Advanced Algorithms Fall 2015

Program Assignment 1

Prof. Jyh-Ming Lien

The goal of this programming assignment is to deepen your understanding on 3-d Voronoi Diagram, Delaunay Triangulation, as well as α-shape and the 3-d crust algorithm. Another goal is to let you familiar with some well known libraries. Therefore, in this assignment, you are allowed to use a package called "qhull". There are 3 parts in this assignment: (1) Delaunay triangulation, (2) α-shape and (3) crust, respectively.

In addition to the implementation of the algorithms, you also need to turn in a report in LaTeX. Your report should include an overview of your implementation intuitions (e.g., how the circumsphere is computed, how the poles are found from qhull), all the example outputs, known bugs, and known limitations.

Use github to maintain your code and the report. Email me your github clone command to my email address *jmlien@cs.gmu.edu* before the deadline.

1 Part 1: 3D Delaunay Triangulation (30 pts)

This part of the assignment is due on Oct 14, 11:59pm, 2015.

1.1 What should you do?

Your goal is to use *qhull* to compute Delaunay triangulation. Your input will be 3D points. In your code, you will lift all points to 4D so that each point is $(x, y, z, x^2 + y^2 + z^2)$. Then you will compute the convex hull in 4D then output the faces of the convex hull as tetrahedron. The package qhull will be provided but, if you like, you can try to download and compile to latest version of it. The output of your program should remain the same as it is now, therefore the provided OpenGL interface can understand your output.

2 Part 2: 3D α-shape (30 pts)

This part is due on Oct 18, 11:59pm, 2015.

For a given value of α, the α-shape includes all the tetrahedra in the Delaunay triangulation which have an empty circumsphere with *squared radius* equal or smaller than α. Algorithm **??** sketches this idea. Examples of α-shape is illustrated in Figure **??**. More detailed information about 3-d

α-shape can be found in this CGAL page and in the paper by Edelsbrunner and Mücke [**?**].

Algorithm 2.1: α SHAPE$(P[1 \cdots n], \alpha)$

comment: P is a set n points

$D =$ Delaunay triangulation of P
for each tetrahedron $t \in D$
do { **if** the circumsphere of t has squared radius larger than α
do Remove t

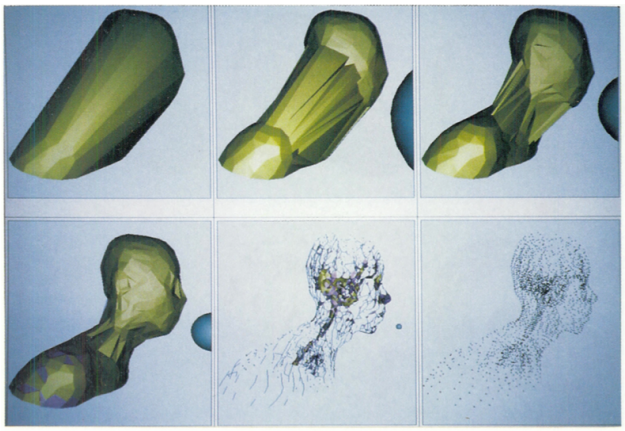

Figure 1: An example of alpha shape with different values

2.1 What should you do?

Your goal is to use *qhull* to implement Algorithm **??**. The output of your program should remain the same as it is now, therefore the provided OpenGL interface can understand your output.

3 Part 3: Crust Algorithm (40 pts)

This part is due on Oct 22, 11:59pm, 2015.

The idea of Crust for surface reconstruction is proposed by Nina Amenta et al. [**?**] in 1998. Briefly, the crust of a set of points is a set of edges (in 2d) or triangles (in 3d) whose circumcircles is empty of (1) input points and (2) the medial axis. A complete crust algorithm is shown in Fig. **??**.

The main challenge of this part of assignment is to find the poles for each site. The poles of site s are the furthest vertices of s's Voronoi cell; one on each side of the surface. An example of the poles is shown in Fig. **??**.

1. *Compute the Voronoi diagram of the sample points S*
2. *For each sample point s do:*
   (a) *If s does not lie on the convex hull, let p^+ be the farthest Voronoi vertex of V_s from s. Let n^+ be the vector $sp+$.*
   (b) *If s lies on the convex hull, let n^+ be the average of the outer normals of the adjacent triangles.*
   (c) *Let p^- be the Voronoi vertex of V_s with negative projection on n^+ that is farthest from s.*
3. *Let P be the set of all poles p^+ and p^-. Compute the Delaunay triangulation of $S \cup P$.*
4. *Keep only those triangles for which all three vertices are sample points in S.*

Figure 2: 3D crust algorithm

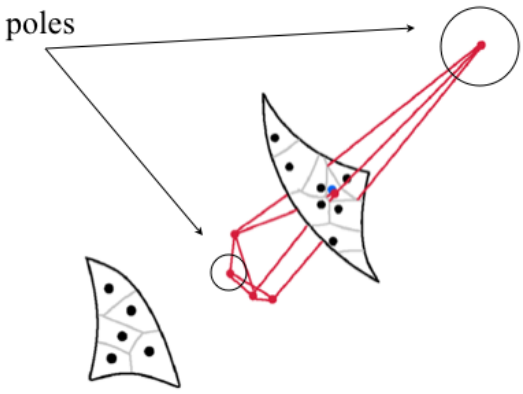

Figure 3: Poles

3.1 What should you do?

Your goal is to use *qhull* to implement the crust algorithm. The output of your program should remain the same as it is now, therefore the provided OpenGL interface can understand your output.