Fall 2015

Report for Advance Algorithm Programming Assignment 1

Moran Kim

1 Implementation Details

To implement Delaunay triangulation in 3-dimension, I used 'QHull()' to compute covex hull of the vertices in 4-dimension. Given set of vertices $V = \{(v_x, v_y, v_z) \in \mathbb{R}^3 | \; for \; v_x, v_y, v_z \in \mathbb{R}\}$, we lift up the points v$\in V$ by adding additional component $V_m = (v_x)^2 + (v_y)^2 + (v_z)^2$. Let us denote $V4 = (v_x, v_y, v_z, v_m) \in \mathbb{R}^4$. Then we can construct a convex hull of V4 which is a composition of tetrahedrons. After computing the convex hull of V4, we select the vertices of the tetrahedron whose facing direction is downside. After storing the information of that selected vertices $v_{selected} \in V4$, we draw tetrahedron of $v_{selected}$ with the part of Qhull() which is Delaunay(). I assumed that the facet of the Deluanay triangulation is the face of convex hull of points that are lifted up in 4-dimension space.

2 Example Output

I list some of the pictures of the output. Given data points in $\mathbb{R}^3$ the result of the Deluanay triangulation in 3D is in Fig 1. I assumed the faset of the Deluanay triangulation in the above. The difference between facet of Deluanay triangulation and the tetrahedron of the Deluanay triangulation can be checked when we see the inside of the each shape.

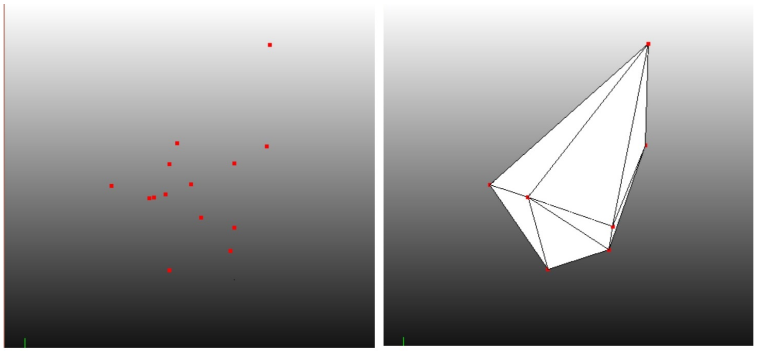

Fig 1. Data point set in 3D and Deluanay triangulation in 3D(right)

3 Known bugs/limitations

There are bunch of macros here so it was difficult to understand the code easily. For the edge part which I didn't put the result here. I'll try this. There is also a degenerate case for the cube(in "data" file). The deluanay triangulation of the cube should generate only five tetrahedrons but the output of the triangulation has more than five tetrahedrons.

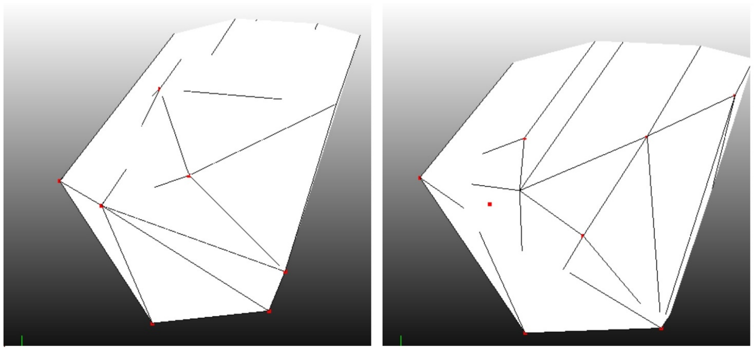

Fig 2.Inside picture of tetrahedrons of Deluanay triangulation and Inside picture of facet of Deluanay triangulation(right)

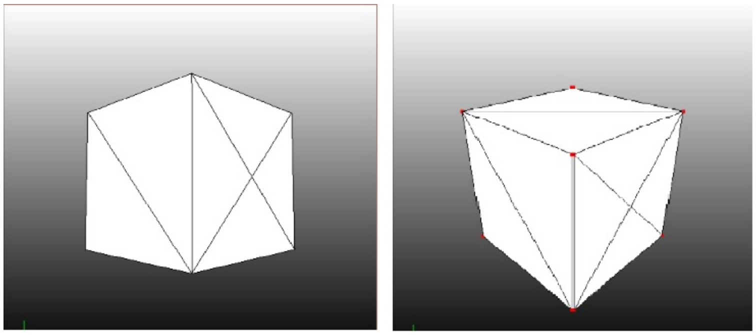

Fig 3.Degenerate case for the cube.