공학석사 학위논문

Optimal support generation for easy removal in a layered manufacturing process

FDM 방식에서의 효율적인 서포트 제거를 위한 최적 설계

2017년 2월

서울대학교 대학원

기계항공공학부

오 정 택

Abstract

We propose an optimization framework for a support removal process. Support removal is an issue among increasing users of fused deposition modeling (FDM). Many users have to use tools to eliminate support structures from 3D-objects. Furthermore, while removing support, the 3D-object tends to break when too much stress is applied. Prior to building a 3D-object, our algorithm provides the optimal orientation that minimizes the amount of support structures using a convex-hull algorithm. Also by providing a deformable support structure that detaches better when the same force is applied, we aim to facilitate the support removal process for FDM users. We have compared our support structure generation algorithm with support structures from existing software using a materials tester, INSTRON 5900R. We also compared the time required to remove the support structure.

Keyword: 3D-printing, support structure, deformability, connectivity, removal, orientation, convex-hull

Student Number: 2015-20737

Table of Contents

List of Figures

Chapter 1. Introduction

Fused Deposition Modeling (FDM), also known as fused filament fabrication (FFF) is a highly publicized technology of 3D-printing. As in (Figure 1), FDM is a process that extrudes melted thermoplastic filament through a heated nozzle, and the nozzle lays filaments upon another. Although many alternatives for FDM such as stereolithography (SLA), selective laser sintering (SLS), and digital light processing (DLP) exist, FDM is still a main contributor in popularizing 3D-printing technology for economic reasons.

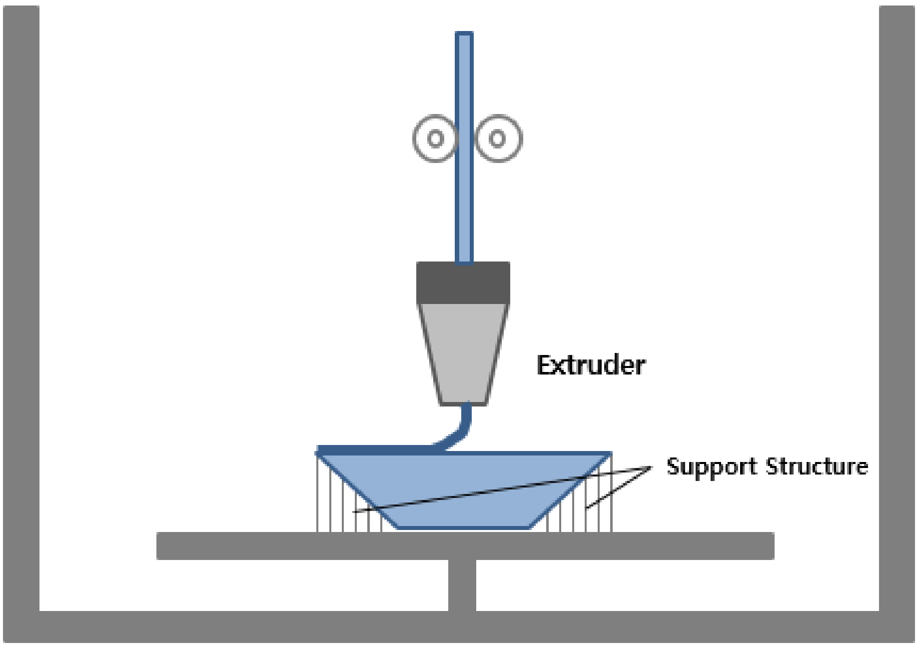

Figure 1. Principle of FDM

While SLS does not require support structures, many additive manufacturing (AM) technologies such as FDM, SLA, and DLP need support structures to prevent the 3D-object from collapsing during production as shown from (Figure 1). Support structures need to be generated considering overhang of a model. Several works to reduce the amount of support material appeared as FDM became popular since the patents were released to the public. Some works focused on generating efficient support structures that uses less amount of material. However, these works lack the consideration of support removal process, and less amount of material does not always lead to less amount of build-time, which is also important.

Many FDM users have trouble when detaching support structure from 3D-object. Usually tools are required to facilitate the removal process and it takes long time to fully remove support structure from 3D-object. In this research, we focused on the support removal process of FDM for users.

Orientation of 3D-object has decisive impact on many important parameters such as build time, surface quality, and etc. To find the optimal orientation in terms of support removal, analysis from diverse rotation is needed. As works by Stava et al. [1], we used convex−hull algorithm to select meaningful rotation candidates from a limitless number of rotation candidates. In order to eliminate support smoothly from a 3D−object, we propose an algorithm to find the optimal orientation where minimum amount of support structure is required.

In addition to optimal orientation, we propose a deformable support structure for easy removal on FDM. For easy removal, we presumed that elasticity and connectivity are two characteristics which make it easier to eliminate support structure from 3D−object. Elasticity or deformability, which makes the support structure deform well when pressure or force is applied, is the main contributory factor for easy removal in our algorithm. Therefore, we concentrated on making support structure to have elastic property by controlling the toolpath of 3D−printer. Connectivity of the support structure is a prerequisite of making the support structure have deformability. Connectivity of support structure also contributes to support elimination of places where tools cannot approach, support generated in 3D−object with hole, for example.

Contribution. As described in Section 3 and 4, we proposed an optimization framework for support removal process by providing optimal orientation and deformable support structure. Also, as in Section 5, we have compared our algorithm with existing software such as line, grid support from Cura™, and tree support from Autodesk® Meshmixer™ using INSTRON 5900R which is a universal testing instrument for tensile, compression, bend, peel, shear, tear and cyclic tests.

Chapter 2. Related work

2.1. Support structure

Since each layer needs to be printed upon beneath layer, 3D-object needs support structure where beneath layer does not exist. Due to the fact that support structure leads to longer build time and waste of materials, several attempts to generate new support structure exist. Autodesk, [2], generated Branching support structures and it reduces print material and print time compared to line structure when printing a 3D-object. Vanek et al. [3] use tree-like support structure, and they compared their algorithm with Autodesk's. In tree algorithm, support pillars regroup until they reach the printing bed. Vanek maintain that their algorithm reduced build time by an average of 11.75% and the amount of material use by 12.4% than Autodesk's branching algorithm. Jeremie et al. [4] proposed more stable support structure than tree-type support using bridging the gap algorithm, which is efficient in both build time and material use. Tree support and support from works of Jeremie is far superior material savings than existing software.

Algorithm of support structures mentioned from [2-4], is densely generated near the surface of 3D−object. However, it is much more difficult to eliminate support from 3D−object when support structure is densely generated. Furthermore, from printing experiments, we sometimes encountered with print failure problems when printing a 3D−object with tree type support structure.

2.2. Infill structure

Overall build time of 3D−printing has always been one of the biggest issue. Works about efficient infill structure [5−7], rather than new support structure, exist to reduce build time and save materials.

2.3. Orientation analysis

Finding optimum build−orientation of a 3D−object has been an important issue in FDM. Optimal build orientation varies along the purpose of printing such as support volume, strength, and surface quality. Users have different preference of the parameters. Ezair et al. [8] used depth−buffer and GPU to quickly find optimal orientation toward minimum support volume. Although this algorithm has merits on fast computing time, it cannot consider overhang, described in 3.1.

Usually, users use angle in range of 30° and 60° as overhang angle. If overhang is not considered, result would be different if model is composed of complex features.

In the works of Umetani et al. [9], based on the stress analysis using Finite Element Method (FEM), the orientation of 3D-object is optimized to increase mechanical strength. In [10], training and learning methodology was developed to preserve important visual features by avoiding support structures on import regions. Fine details of a model can be preserved if the amount of support structure attached to important regions is reduced. In the works of Thrimurthulu et al. [11], surface finish and part deposition time are considered to find the optimum part orientation.

Chapter 3. Support structure

We present the process and algorithm of generating our deformable support structure for easy removal. We also introduce characteristics that make our support structure superior in removal in this section.

3.1. Support region

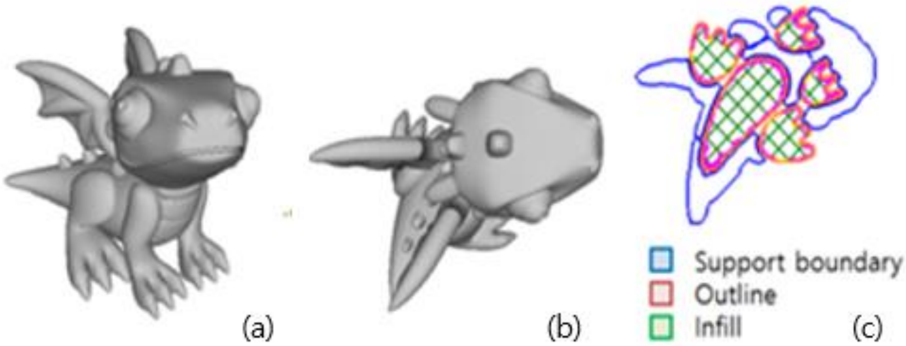

Figure 2. (a) Cute Dragon model front view (b) Top view (c) Cross−section from top view

In order to generate support structure from a model, we need to find region where support structure needs to be generated. In (Figure 2.c), tool−path of support boundary, outline and infill are represented from top view. Support structure should be placed inside the boundary, colored in blue, without crossing outline of a model which is colored in red. We used algorithm from open source Cura™

to find support region.

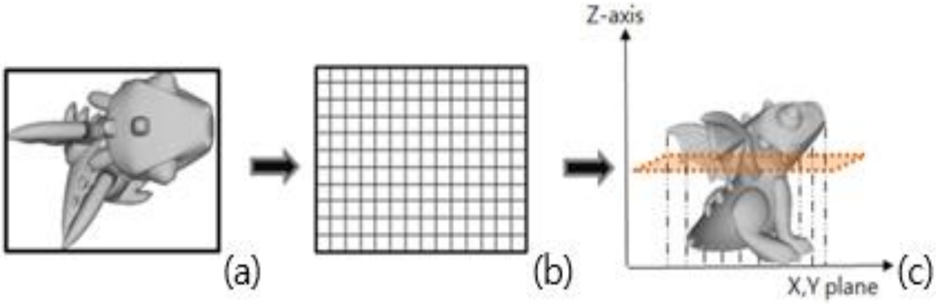

Figure 3. Generation overview of support region (a) AABB (b) Grid (c) Projection

First step is to make AABB (Axis−Aligned Bounding Box) of 3D−object as in (Figure 3.a). To make AABB, we obtain information of x_{min}, x_{max}, y_{min}, and y_{max} by iterating every point on 3D−object. After that, we generate grid as (Figure 3.b) by dividing AABB into n by m rectangles. This process is for checking individually whether the region from a certain layer needs to be supported.

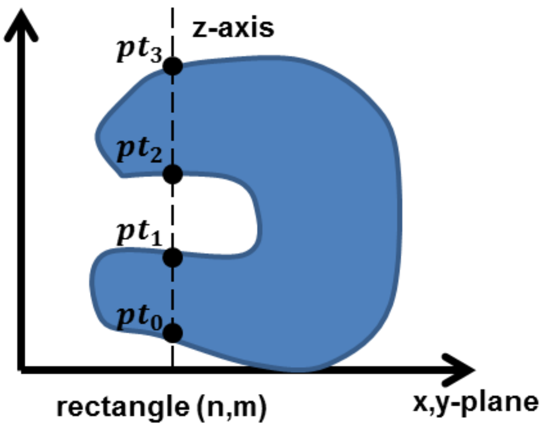

Figure 4. Intersection between model and back−projection line

From a designated point on every rectangle in (Figure 3.b), upper right corner points for example, back projecting a point such as rectangle (n.m) on (Figure 4) onto the 3D-object is necessary. Then, with the back-projection line parallel to z-axis, we save every point (pt_0~pt_3 in (Figure 4)) on the 3D-object that interacts with the line. We also save an angle between the face normal the intersection point is placed on and (0, 0, −1) vector to deal with overhang. Then, with the saved points information, we can obtain support region.

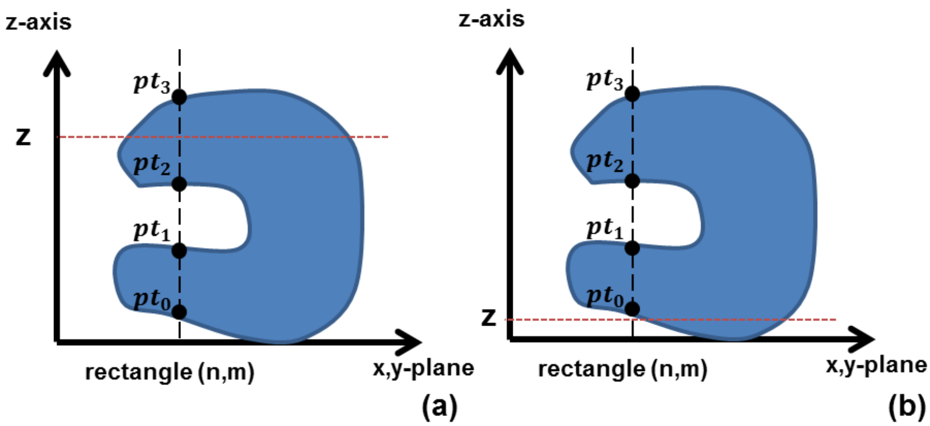

Figure 5. Support region determination in certain layer

Depending on the thickness of a layer, number of layer for a model is decided. To make support structure, for every layer, making support boundary through checking every rectangle in grid is necessary. For explanation, a point called rectangle (n, m) is chosen in (Figure 4, 5).

$$pt_{2i} < z < pt_{2i+1} \qquad (1)$$

If the layer with z−value, which is being checked, is between pt_{2i} and pt_{2i+1}, the point (rectangle(n,m).X, rectangle (n,m).Y, z) is inside the model. So support is not needed on that point.

$$z < pt_{2i} \;\; (i = 0) \;\; or \;\; pt_{2i-1} < z < pt_{2i} \;\; \text{(i=1,2,}\cdots\text{,n)} \qquad (2)$$

If a point, back projected from rectangle (n, m), satisfies condition of (2), overhang angle of pt_{2i} must be considered. If the angle between face normal and (0, 0, −1) vector is smaller than θ_c, the point needs to be supported. Therefore, when saving points in (figure 4), we also save angle between the face normal and (0, 0, −1) vector to deal with overhang condition.

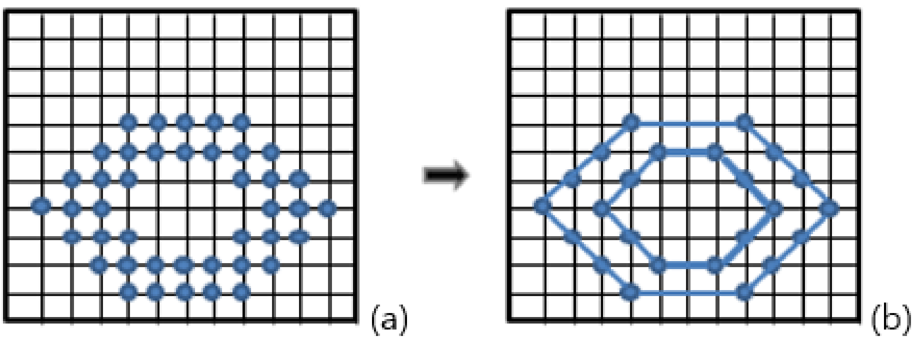

Figure 6. (a) Overhang points (b) Support boundary

Finally, for every layer, we can obtain the region where the support needs to be placed by connecting the boundary points checked to be supported. As in (Figure 6.b), we can obtain support boundary.

3.2. Deformable support structure generation

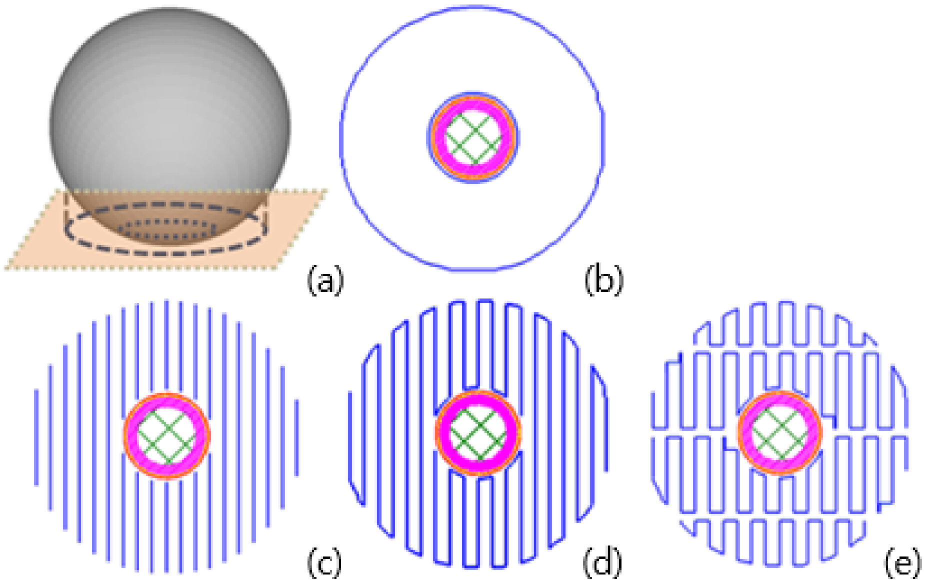

Figure 7. (a) Sphere model (b) Support Region (c) Line (d) Connect (e) Split

(Figure 7) shows the process of generating our support structure. (Figure 7.b, c, d, e) each is a cross−section of the model (Figure 7.a). (Figure 7.c) is the line type support structure. First of all, by connecting lines as in (Figure 7.d), support structure can be eliminated together when force is applied from a user. Also as explained in section 3.3, deformable characteristic facilitate users to remove support easily. Furthermore, we have split support structure as in (Figure 7.e) for large models. The vertical length of split support structure in (Figure 7.e) can be modified by user input.

3.2.1 Line support generation

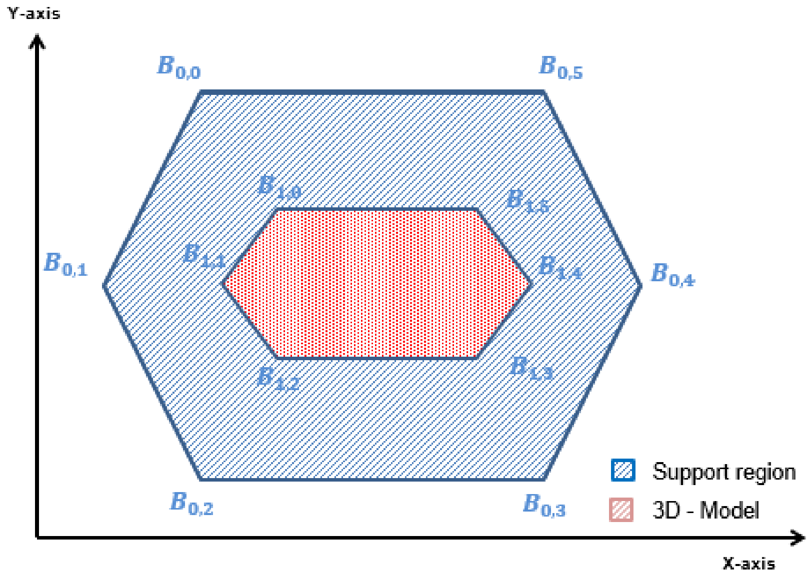

Figure 8. Cross−section of an example model

We have explained in detail how to obtain support region from a model in section 3.1. To explain how to generate support structure within support region, we use simple example as (Figure 8). Boundary points, $B_{n,m}$ $(n\text{: }boundary\ number, m\text{: }point\ index)$, are saved in counter clockwise direction. Support structure should be generated inside support region colored in blue, (Figure 8). If it crosses red region colored in red, support structure is overlapped with 3D-model. In 3D-printing, overlapping in a layer lowers the quality of model outcome. Support structures are normally generated with an offset from 3D-model to ensure high quality of outcome after support removal. So overlapping of support and 3D-object should be prohibited.

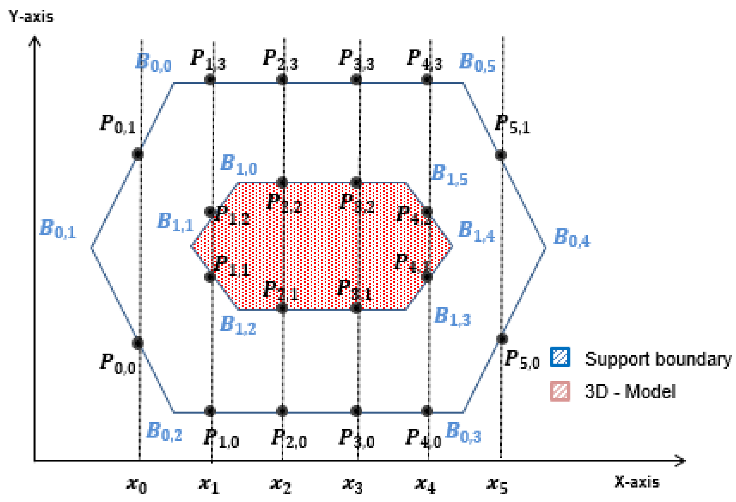

Figure 9. Intersection of the model with support boundary

When user sets the density of support, number of lines and x-coordinates of lines ($x_0 \sim x_n$) are decided. By intersecting support boundary with x-coordinates ($x_0 \sim x_n$), intersection point set $P_{i,j}$ can be obtained. In $P_{i,j}$, i refer to index sorted by x-coordinate and j refer to index sorted by y-coordinate.

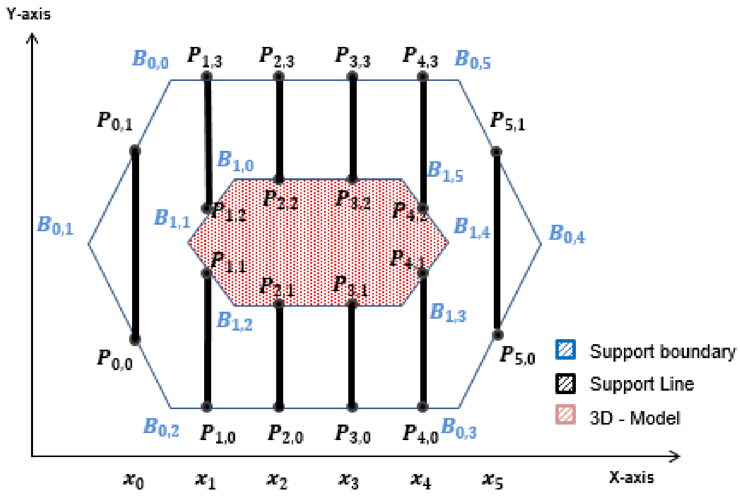

Figure 10. Line support generation

To generate line type support structure, we simply connect $P_{i,j}$, which j is even, with $P_{i,j+1}$ as in (Figure 10). We need to connect lines to make support structure as (Figure 7.d).

3.2.2 Connecting the lines

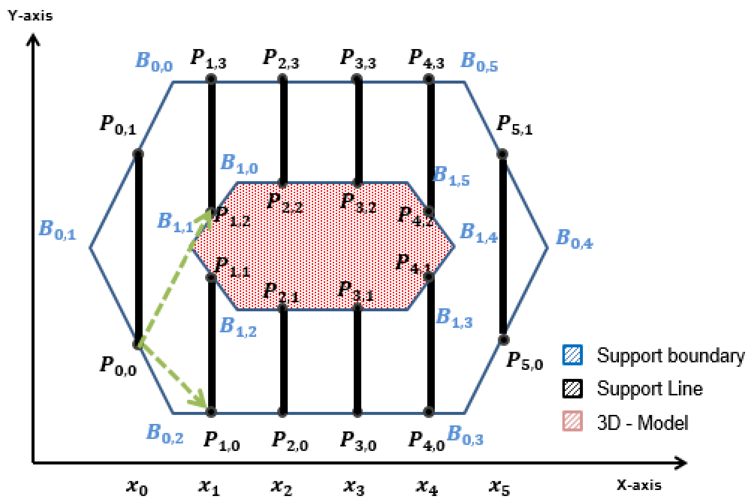

Figure 11. Connecting criteria

To connect the lines with pattern as (Figure 7.d), we've set some criteria as in Algorithm 1.

Algorithm 1. Connecting the lines

if i of $P_{i,j}$ is even
  for all $P_{i,even}$
    Find closest point among $P_{i+1,even}$
      Connect ($P_{i,even}$, $P_{i+1,even}$)
else
  for all $P_{i,odd}$
    Find closest point among $P_{i+1,odd}$
      Connect ($P_{i,odd}$, $P_{i+1,odd}$)

By Algorithm 1, $P_{0,0}$ in (Figure 11) should be connected with $P_{1,0}$. However, when connecting two points with different x−coordinate, connecting line must be inside support boundary. To ensure this condition, line is made following support boundary as in (Figure 12).

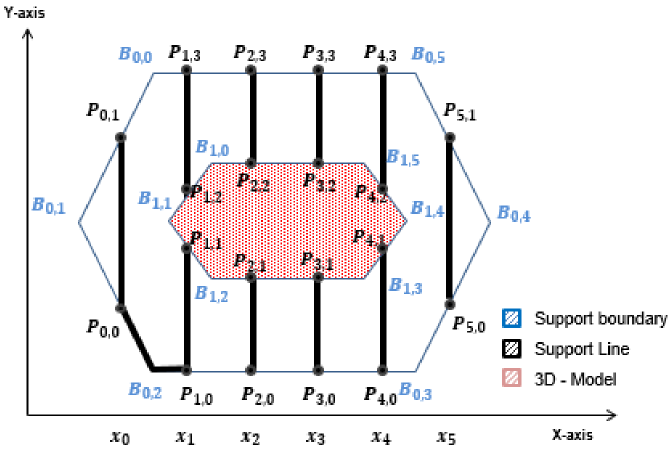

Figure 12. Connecting lines through the boundary line

Same algorithm can be applied until x_4 as in (Figure 13). However, $P_{4,2}$, $P_{4,0}$ will be connected to $P_{5,0}$ using Algorithm 1.

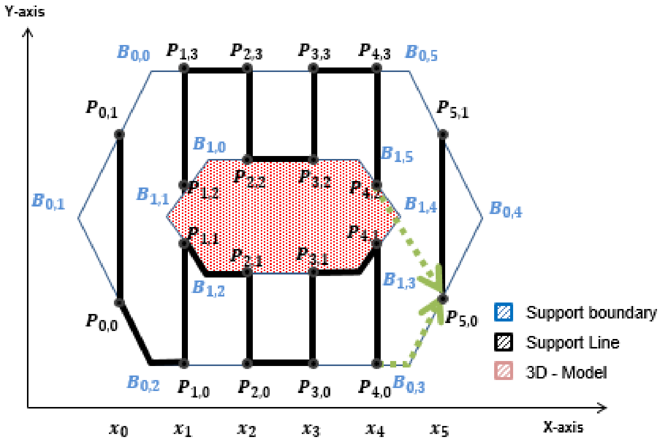

Figure 13. Connecting exception

This overlap problem in (Figure 13) happen when number of points on x_i is larger than x_{i+1}. The number of lines on x_i is larger than x_{i+1}, so the overlap problem occurs.

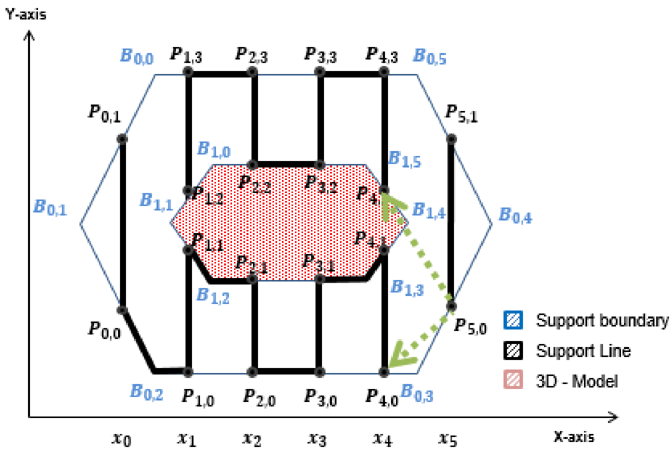

Figure 14. Solution to overlap problem

To solve the problem, when number of points in x_{i+1} are larger, we inspected closest point among $P_{i,j}$ from $P_{i+1,j}$ as in (Figure 14). Then connectivity pattern as (Figure 15) can be obtained. We only connected a side from one line by avoiding overlap problem, because support split into some parts are better in support removal.

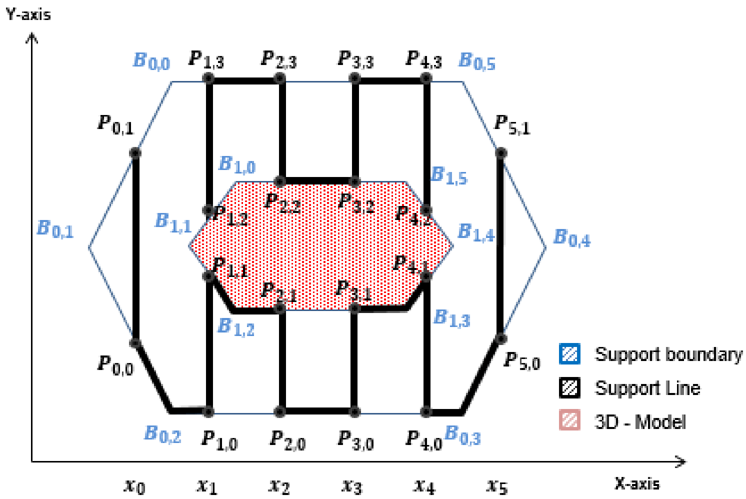

Figure 15. Connected lines

Connecting two points through boundary line is only possible when connecting points share same boundary. We will explain how we connected points which share different boundary. In this case we cannot use support boundary.

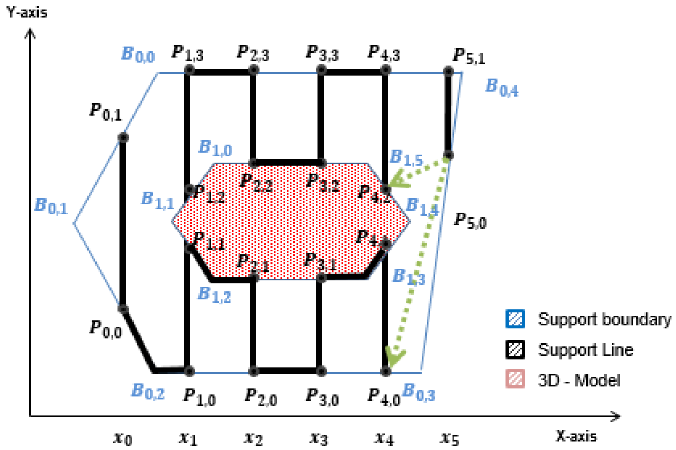

Figure 16. Connecting points which share different boundary

As example in (Figure 16), cases, which connect points sharing different boundary, can happen. However, if we connect these points directly, the green line as in (Figure 17) can collide with the model.

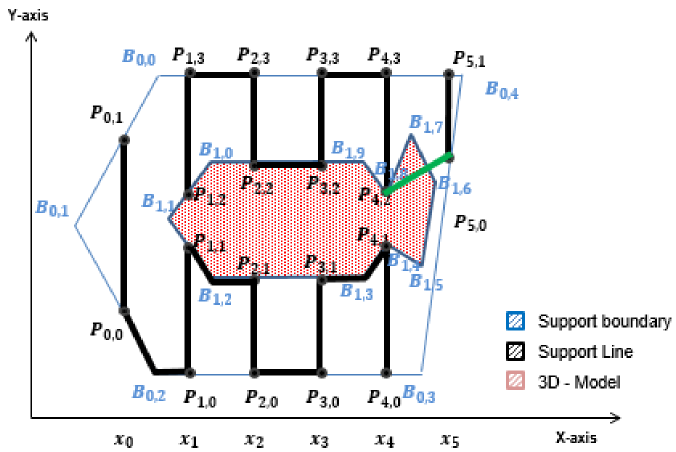

Figure 17. Connecting points without using support boundary

Therefore, when connecting two points which share different support boundary, we check all the lines of boundary, which include selected points to be the closest, with connecting line. For example, in (Figure 17), $P_{4,2}$ is chosen as the closest from $P_{5,0}$, so we check all the boundary lines made with $B_{1,j}$ with connecting lines. If collision is detected, we abandon connecting the two lines. However, this is very rare case, because this happen when cross-section of the model is very sharp and thin, thinner than the gap between x_i and x_{i+1}.

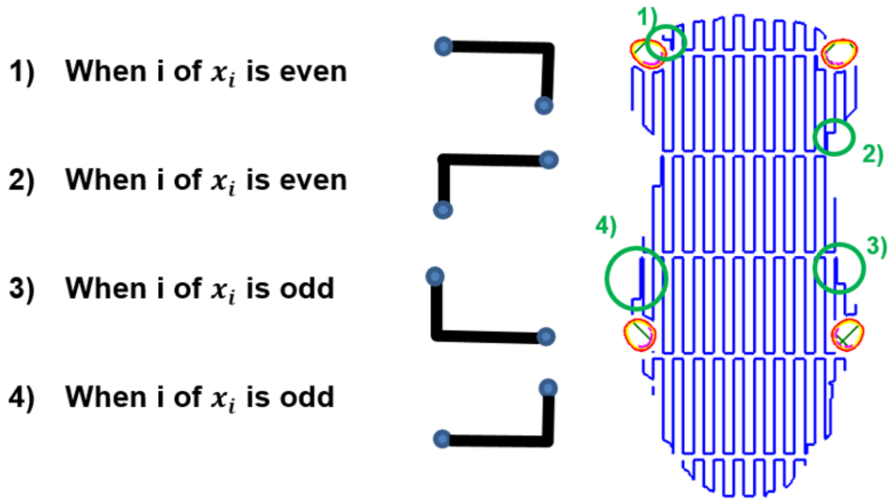

Figure 18. Different boundary connection

Furthermore, for different support boundary connection, we have generated patterns which lower collision with the model as in (Figure 18). Therefore, the Algorithm 1 should be modified as

Algorithm 2.

Algorithm 2. Connecting the lines

```
if n_i is smaller than n_{i+1}
    if i of P_{i,j} is even
        for all P_{i,even}
            Find closest point among P_{i+1,even}
                Connect (P_{i,even}, P_{i+1,even})
    else
        for all P_{i,odd}
            Find closest point among P_{i+1,odd}
                Connect (P_{i,odd}, P_{i+1,odd})
else
    if i of P_{i,j} is even
        for all P_{i+1,even}
            Find closest point among P_{i,even}
                Connect (P_{i,even}, P_{i+1,even})
    else
        for all P_{i+1,odd}
            Find closest point among P_{i,odd}
                Connect (P_{i,odd}, P_{i+1,odd})
```

As explained, when connecting two points with same support boundary, two points are connected through support boundary. If two points share different support boundary, we use pattern in (Figure 18) and perform collision test. To facilitate this, when we save intersection points in (Figure 9), we also save boundary index and index of two boundary points which involve in intersection.

3.2.3 Splitting the deformable support structure

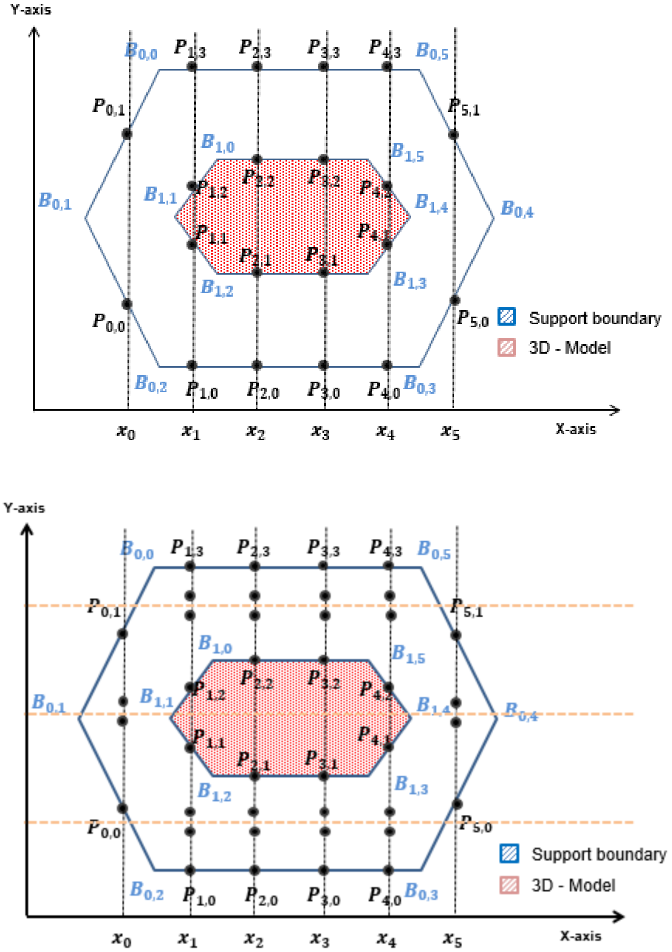

Figure 19. (Above) Figure 11, (Below) Splitting process of support structure

We have split the support structure as in (Figure 7.e). Understanding splitting algorithm is comparatively easy. Before connecting the intersection points into lines, we add some points between intersection points. We add additional points using orange dotted line in (Figure 19). We add a point above the dotted line and below the line at regular intervals. However, we only add these points when these points are inside support region. We check these points are inside $P_{i,2x}$ and $P_{i,2x+1}$.

Algorithm 3. Splitting the support into parts

```
Get y_min, y_max from AABB
Let n_min = y_min / split_length
Let n_max = y_max / split_length

for all x_i ∈ x
    for all B_{n,m} ∈ B
        Find intersection point P_{i,j} using x_i, B_{n,m}, B_{n+1,m}
for all x_i ∈ x
    while (j < size(P_i)) {
        for n = n_min to n_max {
            y = n*split_length
            if( (P_{i,j}.Y − y)( P_{i,j+1}.Y − y) < 0 ) {
                Add a point P(x_i, y+gap)
                Add a point P(x_i, y−gap)
            }
        }
        j+=2
    }
```

Using Algorithm 3, we can add additional points to split deformable support structure. This algorithm is useful when the 3D-object is large. By splitting the support structure into parts, users can easily remove support structure.

3.3. Characteristics of deformable support structure

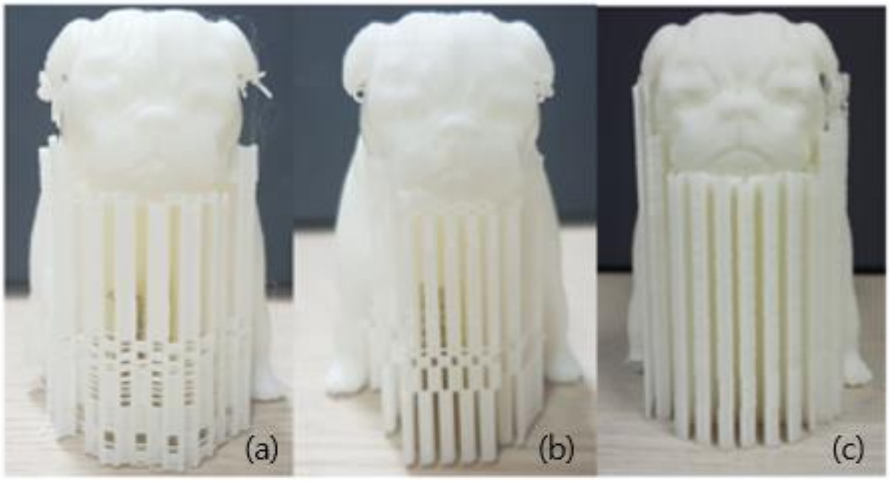

Figure 20. Support structure of (a) Line (b) Simplify (c) Ours

We have tested mostly used support structures among FDM 3D-printing such as Cura™'s line, grid, Meshmixer™'s tree, and Simplify3D support structure. However, grid and tree type of support structure are inferior than others in terms of support removal. (Figure 20) shows support structures of Line, Simplify3D and ours.

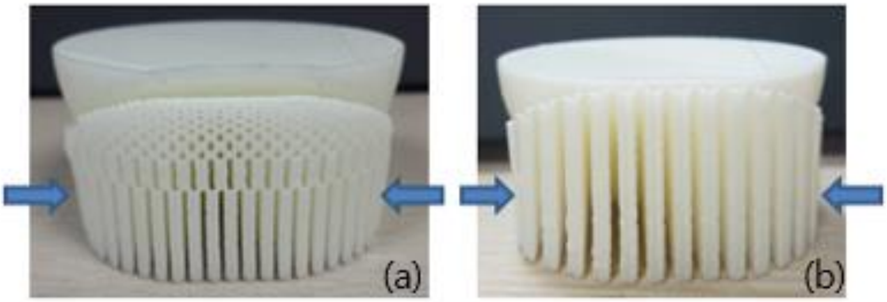

Figure 21. Support structure with (a) inconsistent pattern (b) consistent pattern

As FDM is a layer by layer process, bonding force is much stronger in horizontal direction than in vertical direction. When removing FDM support, users usually apply force as represented in (Figure 21). When force is applied to remove support, breakage happens on the most vulnerable contact surface. If the support structure is connected and has elastic property, the whole support transforms together when force is applied. The characteristic that makes the structure transform well, contribute to easier elimination of support structure. We assumed that inconsistent pattern or twisted support lead to difficult support removal. Therefore, as in (Figure 21.b, 22.b), we have made support structure with consistent pattern, which contribute to deformable structure.

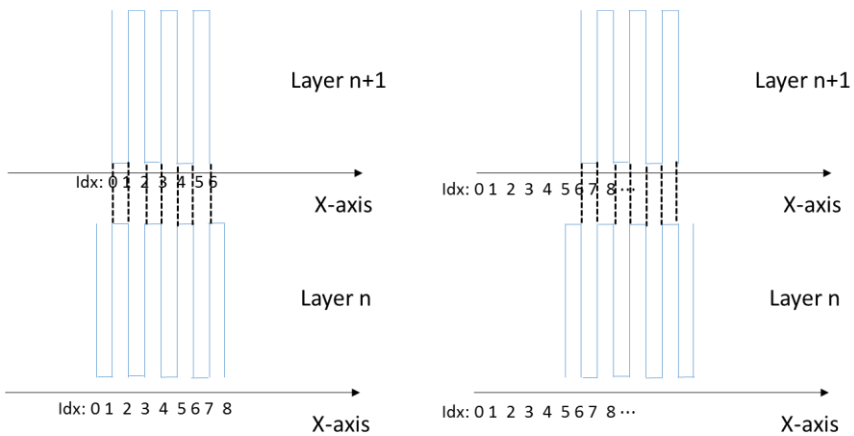

Figure 22. Lines with (a) Relative index (b) Absolute index

Places, where lines are needed, differ depending on index of layer and shape of a model. To make a consistent pattern, we used absolute index of lines as in (Figure 22.b).

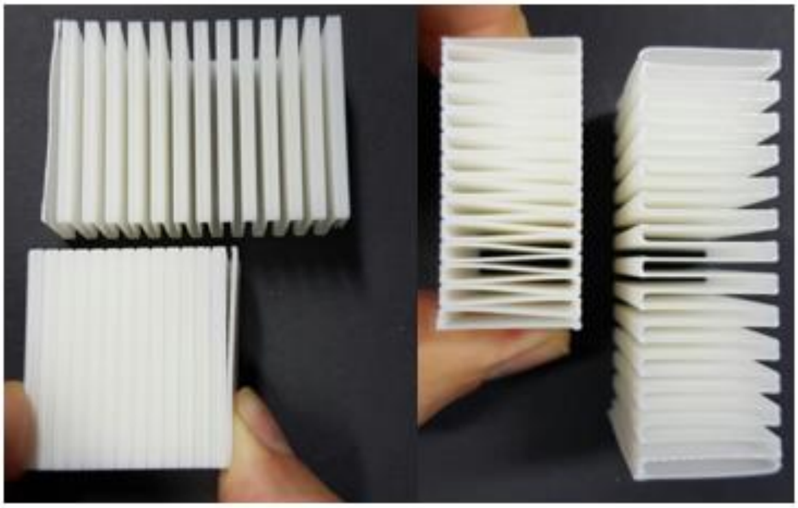

Figure 23. Deformable characteristic of consistent patterned support

As shown in (Figure 23), consistent pattern that almost does not vary along z-direction, have deformability. With very small force, the support structure deforms like a spring. When force is removed, it is restored to initial state as it undergoes linearly elastic behavior if too much force is not applied. This characteristic allows the support structure to be removed easily compared to other types of support structures such as Line, Simplify3D, Tree and etc.

Chapter 4. Part orientation for minimal support

4.1. Overhang

As FDM is a layer by layer process, filaments need to be piled upon another layer. However, it is inefficient to support all the downward facing surfaces of the 3D-object. In 3D-printing, overhang angle is a term, which defines the critical angle θ_c for support generation.

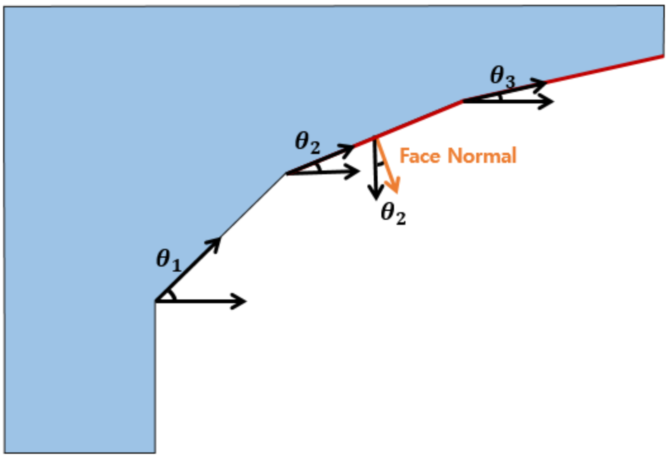

Figure 24. Different overhang angles

3D-object is normally constituted of triangular faces or polygon faces. Using the normal vector of a face, we can obtain an angle θ_i between face and z-axis vector. If the angle θ_i is smaller than overhang angle θ_c, the region needs to be supported. Usually, θ_c is set between 30° and 60°. In (Figure 25), faces that are in overhang region are represented in red color and others in blue. The amount of support required to build a 3D-object differs by the overhang angle θ_c, which user sets.

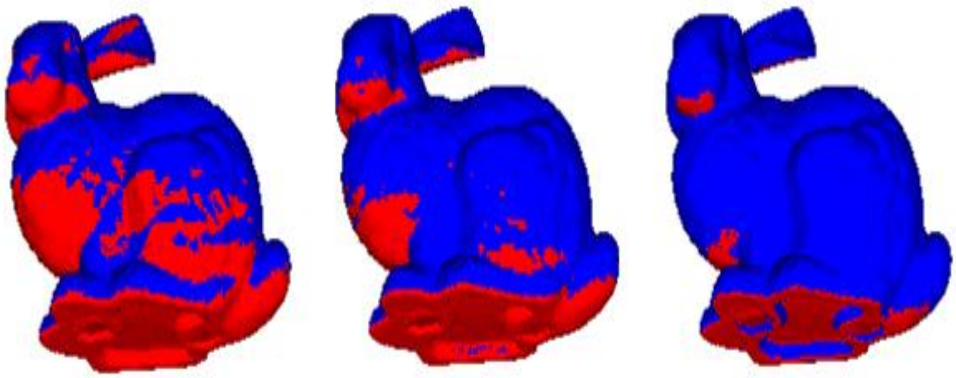

Figure 25. Support needed region, overhang angle of (a) 10° (b) 30° (c) 60°

4.2. Support amount estimation

By slicing process, tool-path of infill of 3D-object, support structure, and boundary of 3D-object are generated. By slicing process, we can obtain g-code, which has information of the toolpath of the nozzle along with the moving speed, size of deceleration and acceleration. Using these information, we can calculate time required to print a model with support and without support. Therefore, estimating the time required to print support structure is possible. However, generating support structure requires slicing process.

Depending on the size of points and faces which constitute the 3D-object, calculation might take long time.

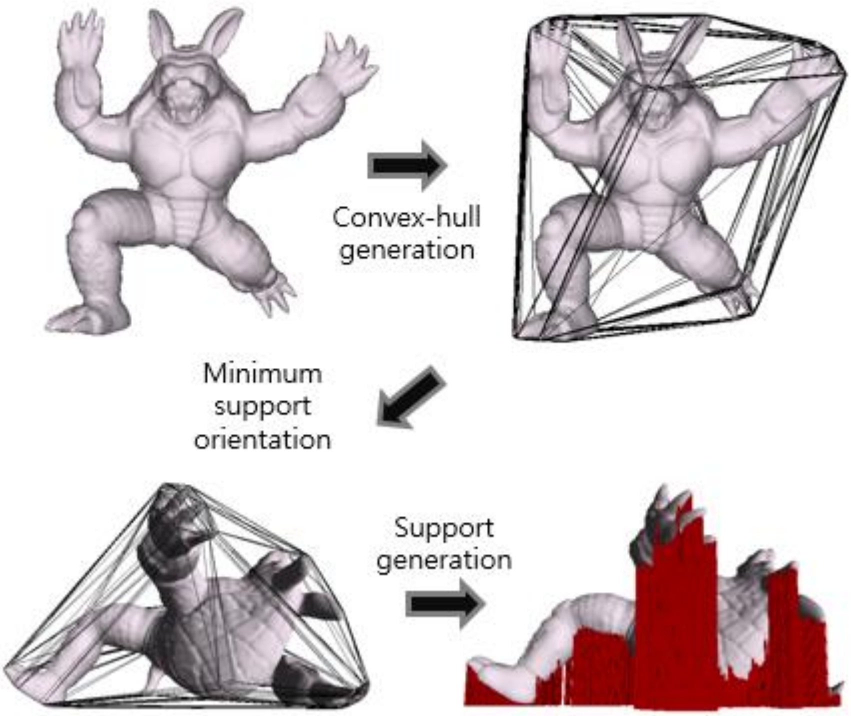

Figure 26. Overview of part orientation for minimum amount of support

However, we want to analyze the amount of support structure from many different orientations. Therefore, we use faster method, which only utilizes the geometry of the 3D-model. Although precise time estimation of support generation is not possible through this method, volume where support needs to be generated can be compared with other orientations.

4.3. Convex−hull algorithm

The amount of support, which is required to build a 3D−object, differs by the orientation of the 3D−object. In order to find the optimal orientation where the amount of support structure is minimal, analysis from diverse posture is required. However, analyzing from diverse orientation takes too much time. As in [1], by using the center of mass of the 3D−object and face from convex−hull, we can obtain stable posture for 3D−object, which requires less support. Then, from the reduced candidates made through the convex−hull algorithm, we can find the optimal orientation. (Figure 26) is the process of finding optimal orientation toward minimal support. We have generated convex−hull algorithm using [12].

Chapter 5. Result

We compared our algorithm with famous 3D-printer software' s support algorithms, which are Cura™' s line, grid, Meshmixer' s tree, and Simplify3D' s. We compared our algorithm with others in two methods. First method was tested by using compression testing instrument, INSTRON 5900R in (Figure 27).

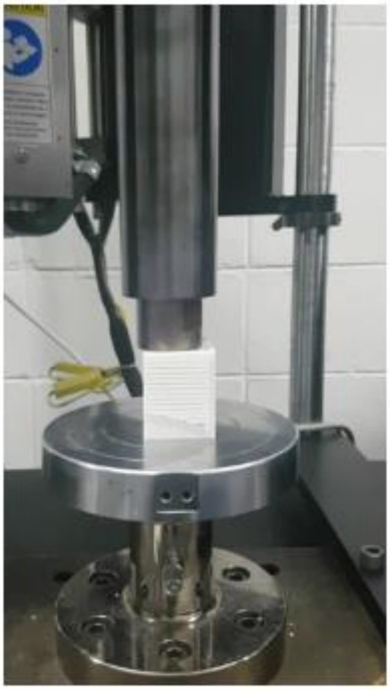

Figure 27. Compression testing instrument, INSTRON 5900R

The machine can measure precisely how much force is being applied to deform certain length. When people remove Cura™' s line, grid, and simplify3D' s support structure, they compress the support structure with their hand or tools like nipper. We have tested the force needed to eliminate the support structure with the same model. We have generated appropriate 3D model in (Figure 28.a) to perform compression test. (Figure 28.c) shows that support structure is twisted with the support algorithm of Simplify3D, while our algorithm shows shape consistency along vertical direction. Similarly, line support structure is twisted and has no deformable characteristic. Although cross section of support structure from simplify3D and line is similar to our algorithm, they lack deformable characteristic due to the twisted structure.

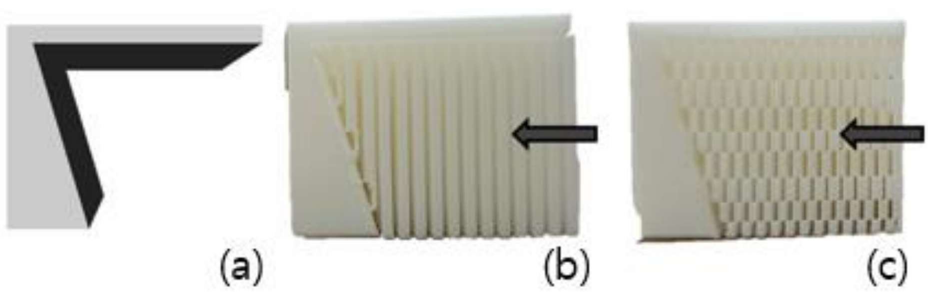

Figure 28. (a) Model (b) Our algorithm (c) Simplify3D

Using compression testing machine, INSTRON 5900R, we compressed the support structure along the direction shown in (Figure 28.b, c). Average force needed to compress the support structure was 217.7N, 130.8N, 22.4N in order of Line, Simplify3D, and our method as shown in (Figure 29). The testing machine measures the force needed to compress the model of certain length.

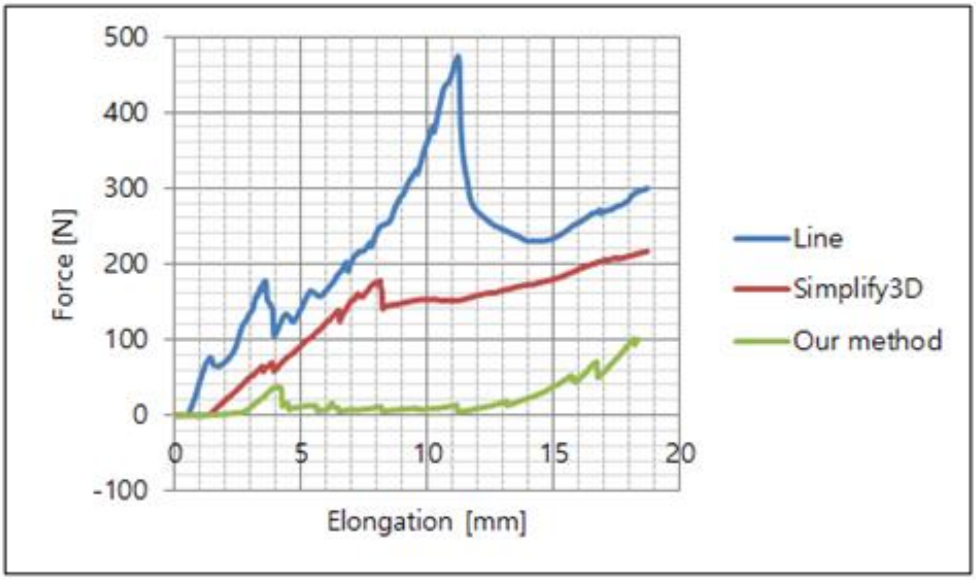

Figure 29. Compression test using INSTRON 5900R

Through the test, we have shown that connectivity and elastic property are main contributory factors of facilitating support removal with less effort. Although Simplify3D' s support is connected, they lack elastic property due to twisted figure as in (Figure 20.b, c, Figure 28.b, c). As the test result in (Figure 29), our method, which has both connectivity and elastic property, needs much less force to deform the same length. In this test, elongation equals the compressed length of support structure. As the structure is compressed, the support is eliminated from the 3D-object.

Comparing the result of Cura™' s line and Simplify3D' s, we could verify that deformability of support structure helps to eliminate support structure.

Figure 30. Cura™'s line, grid, Meshmixer's tree, Simplify3D, and Our algorithm (from left to right)

Some support structures like Cura™' s grid and Meshmixer' s tree are not appropriate for compression test. When removing these supports, compression is not the main contributory factor of support removal. Therefore, to compare these type of support structures with ours, we compared time required to eliminate support structures from same model as in (Figure 30). We repeated the test with different models. The test was done by a proficient FDM user. As seen in the graph in (Figure 31), our algorithm took less time to eliminate support structure. Ours was faster on average by 17.9%, 69.1% and 44.2% than line, tree and simplify3D.

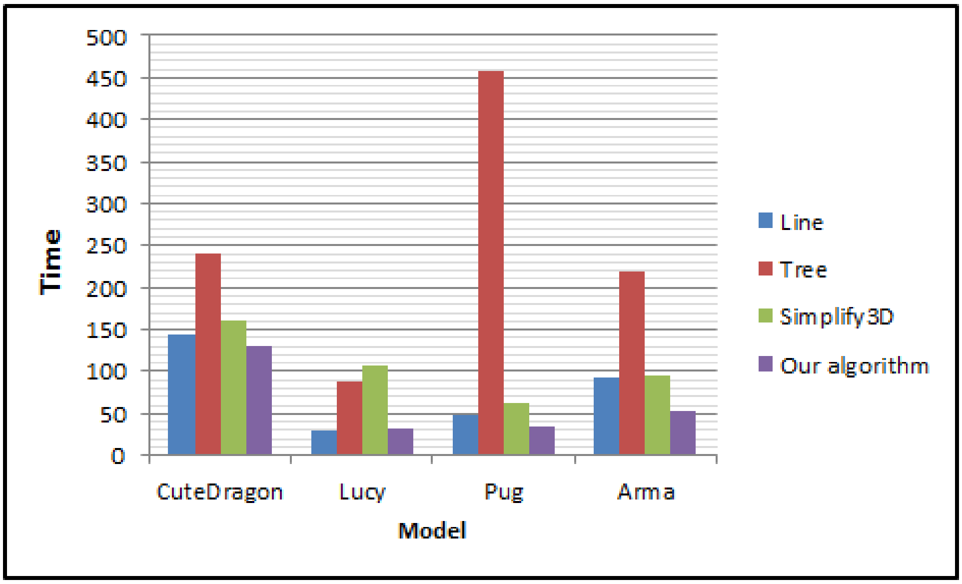

Figure 31. Removal time for different models with different support structures

Figure 32. Tested models (a) CuteDragon (b) Lucy (c) Pug (d) Armadillo

Chapter 6. Conclusion

We have shown the optimal process for support removal by generating deformable support structure. We have provided algorithm for optimal orientation that makes least support structure by analyzing diverse orientations utilizing convex−hull. By finding optimal orientation, support is reduced 0~100% depending on the geometry of the 3D−object.

Furthermore, we have generated optimal support for easy removal. We focused on generating support structure have deformable property. Through the compression test using INSTRON 5900R, we showed that support structure with deformability detaches better with much lower force compared to other support structures. Also, by the test in (Figure 31), we have shown that it takes less time to remove support structure.

Bibliography

[1] Stava, Ondrej, et al. "Stress relief: improving structural strength of 3D printable objects." *ACM Transactions on Graphics (TOG)* 31.4 (2012): 48

[2] Schmidt, Ryan, and Nobuyuki Umetani. "Branching support structures for 3D printing." *ACM SIGGRAPH 2014 Studio*. ACM, 2014.

[3] Vanek, Juraj, Jorge AG Galicia, and Bedrich Benes. "Clever support: Efficient support structure generation for digital fabrication." *Computer Graphics Forum*. Vol. 33. No. 5. 2014.

[4] Dumas, Jérémie, Jean Hergel, and Sylvain Lefebvre. "Bridging the gap: automated steady scaffoldings for 3D printing." *ACM Transactions on Graphics (TOG)* 33.4 (2014): 98.

[5] Wang, Weiming, et al. "Cost-effective printing of 3D objects with skin-frame structures." *ACM Transactions on Graphics (TOG)* 32.6 (2013): 177.

[6] Jiang, Caigui, et al. "Freeform honeycomb structures." *Computer Graphics Forum*. Vol. 33. No. 5. 2014.

[7] Zhang, Xiaolong, et al. "Medial axis tree—An internal supporting structure for 3D printing." *Computer Aided Geometric Design* 35 (2015): 149-162.

[8] Ezair, Ben, Fady Massarwi, and Gershon Elber. "Orientation analysis of 3D objects toward minimal support volume in 3D-printing." *Computers & Graphics*51 (2015): 117-124.

[9] Umetani, Nobuyuki, and Ryan Schmidt. "Cross-sectional structural analysis for 3D printing optimization." *SIGGRAPH Asia* 5 (2013): 1-4.

[10] Zhang, Xiaoting, et al. "Perceptual models of preference in 3D printing direction." *ACM Transactions on Graphics (TOG)* 34.6 (2015): 215.

[11] Thrimurthulu, K., Pulak M. Pandey, and N. Venkata Reddy. "Optimum part deposition orientation in fused deposition modeling." *International Journal of Machine Tools and Manufacture* 44.6 (2004): 585-594.

[12] Barber, C. Bradford, David P. Dobkin, and Hannu Huhdanpaa. "The quickhull algorithm for convex hulls." *ACM Transactions on Mathematical Software (TOMS)* 22.4 (1996): 469-483

초 록

본 연구에서는 FDM 방식의 3D-프린팅 출력물에서 서포트 구조를 쉽게 제거하기 위한 최적 방안을 제시한다. 서포트 제거는 계속해서 늘어나는 FDM 사용자들에게 꾸준히 불편한 점으로 언급되고 있는 문제이다. 일반적으로 서포트 제거가 쉽지 않은 경우가 많기 때문에 사용자들은 공구를 사용하여 서포트 구조를 출력 모델에서부터 제거한다. 서포트 구조를 제거하기 위해 많은 힘이 주어지는 경우에, 출력 대상이 부서지는 경우도 존재한다. 따라서 본 연구에서는 출력 대상으로부터 쉽게 제거되는 변형성을 지닌 서포트 구조를 설계하였다. 쉽게 변형되는 서포트 구조를 설계함으로써, 적은 힘으로 서포트 구조에 변위를 가하고, 서포트 구조의 변위는 출력 대상과 쉽게 제거되는 요인으로 작용한다. 또한 convex-hull 알고리즘을 이용하여 서포트 구조가 최소로 생성되는 자세를 제시한다. 이를 통해 본 연구는 FDM 사용자로 하여금 서포트를 제거하는 데 있어 최적의 방안을 제시한다. 본 연구에서는 제안하는 서포트 구조와 기존의 상용 소프트웨어 탑재된 서포트 구조를 압축 실험 도구 INSTRON 5900R을 이용하여 비교하고, 공구를 이용하여 각각 제거하는 데 걸리는 시간을 비교하여, 제안하는 서포트 구조의 우수성을 검증하였다.

주요어: 3D-프린팅, 서포트 구조, 제거, 변형성, 연결성, 출력 자세

학 번: 2015-20737