Fall 2015

Report for Yunjoo Park

Yunjoo Park

1 Implementation Details

1.1 Problem-solving methods and algorithms

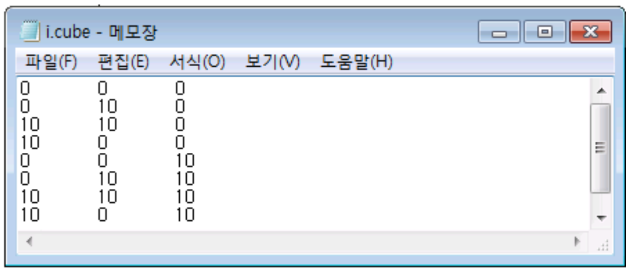
The *crust algorithm* is a algorithm for the reconstruction of surfaces of arbitary topolgy from unorganized sample points in 3D[2]. The input to the algorithm is a set S of sample points from the surface of a three-dimensional object[1]. Let $P := \{p_1, p_2, ..., p_n\}$ be a set of 3D points in the plane. For the crust, firstly, we have to compute the *Voronoi diagram* V_p of the sample points P. A Voronoi diagram is a partitioning of plae into regions based on distance to points in a specific subset of the plane. The Voronoi diagram is dual to its *Delaunay triangulation*. For each sample point p, find the two vertices of V_p fathest from p. One is a pole and the other is antipole of s. And then, union the sample points and poles. For the union, compute the *Delaunay Triangulation*. From the triangulation, discard some triangle with a pole as a vertex and keep only triangles for which all three vertices are sample points in P.

1.2 Platforms

Languages: C

Platform: Microsoft visual Studio 2013

1.3 Examples of Inputs



```
0	0	0
0	10	0
10	10	0
10	0	0
0	0	10
0	10	10
10	10	10
10	0	10
```

Figure 1: An Example of inputs

1. There are 12 test data.
2. Each data file has different values.
3. Each line is a 3D point (x, y, z).

1.4 Problem-solving methods and algorithms

For lifting the given points to 4D, add another point $pt4 = x^2 + y^2 + z^2$. For new set of points, I compute a voronoi diagram by using qhull[3] and get a voronoi vertices of facet lowerdelaunay. Then, find a pole and antipole for all points. If a point o lies on the convex hull, the pole vector $\vec{po}$ is the average of the outer nomals of the adjacents. On the other hand, if a point does not lie on the convex hull, the pole is the farthest voronoi vertex of voronoi vertices from the point. And then, compute the other pole, antipole. An antipole is a vertex farthes away from the point o in direction of $\vec{op}$. By using inner product, find the antipole. And then, add poles and antipoles into *vertices*. For the new set *vertices*, compute the Delaunay Triangulation. The poles have a member variable, *ispole*. By using this value, report triangles whose vertices are all from original set P.

2 Example Output

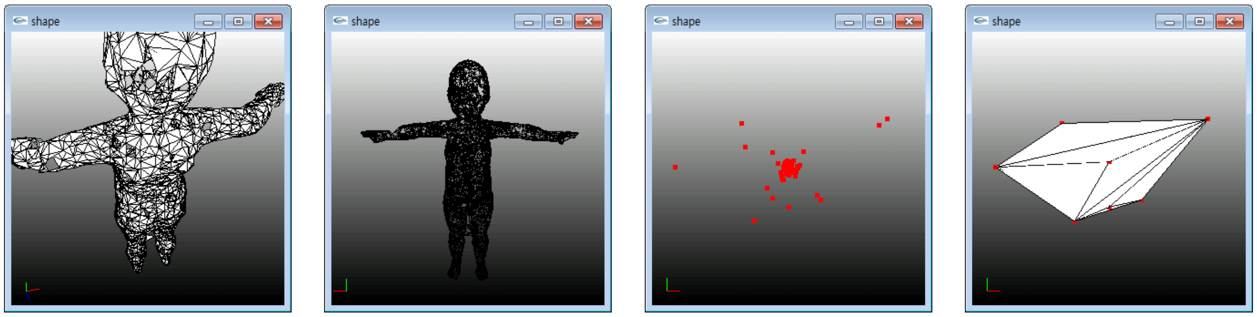


Figure 2: i.bb

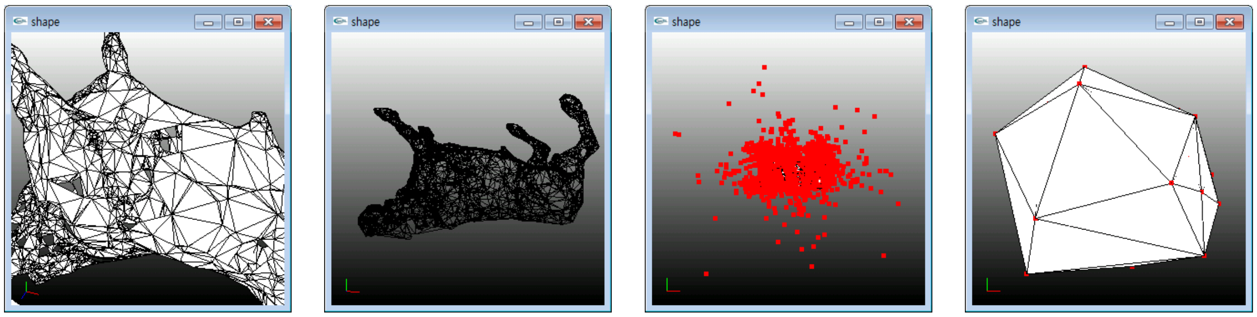


Figure 3: i.bull

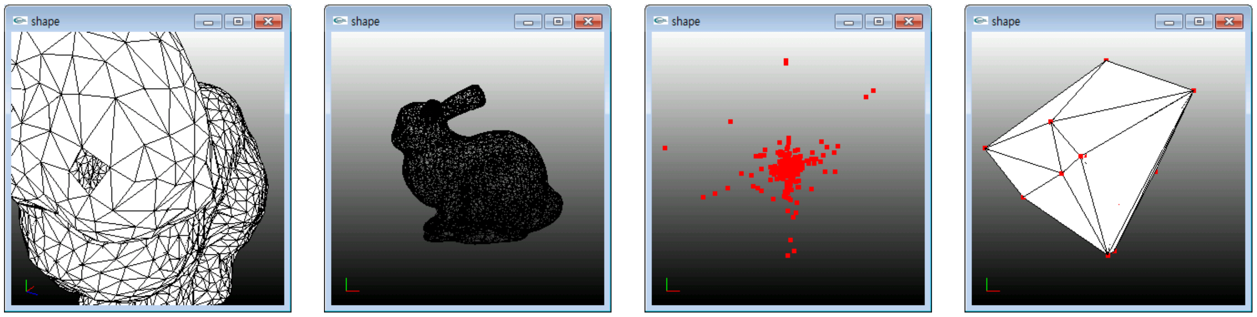


Figure 4: i.bunny

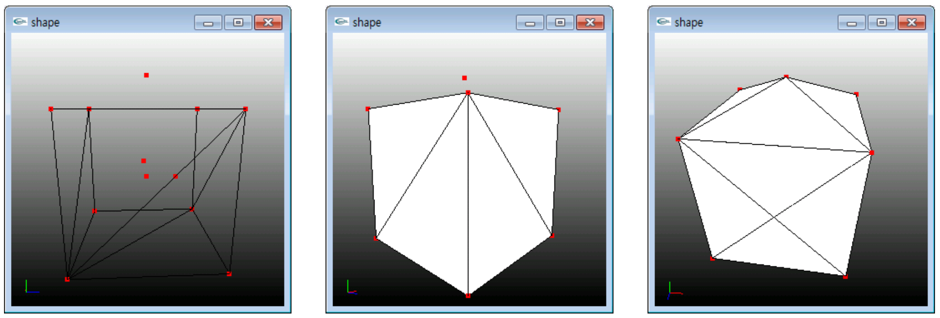


Figure 5: i.cube

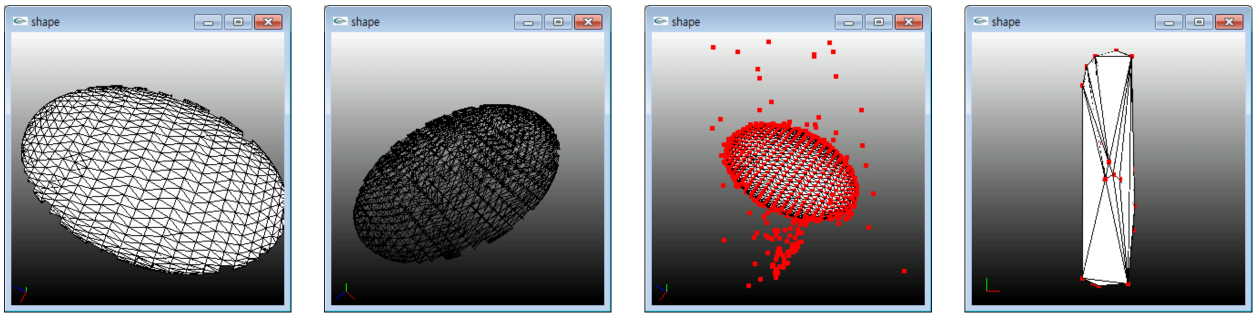


Figure 6: i.ellipsoid

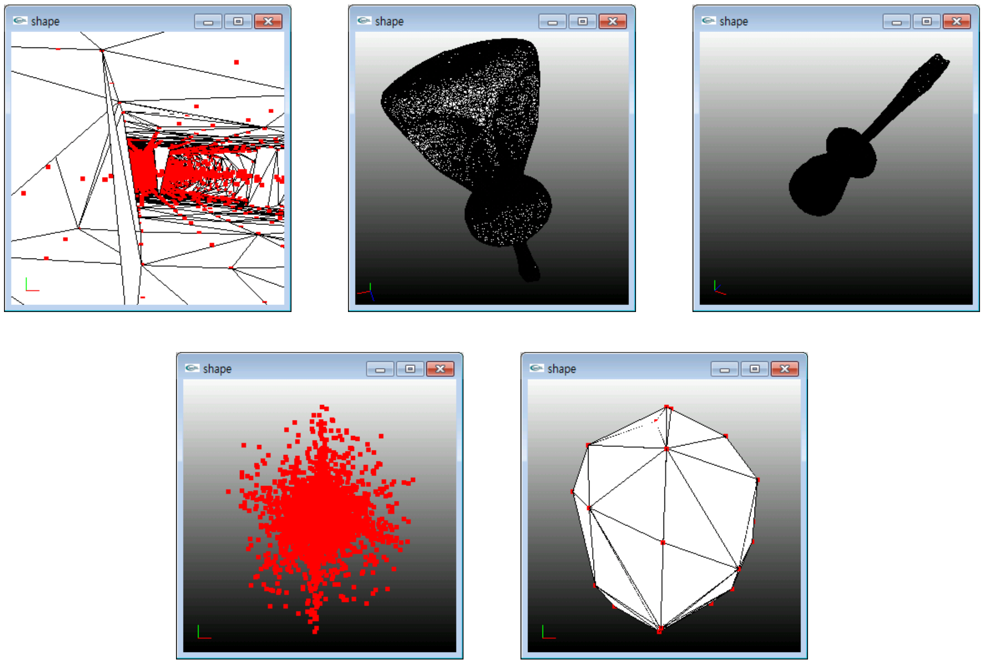


Figure 7: i.screwdriver

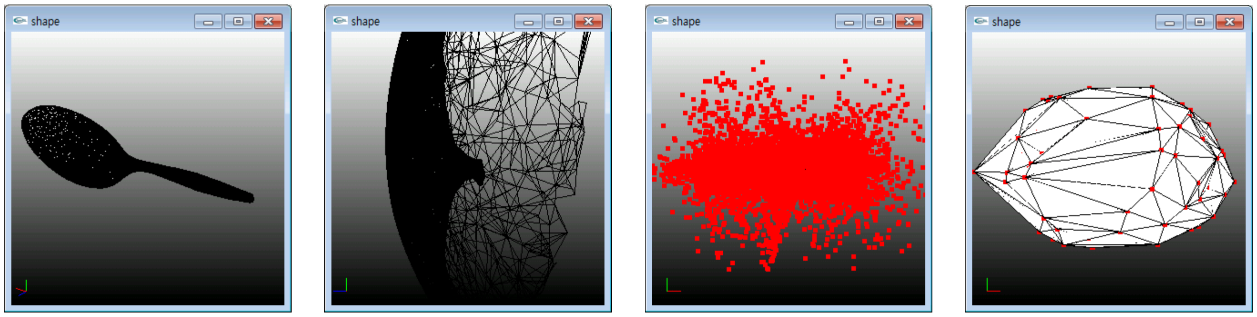


Figure 8: i.spoon

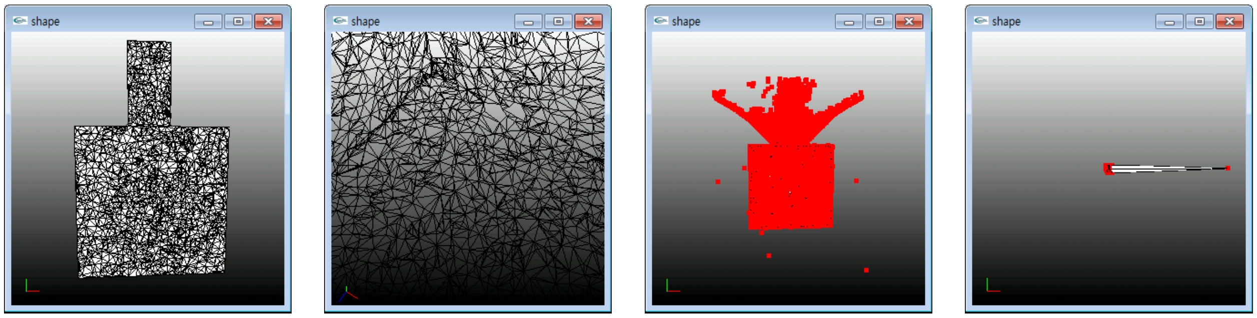


Figure 9: i.T

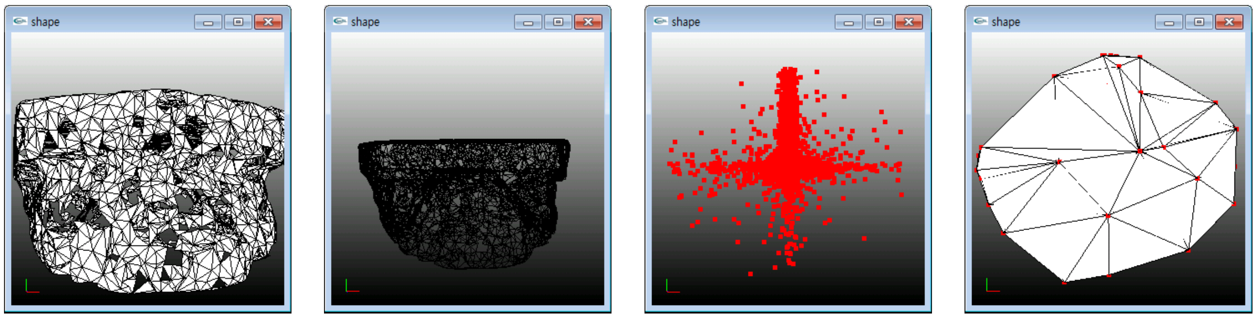


Figure 10: i.teeth

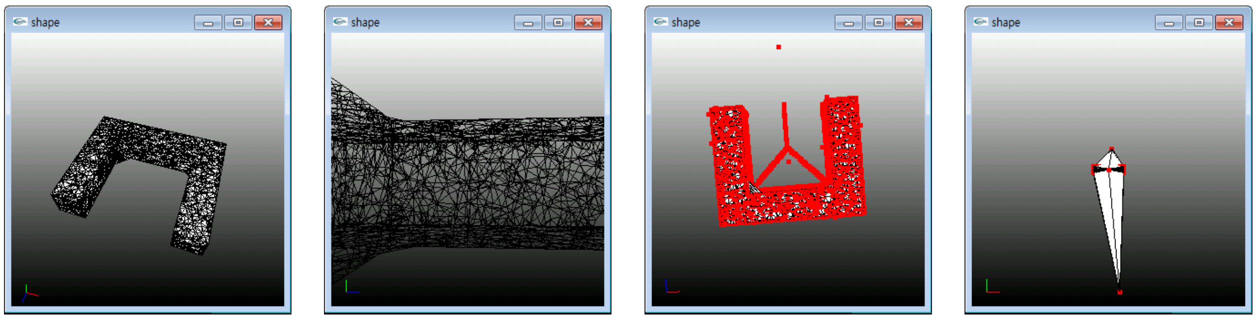


Figure 11: i.U

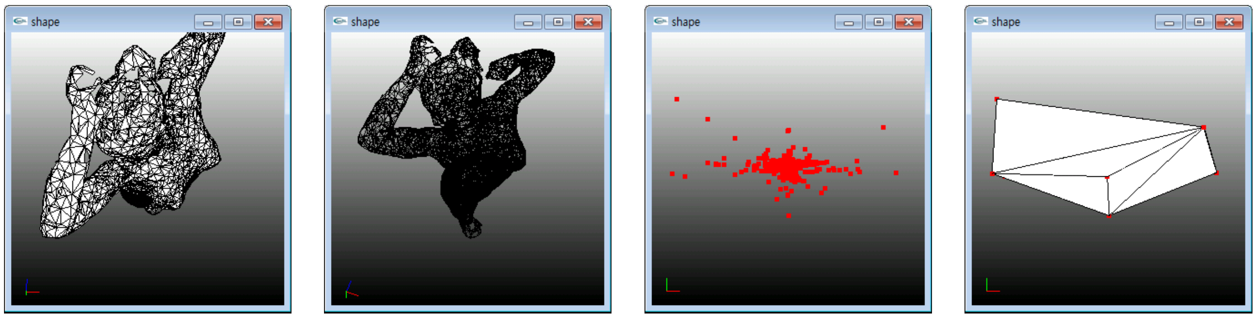


Figure 12: i.woman

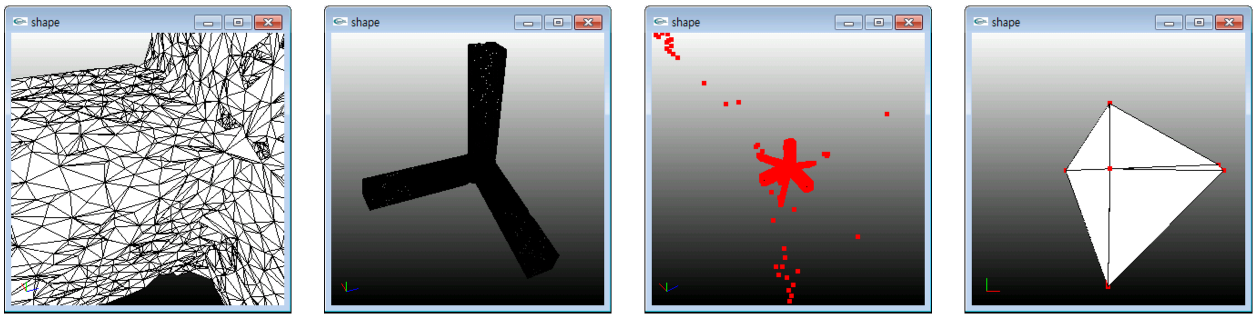


Figure 13: i.Y

3 Know bugs/limitations

To compute the voronoi vertex, I considered the neighboring facets of the vertex. In this process, I went through many trials and errors. I think that one *function* should have only one function in a nice program. Actually, I wanted to make two function; one is for finding poles and the other is for finding antipoles. However, for computing the poles for a site, it is better to compute them in one function. Also, I had to define *the maximum distance* in the process of computing them. If the distance between a site and a vertex is too large, it is difficult to view the result. For example, there is *i.ellipsoid*. So, I set the *MAX_DIST* as 10000.

References

[1] Nina Amenta. The crust algorithm for 3d surface reconstruction. 1999. 1.1

[2] MIT Geometry center. The crust algorithm for 3d surface reconstruction from scattered points. 1.1

[3] UIUC Geometry Center. Qhull. 1.4