
Code
regel-
nummers
checken!

Geometric Algorithms
Assignment 4

Laura Baakman (s1869140)

October 16, 2014

A

B

C

Since the Voronoi diagram is the dual of the Delaunay Triangulation, taking the dual of the second gives the first. The duals of the different parts that make up a doubly connected edge list are given in Table 1.

The code used to determine the duals of the elements of a DCEL is presented in Listing 1. To set the twins I have used the local method `set_twins()` which uses the fact that edges e and their twins e' are placed in the list in the following order: e_1, e'_1, e_2, e'_2, ...

Listing 1: The method `dual` in the class `DCEL`.

```
    def dual(self):
        """Return the dual of the current DCEL."""
        def set_twins():
            for edge_idx in range(0, len(dual_dcel.edges), 2):
                dual_dcel.edges[edge_idx].twin = dual_dcel.edges[edge_idx + 1]
                dual_dcel.edges[edge_idx + 1].twin = dual_dcel.edges[edge_idx]

        def set_next_and_previous():
            for face in dual_dcel.faces:
                face_edges = [edge for edge in dual_dcel.edges if edge.incident_face == face]
                for edge in face_edges:
                    if(not edge.get_destination().is_infinity()):
                        edge.nxt = [e for e in face_edges if e.origin == edge.get_destination()][0]
                    if(not edge.origin.is_infinity()):
                        edge.prev = [e for e in face_edges if edge.origin == e.get_destination()][0]
```

Voronoi diagram	Delaunay Triangulation
Face	Vertex
circumcentre	origin
outer component	incident edge
Vertex	Face
origin	circumcentre
incident edge	outer component
Edge	Edge
origin	incident face
incident face	destination
twin	twin

Table 1: The dual of the elements of the DCEL.

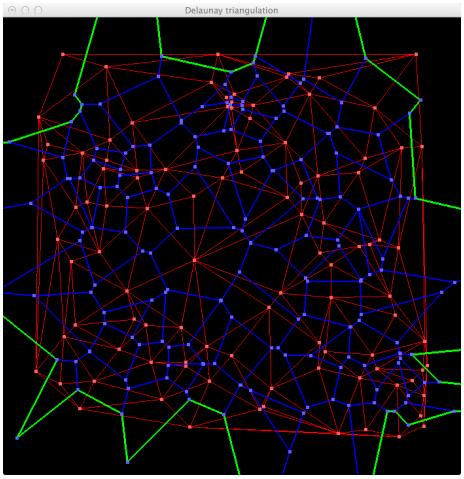

Figure 1: A Voronoi diagram, shown in blue and green, based on the Delaunay triangulation shown in red.

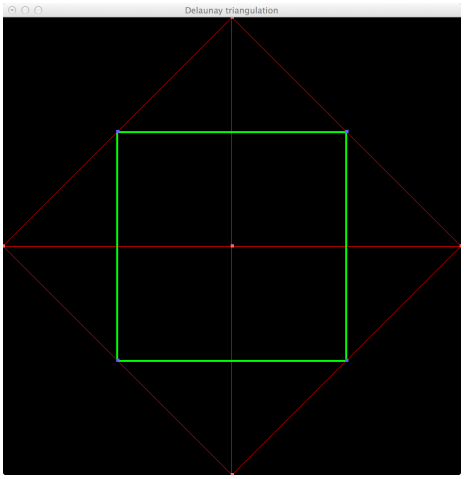

Figure 2: A Voronoi diagram, shown in blue, with edges highlighted in green, based on the Delaunay triangulation shown in red.

```
        dual_dcel = DCEL()
        for edge in self.edges:
            incident_face = dual_dcel.add_face(Face(circumcentre=edge.twin.origin.as_points()))
            origin = dual_dcel.add_vertex(Vertex(coordinates=edge.incident_face.circumcentre))
            dual_edge = HalfEdge(
                origin=origin,
                incident_face=incident_face
            )
            incident_face.outer_component = dual_edge
            origin.incident_edge = dual_edge
            dual_dcel.edges.append(dual_edge)

        set_twins()
        set_next_and_previous()
        return dual_dcel
```

Since I could not figure out what the dual of the next and previous edges where I have used written the method `set_next_and_previous()`. For each `Face` in the `DCEl` this method find the edges adjoining that face and sorts out the next and previous using the coordinates of their vertices.

A Voronoi diagram created by taking the dual of the Delaunay Triangulation is presented in Figure 1. The Boundaries of a simpler Voronoi diagram are highlighted in Figure 2 to show that every half-edge of an outer non-finite face is shown.

The outer boundaries are found with the following algorithm:

1. Find a infinite `HalfEdge edge` with its `origin` on the outer edge of the diagram.
2. Store the origin of this `HalfEdge` as `first_vertex`.
3. **`while`** the destination of `edge` is not the `first_vertex`:
   (a) If `edge` is infinite get the next of its twin.
   (b) Else store edge `edge` as part of the outer boundary and continue with the next of `edge`.
4. Store `edge` as part of the outer boundary.

Which is implemented in Listing 2. The code used to present the results is shown in Listing 3. To run the script with this display code use the option `-voronoi`.

Listing 2: The method `get_outer_boundary_of_voronoi` in the class `DCEL`.

```
    def get_outer_boundary_of_voronoi(self):
```

```
        """Return a list of vertices that form the outer boundary of finite faces of the DCEL."""
        edge = [edge for edge in self.edges if not edge.nxt][0]
        # next(obj for obj in objs if obj.val==5)
        first_vertex = edge.origin
        outer_boundary = []
        while (not edge.get_destination() == first_vertex):
            if(edge.get_destination().is_infinity()):
                edge = edge.twin.nxt
            else:
                outer_boundary.append(edge)
                edge = edge.nxt
        outer_boundary.append(edge)
        return outer_boundary
```

Listing 3: The method `display_delaunay_and_voronoi` in the script `assignmetn4A`.

```
def display_delaunay_and_voronoi():
    """Display the Delaunay triangulation and the Vornonoi diagram."""
    global dcel
    glClear(GL_COLOR_BUFFER_BIT)
    # Draw Delaunay Triangulation
    glLineWidth(1.0)
    glColor3f(1.0, 0.0, 0.0)
    glBegin(GL_LINES)
    for i in range(len(edgs)):
        glVertex2f(xl[edgs[i][0]],  yl[edgs[i][0]])
        glVertex2f(xl[edgs[i][1]],  yl[edgs[i][1]])
    glEnd()
    voronoi = dcel.dual()
    # Draw Voronoi
    glLineWidth(2.0)
    glColor3f(0.0, 0.0, 1.0)
    glBegin(GL_LINES)
    for edge in [edge.as_points() for edge in voronoi.edges if edge.is_finite()]:
        glVertex2f(edge[0][0], edge[0][1])
        glVertex2f(edge[1][0], edge[1][1])
    glEnd()
    # Draw outer boundary of the voronoi diagram
    glLineWidth(3.0)
    glColor3f(0.0, 1.0, 0.0)
    glBegin(GL_LINES)
    for edge in voronoi.get_outer_boundary_of_voronoi():
        edge_points = edge.as_points()
        glVertex2f(edge_points[0][0], edge_points[0][1])
        glVertex2f(edge_points[1][0], edge_points[1][1])
    glEnd()
    # Draw Voronoi Vertices
    glColor3f(0.4, 0.4, 1.0)
    glPointSize(5)
    glBegin(GL_POINTS)
    for bounded_face in dcel.get_bounded_faces():
        circumcentre = bounded_face.circumcentre
        glVertex2f(circumcentre[0], circumcentre[1])
    glEnd()
    # Draw Deaunay Vertices
    glColor3f(1.0, 0.4, 0.4)
    glPointSize(5)
    glBegin(GL_POINTS)
    for i in range(len(xl)):
        glVertex2f(xl[i], yl[i])
    glEnd()
    glutSwapBuffers()
```