CSC4140 report

Final Project: Point Cloud to Mesh

May 23, 2022

Point Cloud to Mesh

Point Cloud to Mesh using BPA and Poisson Surface Reconstruction

Student ID: 119010344

Student Name: Xiao Nan

This report represents my own work in accordance with University regulations.

Signature: Xiao Nan

Contents

1 Mesh evaluation metrics used **3**

1.1 Smoothness 3

1.2 Aspect ratio 3

2 Ball-Pivoting Algorithm **3**

2.1 Principle 3

2.2 Running time and performance results 4

3 Poisson Surface Reconstruction Algorithm **5**

3.1 Implementation 5

3.2 Running time and performance results 5

4 Acknowledgements **6**

1 Mesh evaluation metrics used

1.1 Smoothness

The smoothness is defined as the average among all faces of the ratio between a given face's area as well as one of its neighbour face's area. The more the smoothness is closed to 1 (that is, the area of the current face as well as the adjacent face are closed), the more likely that the mesh looks smooth and has fewer "jumps" on the triangular faces' sizes. A large smoothness, corresponding to a large differences in size between adjacent faces, will result in a poor mesh because the smoothness of the mesh around that face's region is destroyed.

1.2 Aspect ratio

The aspect ratio of a triangle is defined as the length of the longest side of the triangle divided by the length of the shortest side of the triangle, indicating the irregularity of a triangle. The more the aspect ratio is closed to 1, the more likely that the mesh has regular shapes with respect to any one of the triangle (that is, the per-triangular shape is more uniform and regular). On the other hand, when the ratio is too large, the mesh is consider to be highly irregular (in the triangular shape) and one may use techniques like the mesh regularization to reduce such irregularity.

2 Ball-Pivoting Algorithm

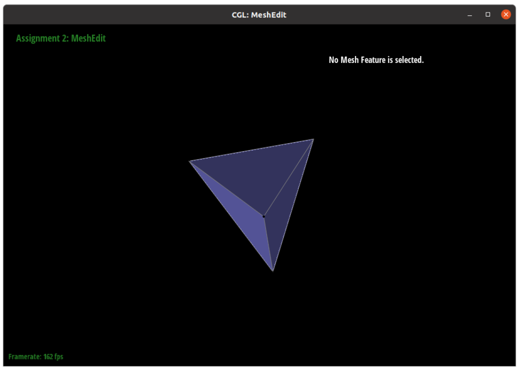
2.1 Principle

The BPA consists of two phases mainly: initialization phase and the loop phase. During the initialization phase, a seed triangle is found (by looping all vertices, trying forming the triangle and selecting the one with no more points inside its circumsphere). After that, the program run into the while loop, in which the program expands the current seed triangle (the currect frontier edge) by keeping poping active edges and pivoting on the given edge with a ball of a given radius rou. The expansion ends when no more points can be hit by the ball and then another seed triangle is selected to continue the next iteration loop. The seed-finding and expanding loop continues till there's no more seed triangle available and then the reconstructed mesh is completed.

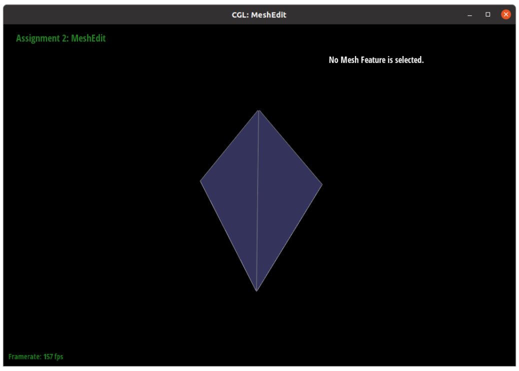
2.2 Running time and performance results

The resulting for running the BPA with the input .ply file as a tetrahedron and a buddha statue are shown in figure 1, in correspondance to the figures' order. We may observe that for simple objects like the triangle, the BPA achieves a satisfactory performance in that it has very few faces to connect and render. But for complex object like the buddha statue, the BPA provides a results that has more holes. This may be due to the fact that the ball radius is set so small that some of the vertices are not discovered by the loop and hence are ignored during loop expansion.

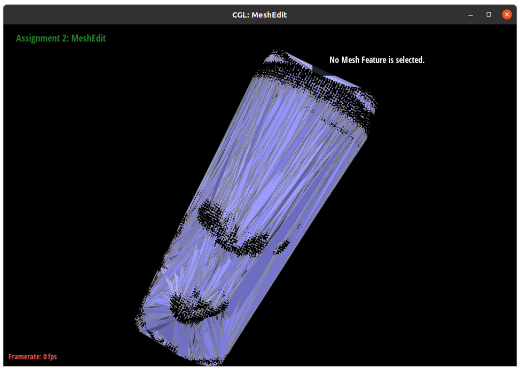
The resulting running time, smoothness and aspect ratio of the reconstructed buddha mesh are shown in figure 1(d), where we can observe that the smoothness and aspect ratio reach a value that is closed to 1 enough. The smoothness approximating equal to 3 indicates that the extent to which the mesh achieves a balanced surface area (that is, each triangle having similar area and there would not be "leap" in triangle area). The aspect ratio approximating 3 indicates that the mesh are not severely deformed (that is, the average irregularity of a particular triangle in the mesh is not too large to be unsatisfactory). The long running time approximating 77 seconds show the limit of speed of the BPA.



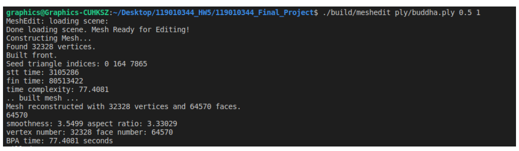
(a) Reconstructed triangle



(b) Reconstructed triangle from another view angle



(c) Reconstructed buddha



(d) Reconstructed buddha execution statistics

Figure 1: Reconstructions using BPA

3 Poisson Surface Reconstruction Algorithm

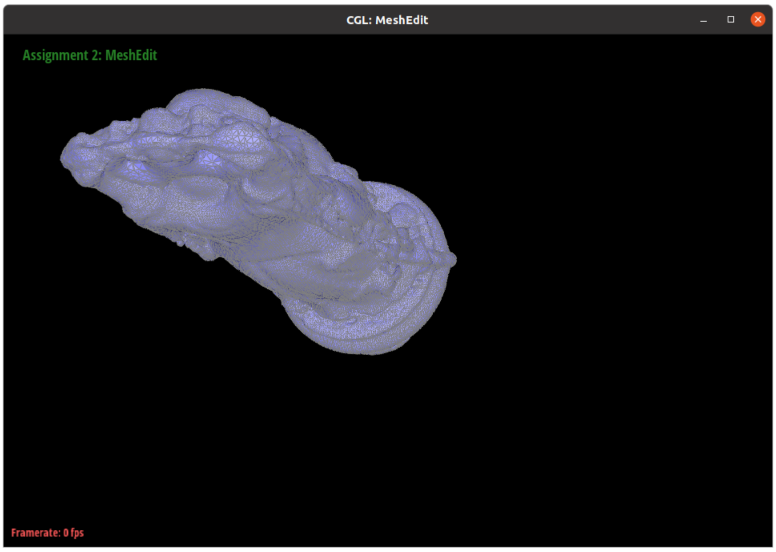
3.1 Implementation

The Poisson Surface Reconstruction Algorithm is implemented based on the PCL, a powerful library that supports a standalone, large scale, open project for 2D/3D image and point cloud processing with various APIs for use. Hence you may first install the libpcl-dev library on the virtual machine before running the PSR on a given .ply file.

3.2 Running time and performance results

The resulting for running the PSR (Poisson Surface Reconstruction) algorithm with the input .ply file as a buddha statue, as well as the metrics evaluated on that, are shown in figure 2, in correspondance to the figures' order. We may observe that for complex object like the buddha statue, the PSR provides a results that is seemingly more smooth and more realistic (which renders the face and hand details of the buddha lively) than the mesh constructed by the BPA, and has almost no hole. Also, the resulting running time (forget printing out in the figure) is only around 29 seconds, a time that is around 3 times faster than that runned by the BPA. Hence we may conclude that the PSR outperforms the BPA in both the execution time and the resulting mesh's smoothness (continuity) as well as regularity.

The only potential limitation of the Poisson Surface Reconstruction Algorithm compared to the BPA is that the number of triangles required for the mesh is large, making the memory consumption of obtaining such mesh too high. Hence such algorithm may be unsuitable to applications where the resolution is not required necessarily to be high and the device memory is limited.



(a) Reconstructed buddha

(b) Reconstructed buddha execution statistics

Figure 2: Reconstructions using Poisson Surface Reconstruction

4 Acknowledgements

Thanks to Dai Yulong (119010054), who helped me understand the principle of Poisson Surface Reconstruction and gave me some hints of using the PCL library for implementation. Thanks to Wang Yupeng (119010322), who discussed with me issues on the ball pivoting operations, and supported me when I was deeply confused and depressed by the pivoting operations :).