Clever Support for 3D Printing

CMPT464/764
Geometric Modeling in Computer Graphics
Course Project

Yiji Wang 301286922
Jack Anderson 301126642

Supervised by
Prof. Richard(Hao) Zhang

April 15, 2016

Abstract

In this project, we implemented a tree-like support structure for 3D printing based on Fused Deposition Modeling(FDM) from paper: Clever Support: Efficient Support Structure Generation for Digital Fabrication.[1] The overhangs pf the model need to be supported by connecting them with either part of the object or the printing layer using support material for 3D printing. Since the support material will be removed after printing, reducing the amount of the support can lead to less printing time and material waste. We implemented a novel, geometry-based approach that can minimize the support material used and provide sufficient support for the object.

Contents

1 Introduction

3D printing is becoming more and more popular in recent years with the general public due to lower prices and greater accessibility of 3D printers. There is less material wasted and greater model complexity when compared with traditional manufacturing methods like forging and CNC. In this project, we focus on the Fused Deposition Modeling(FDM) printers because they are the most common, inexpensive and widely used.

The printing process starts from the printing layer and then successively printing each slice until the printer reaches the top. Every 3D printer has a critical angle which means the maximum angle of a structure the printer can print without support. Therefore to print the overhangs, support is needed. Unlike other printers using cheap material, FDM printers use main printing material for support structures. Since the main printing material is used for supports, the support material usually costs a lot, nearly half of the model and the support material will be manually removed and cannot be reused. Most 3D printing softwares along with the printer generate the support structures by adding columns connecting the overhangs with printing layer or the model, leading to the amount of support material being unnecessarily large. 3D printing will become more accessible to the public with much less printing waste and printing time if less support structure is needed to be printed.

In this project, we implemented a novel, geometry-based solution to minimize the amount of support material. The input model was first oriented to a good position to reduce the area of overhanging parts. Then the sample points were detected from the overhanging parts and the support structure were iteratively built starting with these points. Instead of constructing a column for each overhang point, we built a tree-like structure to minimize the amount of support.

We compared the result of our implementation with the result provided by the author, the performace is close to the results from the Clever Support paper [1] with models having less features while not so good with models having a lot more details. The average difference is around 20%.

2 Implementation

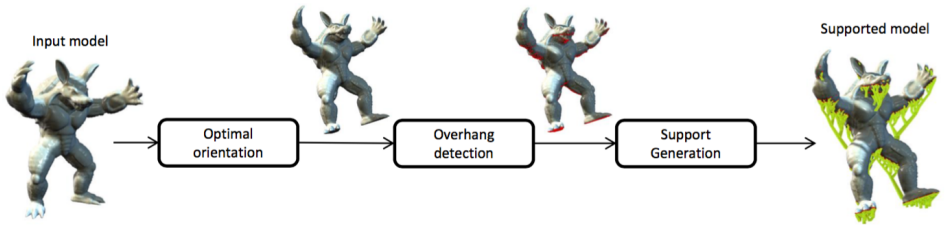


Figure 1: *The pipeline of the system. The input is a 3D model represented by a manifold mesh. Further inputs are: the number of random rotation angles, critical angle of the printer, the sampling distance and the thickness of the support structure.*

2.1 Optimal Orietation

In the first step, the model was rotated to a good orientation with less overhanging parts. Instead of using brute force method, which may give the best result but will lead to much more computation time, we used a simpler and faster heuristics provided in the paper.[2] First the number of random rotation angles K is given as input, we generated K different rotation angles. For each angle, we detected the overhanging points. To accelerate the process, we rotated the printing direction to find the best result among the candidates and applied rotation to the model for the best angle.[3]

2.2 Overhang Detection

The support structure is generated from overhanging points. To obtain the overhanging points, first overhanging parts need to be deteceted and then uniform sampling will be applied to the overhang faces/edges.

2.2.1 Overhang parts detection

There are three types of overhangs need to be detected. Figure 2 shows two of them.

- Point Overhang
  A point will be detected as point overhang if it's locally or globally lower than (in printing direction) all its neighbors.

- Face Overhang
  Face overhang is a triangular face. Faces will be detected as overhang faces if the angle between the face and the printing direction Y_p is

greater than the critical angle. As shown in Figure 2, face 1 doesn't need support because $\alpha_1 < \alpha_c$ while face 2 does for $\alpha_2 < \alpha_c$.

- Edge Overhang
  Edge detection is not used in our implementation because face and point overhang detection will cover the majority of cases where an edge needs to be supported.

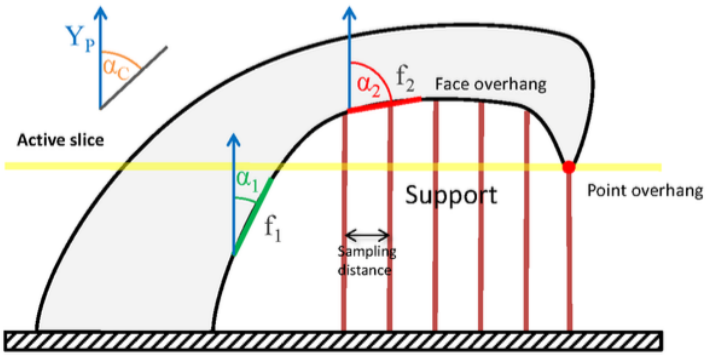


Figure 2: *Point overhang and face overhang example. Point overhang is any point which is lower than all of its neighbours relative to the printing plane. Face overhang is any face whose angle with the printing direction is larger than the critical angle.*

2.2.2 Uniform sampling

The support structure is generated from overhanging points. To obtain the overhang points, we did uniform sampling for the overhang faces and edges. The sampling distance is one of the inputs provided. Here we applied software scan-line rasterization algorithm.[4]

Algorithm is below:

1. Find the minimum and maximum value of z coordinate and x coordinate of the model respectively, denote them as $x_{min}, x_{max}, z_{min}, z_{max}$.

2. Apply scanning face starting from $z = z_{min}$, ending with $z = z_{max}$, moving along its normal $(0, 0, 1)$. If face f has intersection with $z = m$, apply the same scanning method for x coordinate moving along its normal $(1, 0, 0)$.

3. Get overhang points P from intersections of $z = m$, $x = n$ and face f

4. Check whether the interseciton point is in the triangle using method from [5]

5. Repeat step 2, 3 and 4 for every faces of the model

Figure 3: *Result of different sampling distances. The left one uses 0.1mm as sampling distance while the right one uses 0.25mm.*

2.3 Support Wireframe Generation

We defined the set of overhanging points to be P, the printing direction to be $\vec{v}$. To form the support structure, every point $p \in P$ should be connected to a point s. To ensure the printability of the support, the angle a between vector $\vec{sp}$ and $\vec{v}$ should be less than the critical angle α_c. The valid points lie in a support cone C from where the best s point will be found for each $p \in P$.

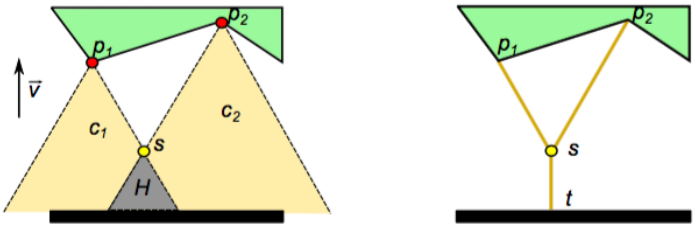


Figure 4: *There are two overhang points p_1 and p_2 with supporting cones C_1 and C_2. We denote the intersection area of two cones and the ground as H where the s point for p_1 and p_2 lies.*

For the support tree, we define a geometric graph $G = (V, E)$ where V is the union of three kinds of points $V = P \cup S \cup T$, where:

- P is a set of overhang points sampled on the mesh,
- S is a set of branching points connecting two parent points, and
- T is a set of points where the support intersects with model or the printing layer.

The goal is to find the T and S in order to minimize the total length of the tree. This problem is related to the Euclidean Steiner Minimial Tree(ESMT) in 2D identified by HWang and Richard[6], whose complexity is NP-complete. In this project, an ESMT is to be found in 3D space; thus the complexity is at least as NP-complete.[7] Therefore, in our implementation, we used a heuristic algorithm slightly different from that provided by the paper.[1] This algorithm is based on greedy strategy. It finds local minimum for every point $s \in Sands \in T$ with minimum Euclidean Distance iteratively and find interseciton of two cones that leads to the sortest tree branch.

2.3.1 Method to obtain intersection of two cones

Given two overhanging points $p_1(x_1, y_1, z_1)$ and $p_2(x_2, y_2, z_2)$ with their corresponding cones C_1 and C_2 and critical angle α. We can get a space H for the candidate connection point s (Figure 4) . We used a fast and simple method to obtain the best intersection point s.[8]

1. Find the projection points of p_1 and p_2 on plane $y = 0$[8]. Denote them as $p_1^{'}$ and $p_2^{'}$.
2. Calculate the Euclidean Distance $D = \sqrt{(p_1^{'} - p_2^{'})^2}$.
   - If $D \geq y_1 \cot(180 - \alpha) + y_2 \cot(\alpha)$, C_1 and C_2 have no intersection. $p_1^{'}$ and $p_2^{'}$ are set to be the children of p_1 and p_2(intersection with ground). Stop.
   - Else proceed step 3.
3. Find the plane P where p_1, p_2, $p_1^{'}$ and $p_1^{'}$ lie in. Note that the plane p is perpendicular to the printing layer.
4. Detect two lines l_1 and l_2. l_1 passes $_p1$ with slope $\tan(180 - \alpha)$ on P and l_2 passes p_2 with slope $\tan(\alpha)$.
5. The intersection s of l_1 and l_2 is the point connecting p_1 and p_2 with least distance.

2.3.2 Algorithm to build the tree

Given a list of overhang points P and corresponding cones C, we used the algorithm 1 to form the tree structure.

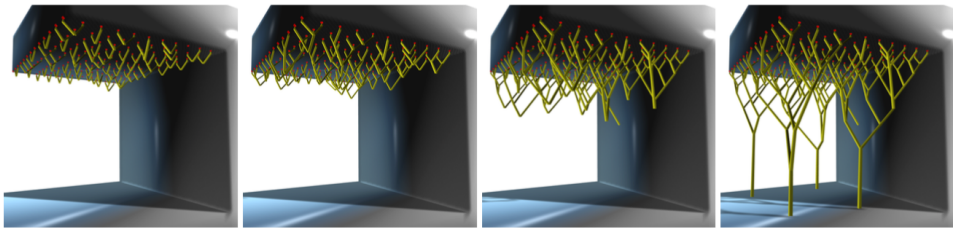


Figure 5: *The algorithm build structure progressively until all overhang points are support and the running time is reduced to* $O(n^2)$. *The most expensive part is to detect the intersection of branches and mesh.*

In the paper[1], they used GPU and depth buffer to find the intersections. In our implementation, we simply applied brute force method to detect every branch intersection with the mesh, which led to one of the limitations.

2.4 Support Structure Design

In the paper, several structures are proposed for support. Only N structure is accepted, because for one- and two-line structure, the support is not stable while for the closed profiles(circle, triangle and square), 3D printer tends to fill inside of them, leading to much more material used. N structure can support the square area with relatively less material. According to the paper the diameter of the N structure is determined by the formula: $d = k.l.\alpha$ where d is the struct diameter, α is the struct angle, and $k = 0.0015$ from experiments. At the end of support structure, tips are added for easier removed form the model after printing.

In our implementation, besides the N structure, we also implemented X structure which is not mentioned in the paper but may lead to less material used. This structure has the stability for printing and also can support the square area.

Algorithm 1: Algorithm to build ESMT besed on greedy strategy

Input : Overhang point list P and corresponding Cone list C
Output: Support point list S

define $D(p_i, p_j) = \sqrt{(p_i - p_j)^2}$
while *P is not empty* **do**
  for *each* $p_i \in P$ **do**
    find $p_j (i \neq j)$ where $D(p_i, p_j) = \min_{j=1}^{|P|} D(p_i, p_j)$
    if *C_1 and C_2 have no intersection* **then**
      set p_i and p_j as End Point
    else
      denote intersecion as s
      set s to be child of p_i and p_j add s into P as new overhang point
    end
    remove p_i and p_j from P
    add p_i and p_j into S
  end
end

for *each* $s_i \in S$ **do**
  if *s_i is End Point* **then**
    if *s_i is not termial point* **then**
      set new point $s^{'}$ as the projection of s_i on plane $y = 0$
      mark $s^{'}$ as termial point, set $s^{'}$ to be child of s_i
      add $s^{'}$ into S
    end
  end
  get child of s_i, denote as s_c
  if *segment(s_i, s_c) has intersection with mesh* **then**
    find interseciton $s_c^{'}$ that minimize $D(s_i, s_c^{'})$
    replace s_c with $s_c^{'}$
  end
end

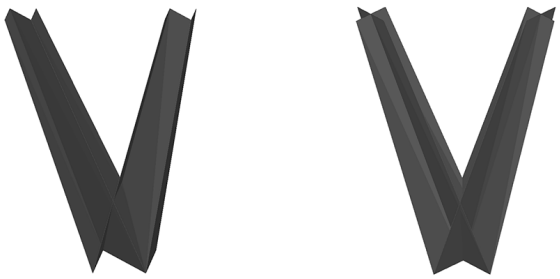


Figure 6: *The N(left) and X(right) structure with increasing weight for each level.*

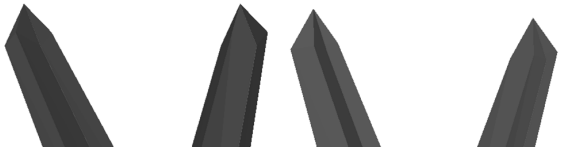


Figure 7: *Tips structure added for N(left) and X(right) strucutre for easier removal of support material.*

3 Summary

In this project, we implemented an optimization framework attempting to minimize the supporting structures needed for 3D printing objects with overhangs. The input is a 3D model represented by a manifold mesh. The model is first oriented in to a better position leading to less overhangs. Uniform sampling is applied to get the overhang points for the support structure which are also leaves of the support tree. Then the tree-like support structure is generated progressively using greedy strategy. At the end N/X structure is used to wrap the tree. And the output is obtained which is a rotated 3D model with support structure also composed of meshes.
Appendix shows output of the program.

3.1 Result and Evaluation

With some constraint, we were not able to print every model we implement with a 3D printer. Thus we used a self-defined evaluation method. The support structure will be composed of faces and vertices. And the amount of support material used is directly related to the area of the support faces, which can be calculated as $V = \sum_{i=1}^{|F|} Area(F_i)$ where F is the set of support faces.

We performed evaluation based on the model with support provided by the author.

Model	Size	Ref	N Struct	Diff(N)	X Struct	Diff(X)
arches	small	9850.90	10197.90	3.52%	9333.91	-5.25%
armadillo	small	7001.35	6768.18	-3.33%	6196.41	-11.50%
bird	small	4463.55	4862.58	8.94%	4903.30	9.85%
dragon	large	7274.38	4944.01	-32.04%	4575.10	-37.11%
mol_thick	large	9756.02	7602.45	-22.07%	7005.44	-28.19%
tree	large	14713.04	6834.60	-53.55%	6145.85	-58.23%
wheel	large	12551.00	17507.98	39.49%	15948.13	27.07%
Average(all)			23.28%		25.31%	
Average(small)			5.26%		8.87%	
Average(large)			36.79%		37.65%	

Table 1: The amount of support material used in our implementation and the result provided by the author(Ref). The orientation is the same as the reference model by removing the support from the input file. Parameters: critical Angle $\alpha = 50°$, sampling distance $d = 0.25mm$, thickness of material $t = 0.1mm$. The difference is below 10% for models with less details while larger for those with a lot more details. This is mainly caused by the difference in the algorithm applied to build the tree and the parameters setting from the author.

3.2 Limitation and Possible ways of improvement

1. Cone-to-mesh intersection detection
   We did not use the GPU and depth buffer to detect the intersection of support with mesh/printing layer. We instead used a brute force method and computed the intersection with whole mesh for every cone. This method is slow and expensive. To improve this, we can apply parallel computing on GPU, which computes the intersection of the cone with each face of the mesh in parallel.

2. Support structure intersection with the model
   We checked the intersection of cones with the mesh during building the support wireframe. If a branch of the tree is very closed to the mesh, it will not be detected to be intersected with the mesh. However, after wrapping the branches with N/X structure, there may be intersection between support and the mesh since each point on the branches is expanded to an area with diameter d. We did not detect the intersection because it's very computationally expensive to detect intersection for every support face with the mesh. This may cause more material used

and unprintability. To improve this, we may apply same method as in Cone-to-mesh intersection detection.

3. Printability of the result in real case
   The output of our implementation will be a rotated model with support structure. We did not check whether the support structure will be able to withstand the weight of mesh or shear forces during printing.[1] We may first use some printing software for testing and then print the result using 3D printers.

4. Global minimum not guaranteed
   To generate the Euclidean Steiner Minimal Tree(ESMT), we used heuristic algorithm which leads to local minimum. Global minimum cannot be guaranteed because finding the ESMT with global minimum is NP-complete. Currently we cannot find out a way to improve this.

We believe that most of the limitations could be addressed as future work of this project. Futhermore, besides having bi-furcation for the tree, we may have mutiple brahches which may save more support material.

References

[1] J. Vanek, J. A. G. Galicia, and B. Benes. Clever support: Efficient support structure generation for digital fabrication. *Eurographics Symposium on Geometry Processing*, 2014.

[2] Dutta D. Alexander P., Allen S. Part orientation and build cost determination in layered manufacturing. *Computer-Aided Design*, 1998.

[3] Rotation about an arbitrary axis in 3 dimensions. http://inside.mines.edu/fs_home/gmurray/ArbitraryAxisRotation/. Accessed: 2016-03-20.

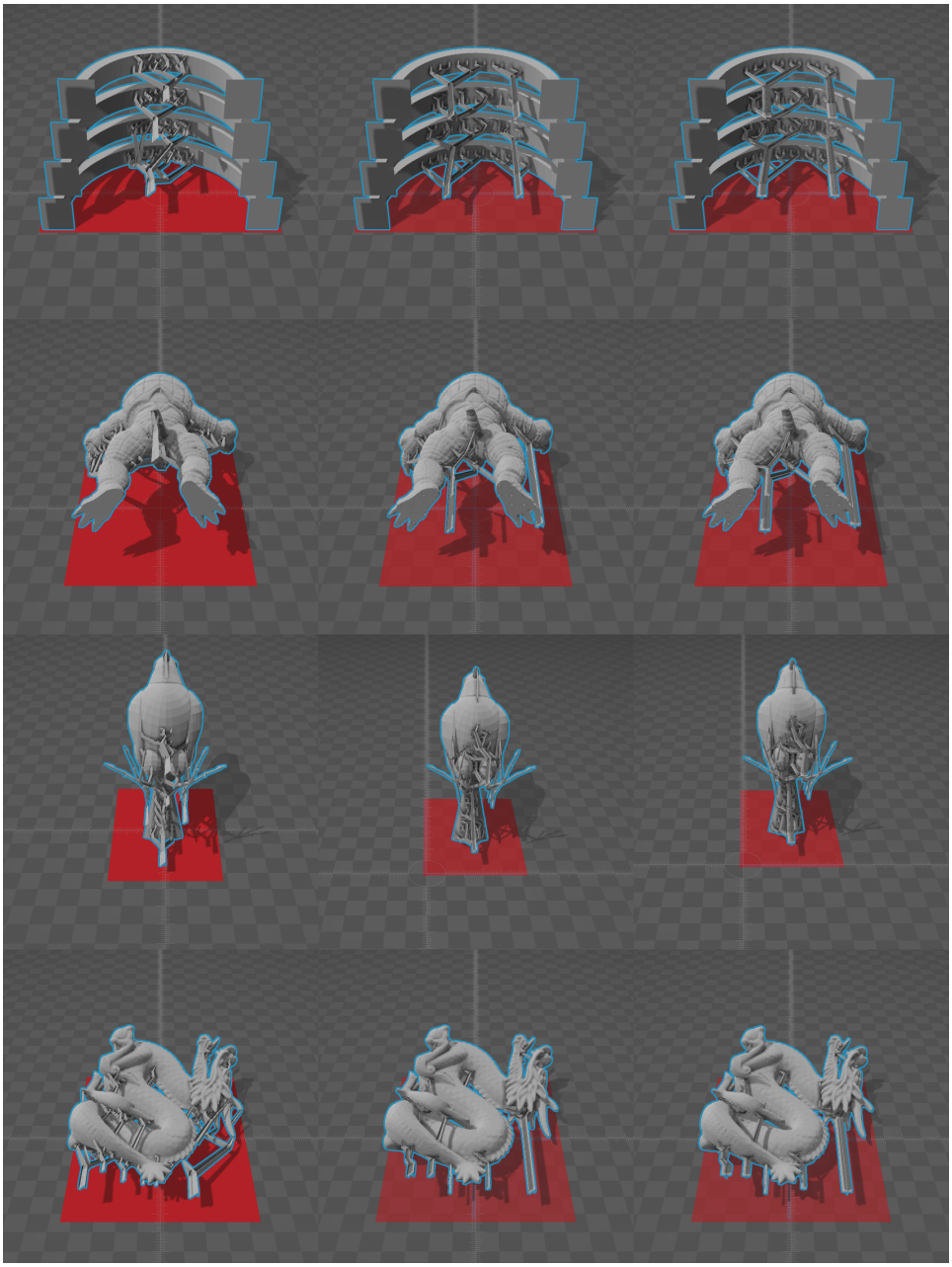
[4] Software rasterization algorithms for filling triangles. http://www.sunshine2k.de/coding/java/TriangleRasterization/Triangle-Rasterization.html. Accessed: 2016-03-16.

[5] Determine if projection of 3d point onto plane is within a triangle. http://math.stackexchange.com/q/544947. Accessed: 2016-03-12.

[6] Richard D. S. HWang F. K. Steiner tree problems. *Networks*, 1992.

[7] Pollak H. Gilbert E. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 1968.

[8] A general principle for the construction of points of the intersection of two cones. http://www.grad.hr/geomteh3d/prodori/prodor_stst_eng.html. Accessed: 2016-03-12.



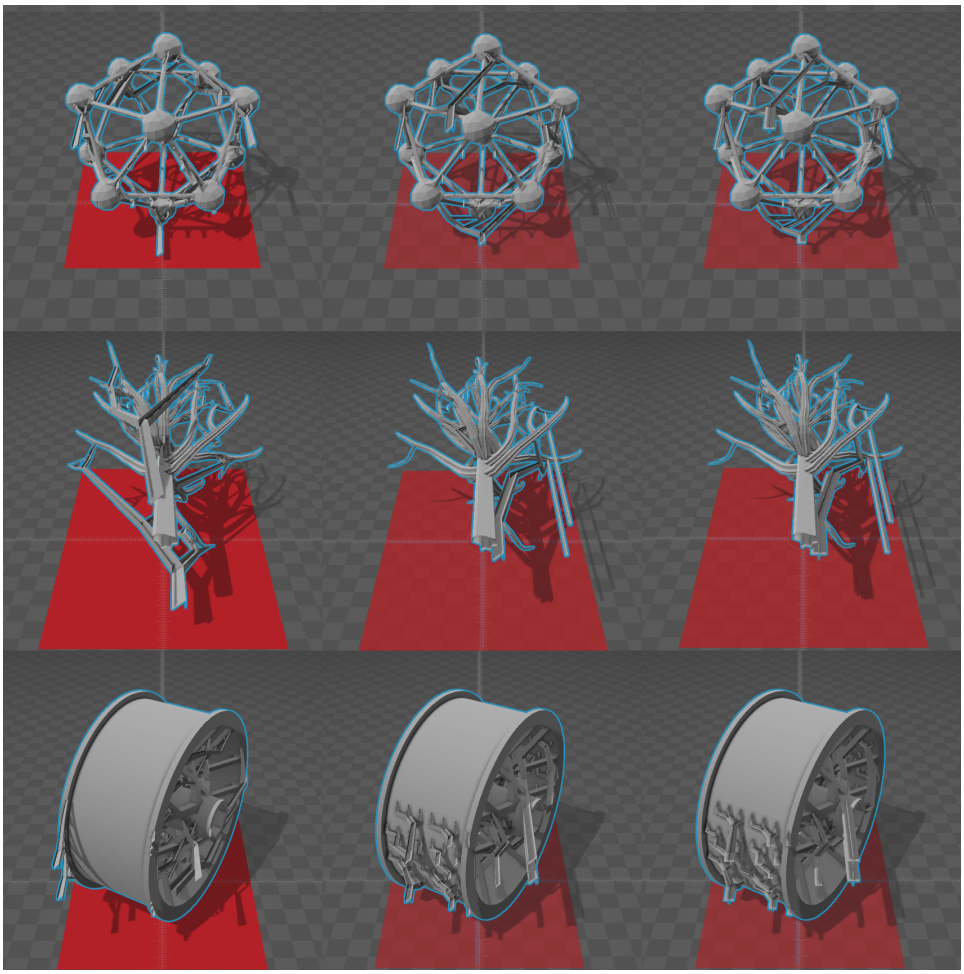


Figure 8: *Left:Result provided by author, Mid:Our result with N structure, Right:Our result with X structure*