INF574 Project Report - Shape Analysis and Optimization for 3D Printing

Wenqing QU Ranyu FAN

Voxelized Internal Carving & 3D Printing Balancing

- Experiments and reflections on the paper "Make It Stand: Balancing Shapes for 3D Fabrication"

1 Introduction

3D printing technology is widely used in various fields today, and many designers will use 3D printing to assist their work. However, the 3D printed models are unbalanced in many cases. Other means such as adding a base and supports are usually required to keep it balanced on the table. To solve this problem, we read the article "Make It Stand: Balancing Shapes for 3D Fabrication" and reproduced the method in the article.

2 Problem Modelling

We have made some simplifications after studying the model in the original paper. In short, we can calculate the yellow part of the figure, the base of support, after we have a model and a defined horizontal plane. This part is actually a polygon defined by the convex hull of all the points of the mesh touching the ground. Therefore, in order to keep the model balanced, we hope that the projection of the center of gravity in the direction of gravity is within this yellow polygon. To be on the safe side, we can manually restrict a smaller polygon, which is the blue polygon in the picture, so that the model is more stable.

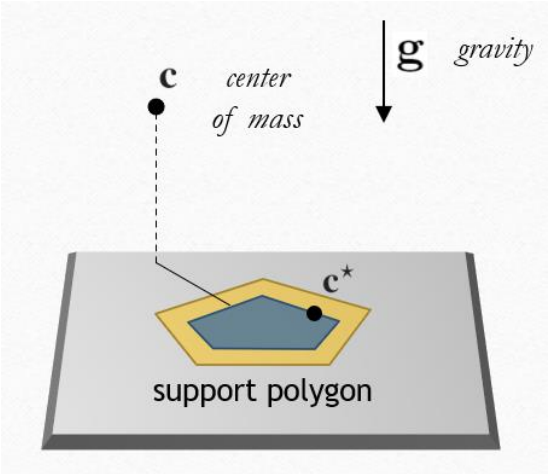

Figure 2.1 Support polygon (Source: "Make-it-stand")

3 Internal Carving/Sculpting and Voxelization

Obviously we need to change the original center of gravity of the model to achieve the goal. We can sculpt a part of the interior of the model into a hollow state to change the original center of gravity. For the sake of simplicity, using voxels for internal sculpting is a very suitable method - it greatly reduces the search time of the program.

4 Implementation and Testing

We decided to use Python for implementation. Our program will first read the file with the suffix .obj under the specified path and store the data in it. The user can manually specify a plane (by giving the coordinates of three points) as the horizontal plane, and the program will then plug the data into the model and calculate the center of gravity. After verifying that it cannot be balanced, the program will try to perform voxelized sculpting from the inside and calculate a new center of gravity until the new center of gravity is within the balanceable range. Finally, the program will output the balanced file in stl format. If a certain number of attempts are unsuccessful, the program returns an error and advises the user to change the plane definition or model shape.

4.1 Problems and solutions

However, we have encountered some difficulties in the implementation process.
According to our approach, after the user has defined the support plane, the program should check whether the imported model intersects the user-defined support plane and return an error if it does not, while if it does intersect it will cut off the part below the support plane and fill in the hollow surface created by the cut.

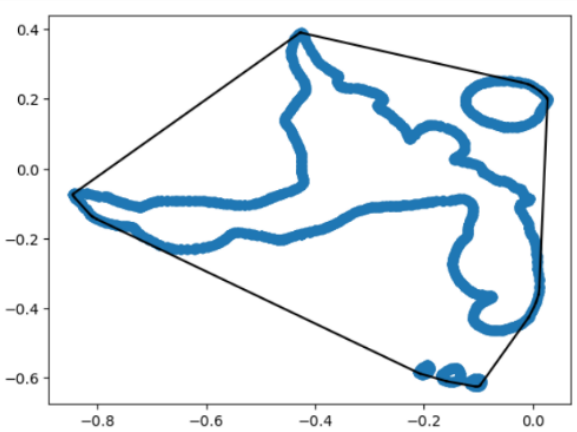

Figure 4.1.1 Profile of the cut surface of the gargoyle model

For example, for a spherical shell, we would cut off the part of the shell that lies "below" the support plane (note that the direction of the normal vector of the plane can be changed according to the user's needs). The spherical shell will then only be half left and a gap will be created where it was cut off - this makes it no longer a closed figure. So we need to get the program to generate a new plane to fill this in.

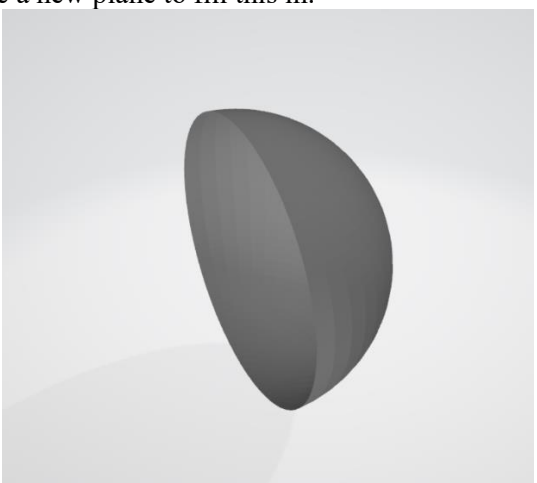

Figure 4.1.2 Spheres successfully pre-treated for voxelization

But the fact that we are reading a triangle mesh makes it more complicated to deal with the cut. We need to consider what happens when a triangle face is cut - obviously in most cases it will be split into a smaller triangle face and a quadrilateral face. At this point we first need to determine which of these two parts is above the plane of support, i.e. the part that needs to be preserved. If the part to be preserved is a triangle, we can delete the original triangle and store the new triangle in the np array of the triangle mesh as it is. For a quadrilateral, we need to convert it into two new triangle faces before storing it. In our practical tests, however, the results were not very satisfactory when given certain support planes. After further investigation we found that this was actually due to the accuracy of the calculation - when the length of the intersection line between the triangle face and the cutting plane was too

short, our program would no longer be able to distinguish between the two intersections and would consider them to be the same point, i.e. there was only one intersection point between the triangle face and the plane. The program then retains the original triangle, resulting in the appearance of what we call a "shark's tooth" serration.

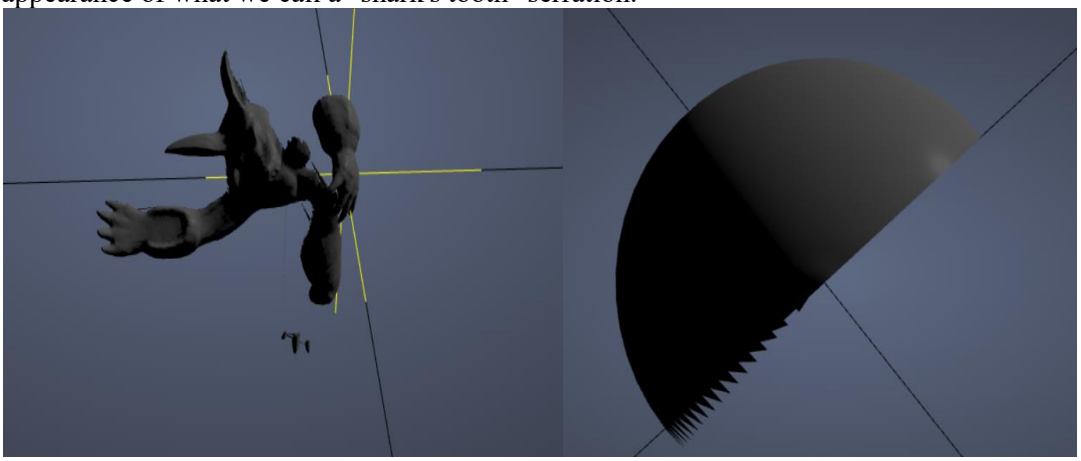

Figure 4.1.3 Failure cases

Unfortunately, we were not able to improve our code on this approach to avoid the bug, and we had to make the difficult decision to abandon all the functionality we had already done in the program in favor of a completely new approach to the problem.
Specifically, we improved all the steps from the definition of the support surface to the point where the model is ready for voxelization. Instead of using the defined support surface to split the original model directly after the user has given it a definition, we treat the support surface and the original model as a new model. We first rotate and translate the entire new model containing the support surface so that the support surface coincides with the xOy plane. At this point we first voxelize the whole model with a high degree of accuracy and then cut the model later. This approach avoids the problems encountered when dealing with triangle mesh and keeps the loss of accuracy within limits. At this point we realize that voxelization can be used not only for internal sculpting, but also to simplify the pre-processing of the model. The sequence of rotational translation followed by voxelization ensures that voxels are always removed 'in one layer' when cutting, making it easier to code. Furthermore, with this method we no longer need to fill in lost planes.

4.2 Procedure flow

In the end our complete program is written according to the following logic.

1. read the .obj file specified in the Input folder and store the data
2. read the coordinates of the three points entered by the user to determine the support plane
3. rotate and translate the support plane and the original model until the support plane coincides with the xOy plane
4. voxelize the rotated model to a high degree of accuracy
5. read the direction of the normal vector of the support plane set by the user, cut the model and remove the voxels below the support plane.
6. Calculate the center of gravity of the cut voxelized model and the supporting polygons.
7. Using the center of gravity calculated in the previous step as a reference, the model is scaled down to a certain proportion while maintaining the center of gravity as the internal sculpting area.
8. Make the internal sculpting area slightly less accurate by voxelizing it to simulate the accuracy of a 3D printer.
9. Find the voxel in the internal sculpting area that is furthest away from the center of gravity, attempt to remove it and verify that the new center of gravity is within the supporting polygon after removal. If it is within the polygon, the engraving is complete;

if not, repeat the above steps until a certain number of times have elapsed without a solution, at which point the program returns a failure message and advises the user to change the model or support plane definition.

10. Export the final result file to the Output folder.

4.3 Results

We tested our program using the same test models as in the article (spheres and gargoyles respectively) with the following results.

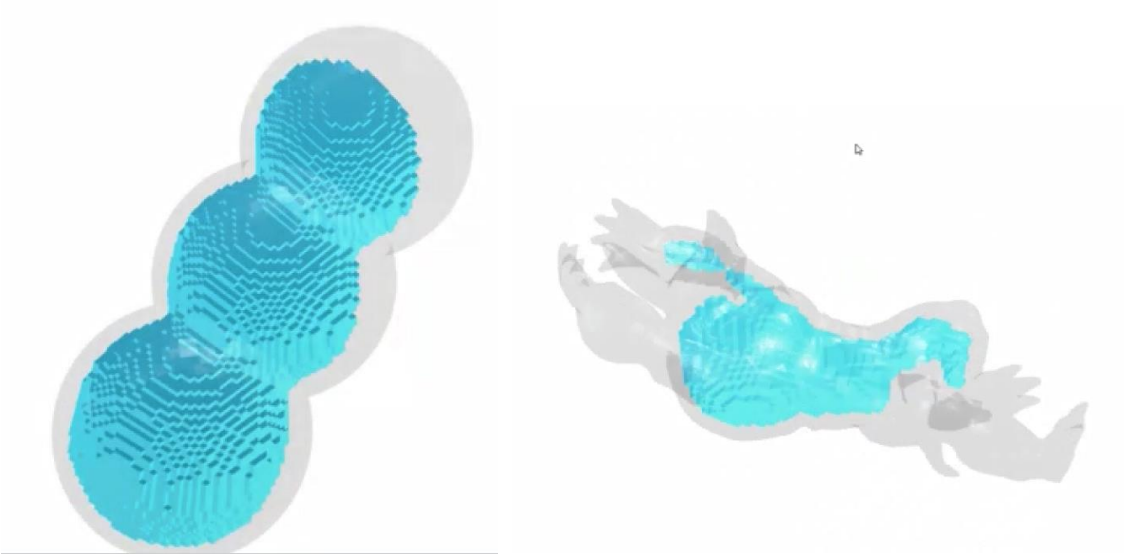

Figure 4.3.1 Results using "spheres" and "gargoyle" as input models

It can be seen that our program correctly outputs its internal voxelized model and selects a part of it for sculpting to balance the model.

5 Reflection: Deformation? Other methods?

We recognize that in many cases voxelization alone is not enough. In the article they also used deforming for balance. But we think this method is not only very complicated to implant, but also has some problems: obviously it will change the appearance of the original model, which is not allowed by designers in many cases; on the other hand, As revealed in the article, sometimes the shape-changing algorithm will cause the original model to collide internally, which will inevitably cause the loss of details. Therefore we hope to find another way to optimize our solution without changing the shape of the model in order to solve the balance problem in more cases.

5.1 Multi-material 3D Printing

Through further research, we learned that many 3D printers are now capable of multi-material 3D printing. This inspired us to use higher density 3D printing materials to fill the unengraved part while engraving inside, thus increasing the offset of the center of gravity.

5.2 High Density Material Filling

PLA or polylactic acid is one of the common FDM 3D printing materials. PLA has a density of 1.24 g/cm3 in most manufacturers. However, when PLA is mixed with materials such as metal or wood, the density changes. Generally speaking, the density of metal-mixed PLA can reach up to 4 g/cm3, which is an ideal filling material. Therefore, we can improve the original program on this basis. When the balance cannot be guaranteed only through sculpting, the program will perform voxelized sculpting inside the entire model, and then fill it with a higher density material to try to achieve balance.

Figure 5.2.1 Result with high density filler

The green part of the diagram represents the evacuated section, while the blue part represents the high-density filler.