CS 558 Assignment 4

Jieru Hu

Analysis

**Overview**

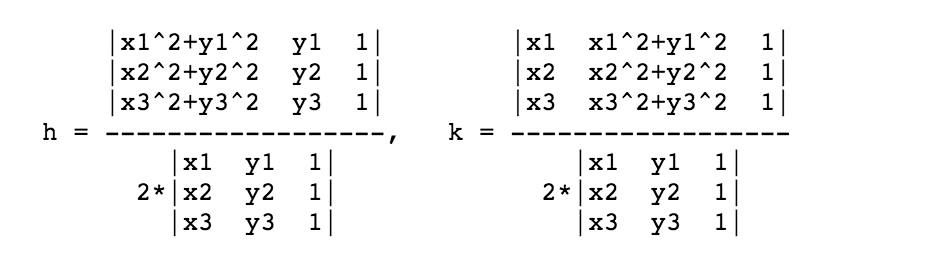
In this assignment, we modified data structure from part 3 and use it to convert the Delaunay triangulation to the Voronoi diagram. We used the instruction specified and constructed a new shape for the given points. At last, we reconstruct the shape for the points based on the crust algorithm.

**(a)Voronoi Diagram**

First, we need to compute an enclosing box with proper size, which contains all the input points. To compute the minimum box to enclose all the points, we have to find the Xmin, Xmax, Ymin, and Ymax. We have two points to locate a rectangle. Then we enlarge four extreme points by a factor of 0.2 of the vertical and horizontal dimension. So overall the length and width of the enclosing box is increased by a factor of 1.4.

To accommodate for the voronoi diagram, the only modification I made to the data structure is add two more instances in the edge object. One is the named “center”, which represents the center of the circumcircle of the triangle corresponded to the edge. Another instance is name “tcenter”, which represents the center of the circumcircle of the triangle corresponded to the twin edge. There is a special case when the edge’s twin edge does not have a triangle, which means that the edge is on the convex hull of the triangulation. In this case, the voronoi edge extends to infinity. However, since all the points are enclosed in a box, we could simply find a point on the enclosing box and connect it with the “center” to form an edge for animation. To find the point on the boundary, we first calculate the line function for the bisector of the edge. It is easy to find the midpoint of the edge by taking average of the two endpoints. Then combined with the “center” point, we could derive the expression for the bisector y = ax + b, by solving a and b. Once we get the expression for the bisector, we extend the bisector until it hits one of the boundary. Then we set the “tcenter” to the boundary point.

I used the method of calculating the determinant to locate the center of the circumcircle given three points, as following:



Here the h represents the x-coordinate of the center of the circumcircle and and k represents the y-coordinate of the center.

After all the edges has its instances filled, I used the turtle to connect the center and the tcenter of each edge, so the voronoi diagram is constructed.

For each edge, computing the the center of the circumcircle and twin circumcircle requires constant time. If there are n triangles in the Delaunay triangulation, it takes less than 3n time to construct the voronoi diagram. So the overall time is linear.

**(b)Shape Reconstruction**

For the shape reconstruction, I followed the instruction in the assignment. For each edge, a, b are the endpoints for the Delaunay edge and p, q are the endpoints of its corresponding dual edge. The query is stated as following: if the circumcircle formed a, b, and p contains q, we draw the Delaunay edge; if the circumcircle does not contain q, we draw the dual edge.

Since for each edge, the query used to decide whether we need to include the edge is a constant time operation, the total time for the shape reconstruction is also linear.

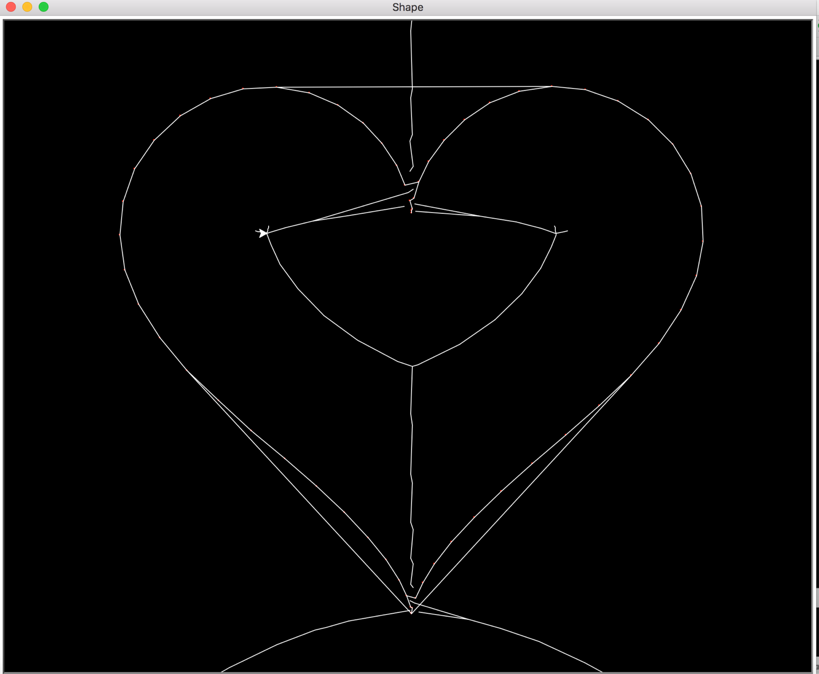
**(c)Crust algorithm**

To implement the crust algorithm on the voronoi diagram, we compute the voronoi diagram first and save the vertices of the voronoi diagram, which are the center of the circumcircle of each triangle. Then we combine the input points and the voronoi points to do another triangulation. Then we do another query on each edge in the Delaunay triangulation: if two endpoints of the edge are both in the set of input points, we keep the edge. Otherwise, we do not include the edge. In the end, we draw all the edges kept at the query.

As we analyzed before, the runtime for the Delaunay triangulation is O(n^2). And the voronoi diagram takes O(n) to construct. In the implementation, I used set to store all the input points. Since, the look up time for each element in the set is O (1), each query takes O(1) to decide. Therefore, the last step takes n\*O(n), which is equal to O(n).

The output from the shape procedure is different from the output from the crust algorithm. The shape reconstruction connects the all the input points and the voronoi points as a shape. However, the crust algorithm only connects all the input points and produces the shape. The shape reconstruction also includes some voronoi edges extend to infinity in some region, as shown in the readme file. The Delaunay edges produced by the shape is the same as the crust. Because after the triangulation of the input points and voronoi points, the only edges with endpoints belong to the original input points are points on the boundary, which forms the shape. The shape procedure will always keep the the boundary edge because the boundary Delaunay edge has its twin circumcircle center at infinity, so it will always keep the Delaunay edge in the query.

**(d)** The voronoi edges produced by the shape procedure are allowed to intersect the Delaunay edge.



In this example, the horizontal edge connecting the two top side of the heart is originally a Delaunay edge. The vertical edge crossing this horizontal edge is formed by segments of voronoi edges. Generally, if the input points form some concave shape, voronoi edges are left after the query test. The shape test connects all the voronoi points in the concave region and points to the infinity. However, the Delaunay edge on the convex hull enclosing this concave region is always kept according to the query test. Therefore, such a Delaunay edge will always connect the line segments formed by voronoi edges.