Geometric Algorithms
Assignment 4

Laura Baakman (s1869140)

October 15, 2014

A

B

The display method `display_circumscribed_circles`, see Listing 1, draws the Delaunay Triangulation and the circumscribed circles of three randomly chosen circles. To run the script `assignment4A` with this display method use the flag `circ`, the result of one such call is shown in Figure 1.

Listing 1: The relevant part of the method `display_circumscribed_circles()`.

```
def display_circumscribed_circles():
    """Display the circumscribed circle of three triangles."""
    # Draw Delaunay Triangulation
    # Draw points
    # Draw circles
    global dcel, cens
    triangle_idxs = sample(
        xrange(len([face for face in dcel.faces if face.outer_component])), 3)
    for triangle_idx in triangle_idxs:
        triangle = dcel.faces[triangle_idx]
        (_, vertices) = triangle.outer_component.get_incident_face()
        vertex = vertices[0].as_points()
        center = cens[triangle_idx]
        radius = sqrt((vertex[0] - center[0]) ** 2 + (vertex[1] - center[1]) ** 2)
        draw_circle(center, radius)
    glutSwapBuffers()
```

Circles are drawn with the provided method `draw_circle` which expects the centre and the radius of circle. The centre of the circle through all three points of a triangle is the circumcentre of the triangle, which is one of the results of `matplotlib.delaunay.delaunay()`. Since we have added the faces of the triangles in the same order as they were provided by the triangulation method the index of a face in `faces` attribute of the `DCEL` is the same as the index of its circumcentre in the `cens`.

To determine the radius of a circle we compute the distance between one of the points on the circle, the vertices of the triangle, and its centre. We use the method `get_incident_face` on the `outer_component` of the face to get the vertices, see Listing 2

Listing 2: The method `get_incident_face()` in the class `HalfEdge`.

```
    def get_incident_face(self):
        """Return the edges and vertices of the incident face."""
        def get_incident_face_helper(current_edge, edges, vertices):
            if(self == current_edge):
                return (edges, vertices)
            else:
                edges.append(current_edge)
                vertices.append(current_edge.origin)
                return get_incident_face_helper(current_edge.nxt, edges, vertices)
        return get_incident_face_helper(self.nxt, [self], [self.origin])
```

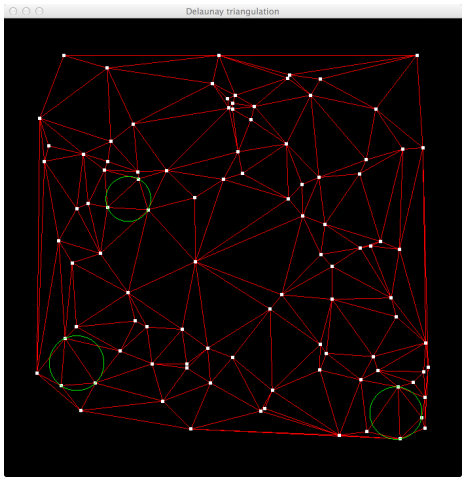

Figure 1: The Delaunay triangulation of the white points is shown in red, the circumscribed circles of three randomly selected triangles is shown in green.