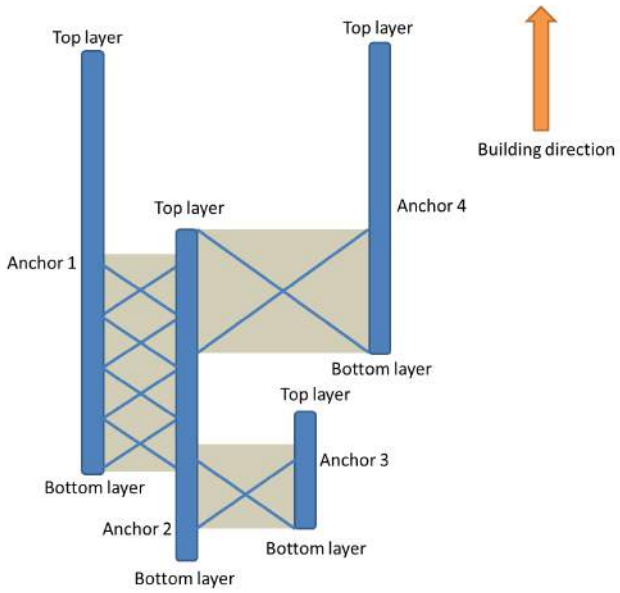

FIGURE 1.15: An illustration of connection generation between pairs of anchors. The gray region indicates the overlap range between two anchors.

and finally we have all the connection graphs G_i. For the closest neighbor-based approach, for each anchor, connect it with its two closest neighbors that overlap with it. Hence, we can directly get the connection graphs G_I. Fig.1.14 gives an illustration for both MST-based approach and closest-neighbor-based approach. In Section 1.4.5, a comparison and discussion about these two approaches is provided.

For the operator that generates connection between two anchors in the overlap range, first calculate a directing image between their projected positions on the binary image. All the pixels intersecting with the line segment connecting the two anchors are counted as connection structure pixels. This image indicates the path that can connect the two anchors (see Fig.1.16). Then, distribute the connection structure pixels on the path into the binary images of layers slicing a unit of connection structure (a single connection structure in cross-shape). Figure 1.17 provides a demonstration for the pattern of the distribution. For two adjacent layers, the pixels for the connection structure belong to the connection path while the intersection of connection structure pixels between the two layers

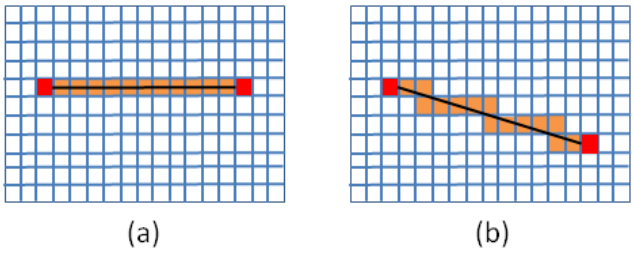

FIGURE 1.16: Directing image calculation for connection operator for SLA. The red pixels indicate anchor positions and orange pixels indicate the connection structure pixels; (a) and (b) show the resultant connection path for two different pairs of anchors.

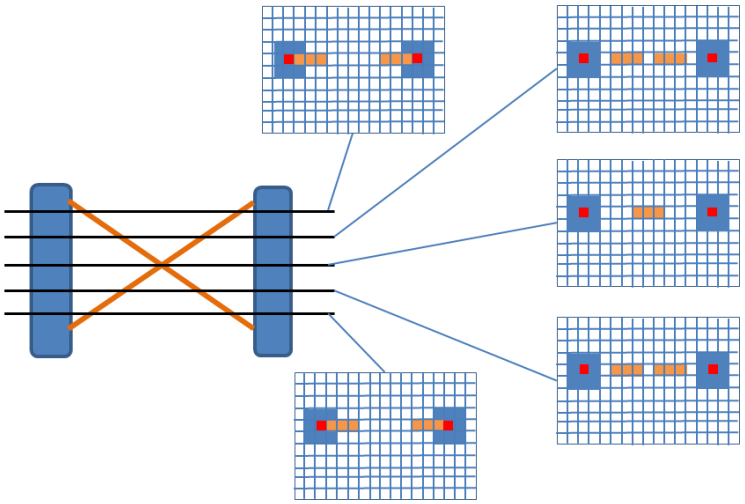

FIGURE 1.17: Connection structure pixel distribution among consecutive layers.

should not be empty. The position of the connection structure pixels depends on the layer height in a unit of connection structure.

1.4.5 Results and Discussion

We have implemented the proposed approach in a C++ program. The examples shown in this section are all tested on a PC with Intel Core i5-3450 CPU @ 3.10GHz.

We tested several models to demonstrate the effectiveness and efficiency of the presented region subtraction-based support generation approach (see Figuress.1.18, 1.19, 1.20, and 1.21). Computational statistics can be found in Table 1.2. All the tests are conducted with layer thickness 0.25 mm except for the two models shown in Fig.1.7. For simplicity, all the testing cases for SLA are conducted with the setting $t = t_a$ and with the MST-based approach for connection graphs generation if not otherwise specified. In our prototyping system, we did not use parallel computing to accelerate the proposed approach. As stated above that this approach is based on a binary image and mainly used Boolean, morphological operations. If parallel computing is involved, this approach's performance can be further improved.

We also have a comparison between the MST-based anchor connection and closest-neighbor-based connection. Table 1.1 gives the comparison for the four different models in terms of the

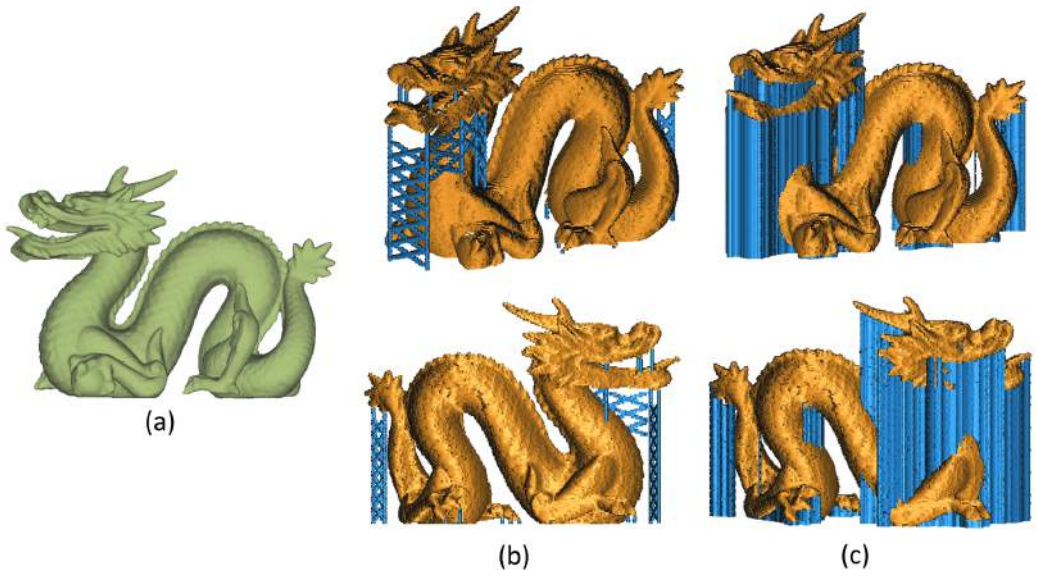

FIGURE 1.18: Support structure generation for dragon model: (a) original dragon model in LDNI, (b) anchor support for SLA with $t = t_a = 3.81$ mm, and (c) support for FDM with $t = 0.51$ mm.

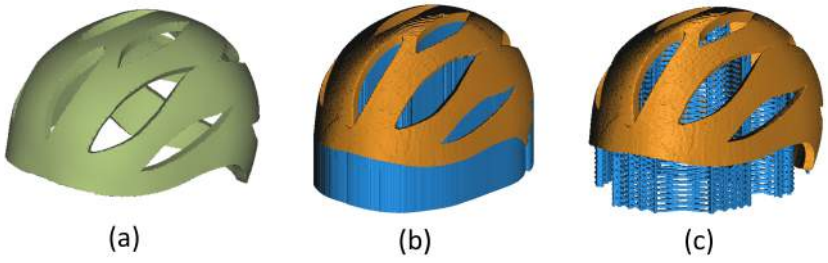

FIGURE 1.19: The support structure generated for helmet model: (a) original helmet model in LDNI, (b) support for FDM with $t = 0.51$ mm, and (c) support for SLA with $t = t_a = 3.81$ mm.

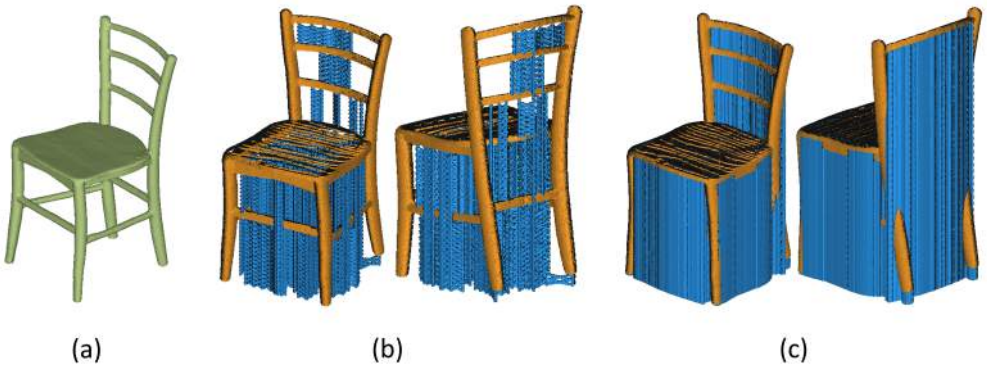

FIGURE 1.20: The support structure generated for chair model: (a) original model, (b) anchor support generation with $t = t_a = 3.81$ mm, and(c) support for FDM with $t = 0.51$ mm.

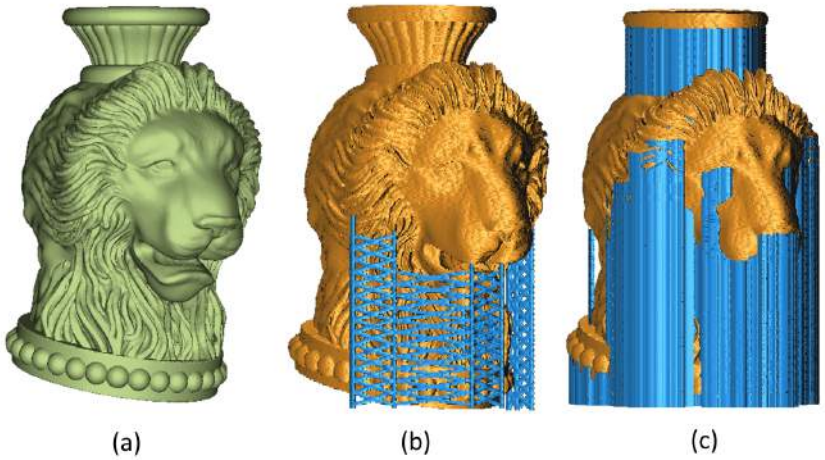

FIGURE 1.21: The support structure generated for visa-lion model: (a) original model, (b) anchor support generation with $t = t_a = 3.81$ mm, (c) support for FDM with $t = 0.51$ mm.

resultant connection volume and connection graphs generation time. There is no significant difference in the resultant connection volume between the two approaches. In fact, both of them take effort to minimize the edge length of the connection graphs. The MST-based approach tends to perform global minimization while the closest-neighbor-based approach tends to locally minimize the edge length. As for the time efficiency, there is also difficulty in observing any significant difference. For the connection structure the term rigidity refers to two aspects: self-support and withstand shearing force (usually due to sweeping operation or platform acceleration) during the building process. In terms of self-support, the closest-neighbor-based approach is better since it tends to avoid long-distance connection. Long distance connection requires a large overlap between the two anchors (see Figure.1.22). When building the connection link, long distance connection is easier to fail due to its larger weight. As for withstanding the shearing force, ideally it requires the anchors to form loops and the dimension of the loop in the shearing direction should not be too small. While this two strategies do not enforce this as a hard constraint, we need to compromise between rigidity and material efficiency. Although two strategies work well in our present prototype system, a more sophisticated approach can be developed in the future by modeling the shearing force case by case and optimizing material efficiency within sufficient rigidity.

The proposed support generation approach gives the user flexibility to control the self-support feature threshold according to the different specifications for different layer manufacturing systems. We also investigate the influence of the self-support feature threshold t and anchor effectiveness region radius t_a on support structure generation for both FDM and SLA. Figure 1.23 shows that the larger the value of t is, the less FDM support is needed because there will be more of a self-support feature. A similar result can be found in Figure 1.25. Figure 1.24 provides the data analysis for support generation for FDM with experimental tests on the dragon model. From Figure.1.24(b) we can see that the overall time consumption initially drops dramatically and then rises up slowly as t increases. The overall time consumption consists of the support region preparation time and the support contour extraction time. The main reason for the overall time drop at the beginning is due to the drop of support contour extraction. When t is small, the support region for each layer becomes larger. Consequently, this will increase the total region boundary length and also the contour extraction processing time. The support region processing time increases slowly as t increases, while the contour extraction time drops slowly after $t = 0.254$ mm. As indicated by Figure.1.24(c), the

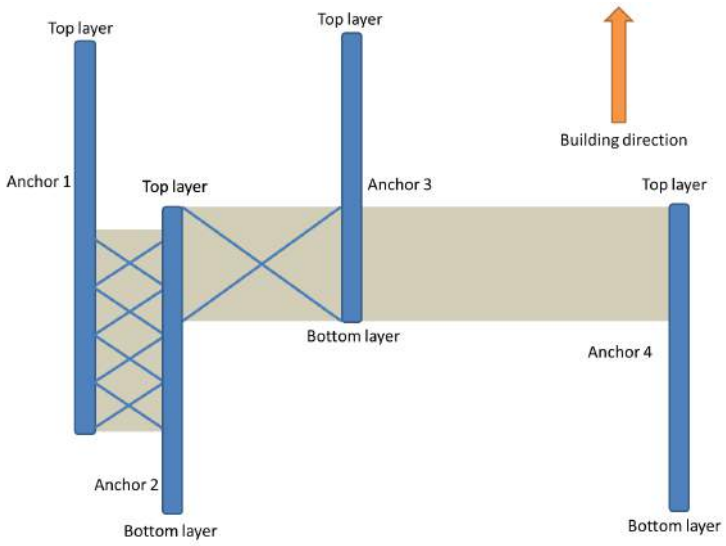

FIGURE 1.22: Long distance connection demonstration. Note that the overlap region between Anchors 3 and 4 is not big enough for even one unit of connection. And the connection between Anchors 2 and 3 may fail during building because the unit's self-weight is greater than the connection between Anchors 1 and 2.

Table 1.1: Comparison Between MST-Based and Closest Neighbor Based Anchor Connection Generation for SLA*

Examples	Support Volume in mm^3 (MST)	Support Volume in mm^3 (Closest Neighbor)	Time in sec (MST)	Time in sec (Closest Neighbor)
Dragon	88.07	94.80	0.79	0.70
Helmet	5,590.76	5,470.00	3.33	2.35
Chair	22,836.05	23,052.01	3.63	4.77
Vase-lion	590.09	552.24	2.28	1.43

*Reported SLA tests use the setting $t = t_a = 3.81$ mm, the time consumption includes only the anchor connection generation time.

volume of support structure for FDM keeps dropping as t increases while the slope becomes smaller and smaller. Figure 1.26 provides the data analysis for support generation for SLA with experimental tests on a helmet model. From Figure.1.26(b) we can see that the overall time consumption drops as t and t_a increase. It consists of anchor map preparation time and connection structure generation time. When the t,t_a is small, the number of anchors is large. This results in more time at the initial stage of generating a connection structure. When t,t_a become large, there will be fewer anchors and hence, the time for connection structure generation tends to decrease to a very low level. The time consumption for anchor map generation drops slightly as t,t_a increase. When t,t_a become large, the anchor map time consumption starts to dominate the overall time consumption. From Figure.1.26(c) we can see that the volume for SLA support decrease as t, t_a increases while

Table 1.2: Statistics of Experimental Tests of Support Generation for FDM and SLA

Examples	Model Size (x mm×z mm×y mm)	r' (mm)	Time for FDM* (min)	Time for SLA$^{\#}$ (min)
Dragon	50.80×22.86×28.956	0.050	0.79	0.70
Helmet	84.58×108.71×73.66	0.114	3.33	2.35
Chair	67.82×71.12×132.08	0.101	3.63	4.77
Vase-lion	56.39×46.23×76.20	0.076	2.28	1.43

*Reported FDM tests use the setting $t = 0.381$ mm; the time consumption includes final support region F_i generation and contour ∂F_i extraction.
$^{\#}$Reported SLA tests use the setting $t = t_a = 3.81$ mm; the time consumption includes the anchor support map A_i for all layers and connection structure generation.

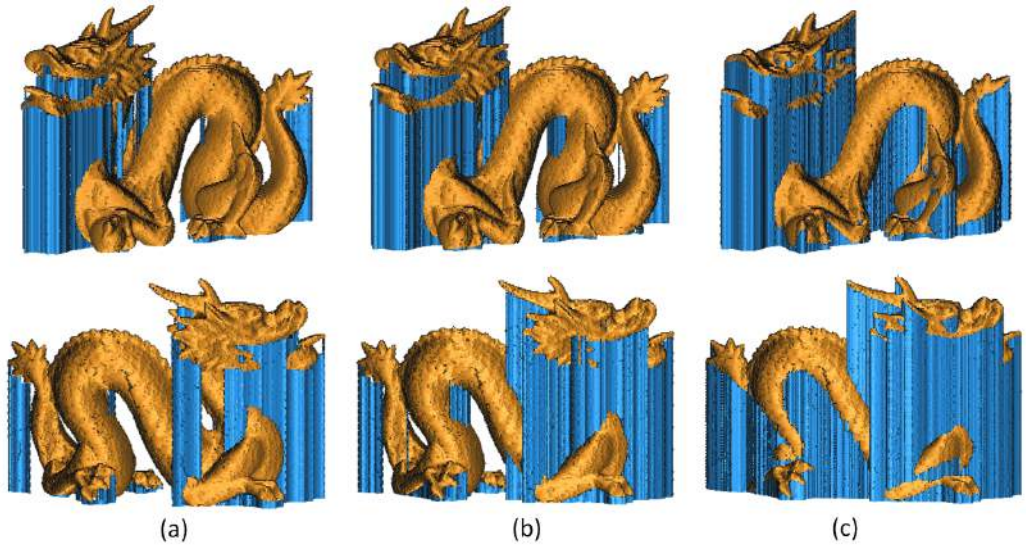

FIGURE 1.23: The comparison of different FDM support for different t value: (a) FDM support with $t = 0.508$ mm, (b) FDM support with $t = 0.381$ mm, and (c) FDM support with $t = 0.254$ mm.

the slope becomes smaller.

1.5 Topologically Faithful Slicing Contour Generation

Slicing CAD models is a crucial operation for generating tool paths in layered manufacturing. Conventional slicing algorithms focus on computing the intersection curves between a model represented by triangular meshes and a sequence of parallel planes, which becomes an unstable step for the whole procedure of layered manufacturing if the triangular meshes are self-intersected (or overlapped). In addition, more and more modeling approaches represent objects with complex struc-

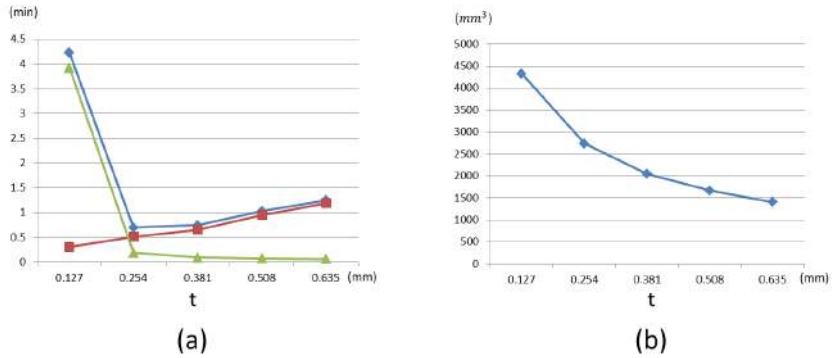

FIGURE 1.24: Data analysis for FDM support generation on dragon model: (a) the chart of time consumption for different t values, and (b) the chart of total support volume for different t values.

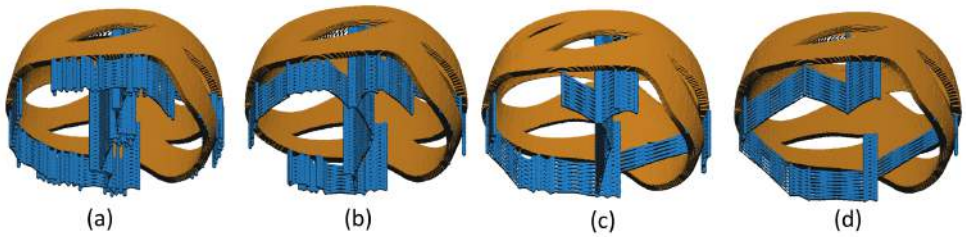

FIGURE 1.25: The comparison of different anchor support for different t, t_a values: (a) anchor support with $t = t_a = 2.54$ mm, (b) anchor support with $t = t_a = 3.81$ mm, (c) anchor support with $t = t_a = 5.08$ mm, and (d) anchor support with $t = t_a = 6.35$ mm.

tures by implicit solids since such representations are mathematically compact and robust. When conventional slicing methods are used, the implicit solids must be first tessellated into triangular meshes and then be intersected by slicing planes. However, generating a self-intersection free and topologically faithful polygonal model from an implicit solid is not easy (see [14, 27] for detailed discussions). Specifically, the triangular models produced always have problems like gaps, degenerated triangles, overlapped facets, non-manifold entities, and self-intersections. Using conventional slicing techniques to generate contours for layered manufacturing from such problematic triangular meshes will result in an incorrect object. For example, as shown in Figure.1.2, unexpected gaps are produced on the Buddha model fabricated by FDM. This gap is caused by the self-intersected polygonal model, which brings in inverse in/out membership classifications to some planar contours.

The topology of the final model to be fabricated is usually required to be homeomorphic to the given solid. This is very important to applications such as biomedical engineering, e.g., a fabricated model with an incorrect topology may merge two tubes which should be separated into one, which is very dangerous for medical treatments. Meanwhile, shape approximation errors between the extracted contours and the exact ones defined by intersecting the given solid with the slicing plane must also be controlled. To fabricate an object with high accuracy in a conventional way of tessellation, a massive number of triangles may be generated, and storing them in-core will use up the memory of a computer system. This motivates our research on the development of a direct slicing algorithm to generate self-intersection free and topologically faithful contours from a general implicit solid.

Recall that for a given implicit solid H, its reconstructed surface M and a slicing plane P, we com-

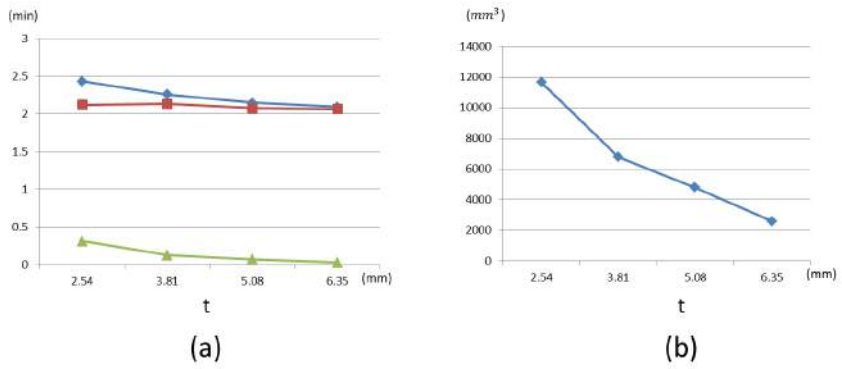

FIGURE 1.26: Data analysis for SLA support generation on helmet model: (a) the chart of time consumption for different t and t_a values, and (b) the chart of total support volume for different t and t_a values.

pute contours $C = M \cap P$, which are topologically faithful and self-intersection free. The presented approach consists of three major steps. First, a binary image I of an r-regular solid H is sampled on the slicing plane P, where an appropriately selected sampling distance r' (with its bound relating to the value r) ensures that the contour $\bar{C}^0$ generated from I is homeomorphic to C. Second, $\bar{C}^0$ is iteratively smoothed into $\bar{C}^m$ by a constrained Laplacian operator that prevents topological changes and self-intersections. Finally, a constrained contour simplification is applied to simplify $\bar{C}^m$ into a contour $\tilde{C}$, which has fewer line segments, and $\tilde{C}$ satisfies the three requirements previously given in the problem definition. Proofs for the correctness of $\tilde{C}$ can be found in [11]. A flowchart of the provided direct slicing approach is presented in Figure.1.27.

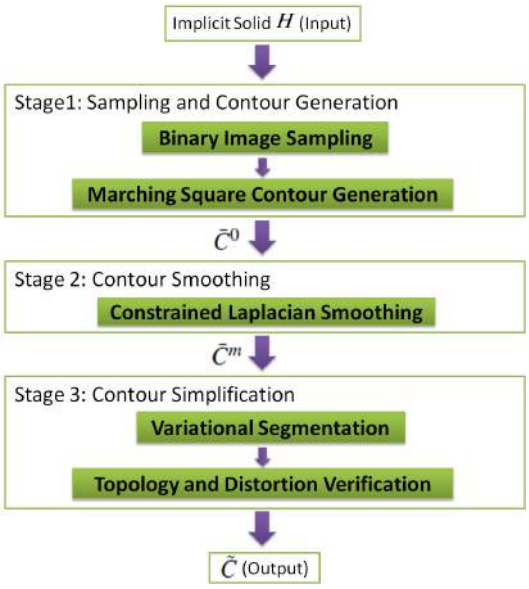

FIGURE 1.27: A flowchart of our direct slicing approach [11].

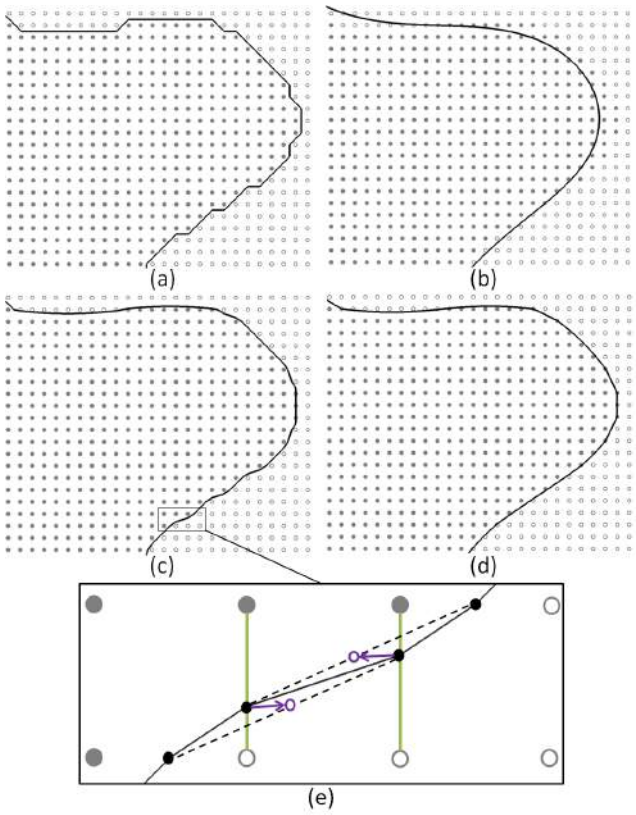

FIGURE 1.28: A comparison of different smoothing strategies on the contour generated for the binary image region shown in Fig. 1.4(c): (a) the contour reconstructed by the topology preserving marching square method, (b) the shrinking contour after ordinary Laplacian smoothing, (c) the resultant contour after projection-based constrained smoothing, (d) the resultant contour after sliding-based constrained smoothing, and (e) the zoom-in view of contour vertices stuck in sub-optimal shape.

1.5.1 Contouring and Constrained Smoothing

After obtaining a binary image, the marching square method introduced in [18] is used to generate an approximate contour $\bar{C}^0$ that is topologically faithful. Defining the edge on a square grid with different in/out status on its two endpoints as a *stick*, the contour $\bar{C}^0$ can be formed by the line segments linking the middle points of sticks in all grids. Since line segments on the contour $\bar{C}^0$ are generated by linking the middle point of sticks, there is no self-intersection on $\bar{C}^0$. However, the shape of $\bar{C}^0$ is not smooth [e.g., the contour shown in Fig. 1.28(a)], so a Laplacian operator-based smoothing technique is applied to improve it. The advantage of Laplacian smoothing is its efficiency and stability, but the major drawback is that the unwanted shrinkage always occurs when it is iteratively applied to a closed shape (in 2-D or 3-D). A constrained Laplacian smoothing is developed here, which intrinsically solves the shrinkage problem since we guarantee generation of topologically faithful contours. In other words, the smoothed contours cannot violate the in/out status of any sampling node on the binary image I. To ensure that, a good strategy is to constrain each vertex $\mathbf{v}_i$ on the contour only to slide on the stick gripping it during the smoothing. In addition, by applying this "*sliding-on-stick*" strategy, we can guarantee that the resultant contours are self-intersection free [11].

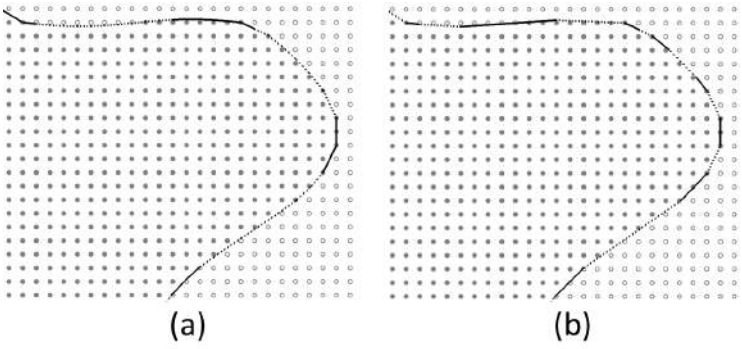

FIGURE 1.29: An illustration of contour simplification for the smoothed contour shown in Fig.1.28(d): (a) the variational clustering result on the contour with different line types representing different regions and (b) the final simplified contour after topology and distortion verification.

1.5.2 Contour Simplification

The smoothed contour, $\bar{C}^m$, usually provides a very good shape approximation of the exact contour generated by $\partial H \cap P$. However, to ensure the topological faithfulness, a relatively small value of r' may be selected for a model with large dimensions. This leads to contours with a lot of very short line segments, which significantly increase the memory cost. The situation becomes more serious if the software controlling the RP machine does not run in an out-of-core manner (i.e., loading the contours for all layers from the contour file at the same time). Moreover, using too many small line segments to represent the contours will dramatically decrease the efficiency of subsequent processing steps in RP, like generation of supporting structure and tool-path planning. Based on our observation, in the smoothed contours, there are always several successive edges lying almost in the same straight line, which implies these edges can be simplified into one single edge with little distortion error introduced (see Fig. 1.29). Therefore, a contour simplification algorithm preserving topology and shape approximation error is investigated in this section to further improve the topologically faithful contours for slicing implicit solids. Details can be found in [11].

1.5.3 Results and Discussion

Our approach has been implemented in a C++ program. The examples shown in this section are all tested on a PC with Intel Core 2 Quad CPU Q6600 2.4 GHz.

The two engineering models shown in Figs.1.30 and 1.31 give a comparison between the traditional slicing approach and the proposed approach. Due to the self-intersection in the tessellated triangular meshes from implicit solids, the traditional slicing approach may produce incorrect contours and consequently, the toolpath of part material will also have defects [see Figs.1.30(b) and 1.31(b)]. As a result, unwanted films will be produced for both of the two examples . In addition, the filigree and truss models demonstrate that the presented approach can easily handle the solids with complex topology.

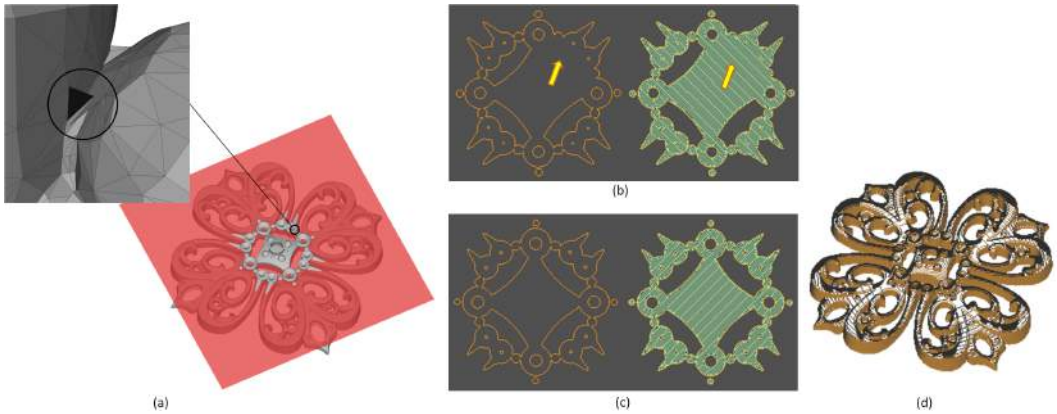

FIGURE 1.30: An example of slicing the Filigree model: (a) a mesh tessellated from implicit Filigree model; the self-intersection feature is shown in the black circle, (b) the contours and their corresponding tool path (in green) generated by Insight^{TM} version 7.0 on the layer with 19.56 mm height; the yellow arrow is pointing to the incorrect contour generation, (c) the contours generated by the presented approach for the same layer and their corresponding tool path, and (d) the rendered slicing contours generated by the presented approach.

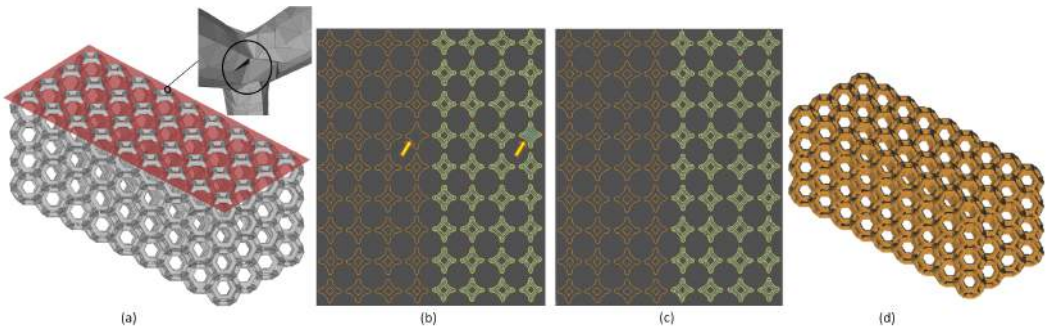

FIGURE 1.31: An example of slicing the Truss model: (a) a mesh tessellated from implicit Truss model; the self-intersection feature is shown in the black circle, (b) the contours and their corresponding tool path (in green) generated by Insight^{TM} version 7.0 on the layer with 61.97 mm height; the yellow arrow is pointing to the incorrect contour generation, (c) the contours generated by the presented approach for the same layer and their corresponding tool path, and (d) the rendered slicing contours generated by the presented approach.

1.6 Conclusion

In this chapter, two major parts of algorithms for layered manufacturing in image space are presented. First, in Section 1.4, a support generation approach based on sampled binary images for slices is introduced, then an error-bound conservative region subtraction which can produce safety guaranteed and material efficient support region is presented. Support generation methods are developed for FDM and SLA with the region subtraction as a core technique. Meanwhile, the proofs for the good properties of the presented support generation approach are provided. Second, in Section 1.5, a direct slicing approach for implicit models is presented, which directly slices an implicit

model by sampling a binary image on each layer and extracting self-intersection free and topology correct contours.

The presented approach is generally more robust than conventional support generation and slicing approaches, according to the following aspects.

- As shown in Section 1.1, the robustness of the conventional approach highly depends on the quality of input triangular mesh. Any non-manifold feature may lead to incorrect contour, and hence incorrect part obtained in fabrication.
- For support structure generation, the model-based conventional method uses triangular mesh to generate support mesh directly and then slice the support mesh. Its robustness highly depends on the quality of the input model. Moreover, it suffers from the numerical errors, as mentioned above, when tracing the contour on the support mesh. The slicing-based conventional method requires polygonal operations including offset, Boolean operation in order to calculate support structure contours from part slicing contours. While the provided support generation method is totally based on discrete binary image operations like Boolean and morphology to compute the support structure region, it can provide self-intersection free contour using the proposed contour processing technique if needed. It is obvious that discrete Boolean and morphological operation on a binary image is much more robust than polygonal operations based on floating-point arithmetic.
- As stated in Section 1.4, the presented support generation method calculates the reliability support region, which cannot be guaranteed by present slicing-based support generation approaches. The reliable property of the support region will make the fabrication process more robust.
- In our direct slicing, instead of tracing the contour on the model, which is not robust, a binary image is sampled on the slicing plane and extract contour from the 2-D uniform grid. In the tracing-based contour generation, one needs to use floating-point arithmetic to determine whether an edge intersects with the slicing plane. This may lead to tracing failure due to numerical errors in cases like when the edge is nearly parallel with the slicing plane. On the other hand, in binary image sampling, only query with respect to implicit model is required. Since the implicit model is always mathematically compact, the query should be robust to evaluate and the result should be accurate. Also, the marching square method can guarantee self-intersection free and topology correct contours, which is important to fabricate part robustly.

Since most processes of the presented approach are based on binary image operations, which gives us an opportunity to boost the performance using parallel computing architecture on devices like *Graphics Processing Unit* (GPU).

- For support generation: the region subtraction stage should not be difficult to parallelize fully. It consists of scan-based discrete distance map evaluation, Boolean operation, and morphological operation on binary image. Boolean and morphology operations can be easily parallelized with each thread handling one pixel. For scan-based discrete distance map calculation, there is already an algorithm presented in [23] dealing with it. For region cleaning technique for FDM, the closing operation involved is also a morphological operation which can be implemented by a dilation followed by an erosion. Hence, this technique can also be parallelized. In anchor support generation for SLA, the scan-based anchor generation portion can be implemented as three grow-swallow operations with each grow-swallow per scan. As we mentioned above, grow-swallow only consists of Boolean and morphological operations and can be parallelized. For the anchor connection graph construction, there is already some recent approach on parallelizing MST algorithm on GPU [28]. The anchor connection operator needs to calculate the connection path on binary image between two anchors. This can

be parallelized with the anchor graph stored in shared memory and each thread handling one pixel. The pixel for connection structure can also be determined in a parallel way with each thread handling one pixel.

- For direct slicing: the binary image sampling can be highly parallelized since the in/out classification of each pixel can be evaluated independently. In the marching square method, each thread can handle one grid edge independently to see if it is a stick. After finding all the sticks, the rest is just creating instances for vertices and edges and there is little computation involved. Hence this portion does not need to be parallelized. For the constrained smoothing stage, each vertex can be stored with its two neighboring vertex coordinates in a single thread in order to calculate the intersection point and update the vertex position independently in each thread. For the simplification stage, the variational segmentation requires requires global adjustment of not only the number of the segment regions but also the shape of each region in order to optimize the global error metric. Hence, it is difficult to parallelize the segmentation at present. In the topology and distortion verification, the verification can be conducted in each segment region independently and so, this portion can be parallelized. However, if the region needs to be further segmented, this can only be performed on a CPU as stated above.

According to the analysis presented in this section, except for the contour simplification stage, almost all the other processing of the presented approach can be parallelized using hardware acceleration. Hence, the presented approach still has the potential for boosting its performance and the parallelization falls within our plans for future work.

Acknowledgments
The research conducted in this paper is supported by Hong Kong RGC/GRF grants (CUHK/417109 and CUHK/417508). The third author acknowledges the financial support from the National Science Foundation under grant CMMI-0927397 and CMMI-1151191. The authors would like to thank Yuen-Shan Leung and Allan Mok for generating the LDNI solid and the FDM object shown at the beginning of this paper, and Ms. Siu Ping Mok for proof reading the manuscript

---

1.7 Nomenclature

1.7.1 Abbreviations

1. STL stereolithography, page 3
2. LDNI layered depth normal image, page 4
3. FDM fused decomposition modeling, page 4
4. SLA stereolithography, page 5
5. MLS moving least square, page 6
6. UDM euclidean distance mapping, page 6
7. CVT centroidal voronoi tessellation, page 18
8. MST minimal spanning tree, page 18
9. TVS truncated volume segment, page 31

1.7.2 Symbols

1. H Input implicit model, page 5
2. ∂H Boundary of the model H, page 7
3. l_i Slicing image for the ith layer, page 4
4. P_i Part region (a set of pixels) on l_i, page 5
5. l_{i+1} The above adjacent slicing image of l_i, page 7
6. P_{i+1} Part region on l_{i+1}, page 7
7. S_i General support region produced by region subtraction on l_i, page 5
8. S_{i+1} General support region on l_{i+1}, page 7
9. F_i Support region produced for FDM on l_i, page 5
10. A_i Anchor map produced for SLA on l_i, page 5
11. ∂P_i Boundary contour for the part region P_i, page 5
12. ∂F_i Boundary contour for the support region F_i, page 5
13. M Reconstructed boundary surface for input model H, page 7
14. r Radius of the osculating ball used to define r-regular solid, page 7
15. r' Cube width of the 3-D sampling grid, page 7
16. P Slicing plane, page 7
17. d Self-support feature threshold, page 7
18. $\triangle$ Self-support region, page 7
19. I Binary image sampled on the slicing plane P, page 7
20. $\tilde{C}$ Final region contour generated from I, page 7
21. $\tilde{C}^0$ Initial region contour generated from I, page 8
22. Ψ Shadow region, page 10
23. Γ Outward offset region, page 10
24. $\widetilde{S}_i$ Intermediate support region to produce S_i, page 10
25. $\mathscr{C}$ Element structure, page 10
26. γ Dilation, page 10
27. ε Erosion, page 10
28. ζ Opening, page 10
29. η Closing, page 10
30. A_{i+1} Anchor map on layer l_{i+1}, page 17

31. $\widetilde{A}_i$ Intermediate anchor map to produce A_i, page 17
32. t_a Anchor effective region radius, page 17
33. $\bar{C}^m$ Contour after smoothing operation, page 28

References

[1] R.E. Barnhill, G.E. Farin, M. Jordan, and B.R. Piper. Surface/surface intersection. *Computer Aided Geometric Design*, 4:3–16, 1987.

[2] R.E. Barnhill and S.N. Kersey. A marching method for parametric surface/surface intersection. *Computer Aided Geometric Design*, 7:257–280, 1990.

[3] K. Chalasani, L. Jones, and L. Roscoe. Support generation for fused deposition modeling. In *Solid Freeform Fabrication Symposium 1995*, pages 229–41, University of Texas, Austin, TX, 1995.

[4] Y. Chen and C.C.L. Wang. Layered depth-normal images for complex geometries - part one: Accurate sampling and adaptive modeling. In *ASME IDETC/CIE 2008 Conference, 28th Computers and Information in Engineering Conference*, New York, NY, 2008.

[5] J.W. Choia, F. Medinaa, C. Kima, D. Espalina, D. Rodrigueza, B. Stuckerc, and R. Wicker. Development of a mobile fused deposition modeling system with enhanced manufacturing flexibility. *Journal of Materials Processing Technology*, 211(3):424–432, 2011.

[6] C.K. Chua, K.F. Leong, and C.S. Lim. *Rapid Prototyping: Principles and Applications*. World Scientific, Singapore, 2003.

[7] M. Couprie and G. Bertrand. Topology preserving alternating sequential filter for smoothing two-dimensional and three-dimensional objects. *Journal of Electronic Imaging*, 13(4):720–730, 2004.

[8] P.E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14(3):227–248, 1980.

[9] R.T. Farouki. The characterization of parametric surface sections. *Computer Vision, Graph and Image Processing*, 33:209–236, 1986.

[10] R.C. Gonzales and R.E. Woods. *Digital Image Processing*. Prentice Hall, 2 edition, 2002.

[11] P. Huang, C.C.L. Wang, and Y. Chen. Intersection-free and topologically faithful slicing of implicit solid. *ASME Journal of Computing and Information Science in Engineering*.

[12] X. Huang, C. Ye, S. Wu, K. Guo, and J. Mo. Sloping wall structure support generation for fused deposition modeling. *International Journal of Advanced Manufacturing Technology*, 42, 2009.

[13] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. *ACM Transactions on Graphics*, 21(3):339–346, 2002.

[14] T. Ju and T. Udeshi. Intersection-free contouring on an octree grid. In *Proceedings of Pacific Graphics*, 2006.

[15] C.F. Kirschman, C.C. Jara-Almonte, A. Bagchi, R.L. Dooley, and A.A. Ogale. Computer aided design of support structures for stereolithographic components. In *Proceedings of the 1991 ASME Computers in Engineering Conference*, pages 443–448, Santa Clara, CA, 1991.

[16] J. Kleinberg and E.Tardos. *Algorithm Design*. Addison-Wesley, first edition, March 2005.

[17] R.B. Lee and D.A. Fredericks. Intersection of parametric surfaces and a plane. *IEEE Computer Graphics and Application*, 4(8):112–117, 1981.

[18] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4), 1987.

[19] R.C. Luo and Y. Ma. A slicing algorithm for rapid prototyping and manufacturing. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2841–2846, 1995.

[20] K. Prashant, M. Anne, and D. Debasish. A review of process planning techniques in layered manufacturing. *Rapid Prototyping Journal*, 6(1), 2000.

[21] X. Qian, K. Li, and Y. Chen. Direct geometry processing for tele-fabrication. In *Proceedings of the ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Chicago, IL, 2012.

[22] Y. Qiu, X. Zhou, and X. Qian. Direct slicing of cloud data with guaranteed topology for rapid prototyping. *International Journal of Advanced Manufacturing Technology*, 53(1-4), 2011.

[23] J. Schneider, M. Kraus, and R. Westermann. Gpu-based real-time discrete euclidean distance transforms with precise error bounds. In *International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 435–442, 2009.

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

[25] B. Swaelens, J. Pauwels, and W. Vancraen. Support generation for rapid prototyping. In *Proceedings of the 6th International Conference on Rapid Prototyping*, pages 115–121, University of Dayton, OH, 1995.

[26] S.E. Umbaugh. *Computer Vision and Image Processing*. Prentice Hall, 1998.

[27] R. Varadhan, S. Krishnan, L. Zhang, , and D. Manocha. Reliable implicit surface polygonization using visibility mapping. In *Proceedings of Symposium on Geometry Processing*, 2006.

[28] V. Vineet, P. Harish, S. Patidar, and P.J. Narayanan. Fast minimum spanning tree for large graphs on the GPU. In *Proceedings of the Conference on High Performance Graphics 2009*, pages 167–171, New York, NY, 2009.

[29] C.C.L. Wang and Y. Chen. Layered depth-normal images for complex geometries - part two: Manifold-preserved adaptive contouring. In *ASME IDETC/CIE 2008 Conference, 28th Computers and Information in Engineering Conference*, New York, NY, 2008.

[30] D. Webb, V. Verdes, and C. Cassapis. Computer-aided support-structure design for stereolithography models. In *Proceedings of the 5th International Conference on Rapid Prototyping*, pages 221–228, University of Dayton, OH, 1994.

[31] P. Yang and X. Qian. Adaptive slicing of moving least squares surfaces: Toward direct manufacturing of point set surfaces. *ASME Transactions Journal of Computing and Information Science in Engineering*, 8(3), 2008.