ME/IE/CS 558

Assignment 4

Due April 18, 2016

For all assignments: *Unless specifically indicated, you are free to use any publicly available sources: papers, books, programs, online material, etc. – as long as you clearly indicate and attribute the origin of the information.*

The goal of these assignment is to modify your program for computing Delaunay triangulation (Assignment 3) to compute the Voronoi diagram for a (triangulated) set of points (sites) P, and to use it for the problem of 2D curve reconstruction. You may find Exercise 5.7.5(2) on page 189 in the O'Rourke's book helpful.

1. (Analysis (50 points))
   (a) Describe modifications (if any) to your data structures for Delaunay triangulation and give a high-level description of the algorithm to compute (and draw) the Voronoi diagram. In order to deal with the Voronoi edges that go off to infinity, enclose your triangulation into a (much larger) box (or disk, or triangle); instead of points at infinity, you can now store the points of intersection between the Voronoi edge and the enclosing box.
   (b) Recall that the Voronoi and Delaunay edges are dual to each other. This means that for every edge in your data structure, you actually have a choice of which edge to draw. Let a, b be the end points of the Delaunay edge, and p, q are the end points of the corresponding (dual) Voronoi edge. Consider a drawing procedure *Shape* that operates as follows: if there exists a circle through points a and b that does not include the points p and q, then draw the Delaunay edge; else draw the dual Voronoi edge. Design an efficient procedure to perform this test.
   (c) Recall the idea of the "crust algorithm" for shape reconstruction. First construct a Delaunay triangulation of input points; then add the circumcircles and recompute DT. If the Delaunay edges appear in both triangulations, they are deemed to be good candidates for boundary of the reconstructed shape. Compare the result of this procedure and the output you expect to obtain from the *Shape* procedure outlined above. Are the Delaunay edges produces by the *Shape* different or the same, and why?
   (d) Now consider the Voronoi edges produced by the *Shape* procedure. Explain whether or not they are allowed to intersect the Delaunay edges produced by *Shape* and why.
2. Implementation & Testing (50 points)
   (a) Implement the program to produce a graphic output showing the computed Voronoi diagram.
   (b) Implement the program for the *Shape* procedure from the analysis portion of the homework.
   (c) Test both programs on the same data you used to test your program to compute the Delaunay triangulation.

(d) Test the *Shape* procedure on the points generated by on some closed curve. You can create the curve by inputing the ordered list of points, or generate points using parametric description of the curve. How well do the Delaunay edges reconstruct the curve? Speculate on the significance of the Voronoi edges produced by your procedure. A good source of planar parameteric curves is http://mathworld.wolfram.com/topics/PlaneCurves.html

Deliverables

Please use the course website to submit a single zip named FirstName_LastName_HW4.zip The zip archive should contain: (1) the analysis portion of the assignment, (2) the documented python source file, and (3) a PDF readme file specfying the instructions for running the code. It should also include at least 1 sample run with input and output, and specify any specific dependencies or requirements of your code.