BILKENT UNIVERSITY
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER ENGINEERING

CS 478

Project Description
Delaunay Triangulation In 2D

Gürkan Gür
21602813

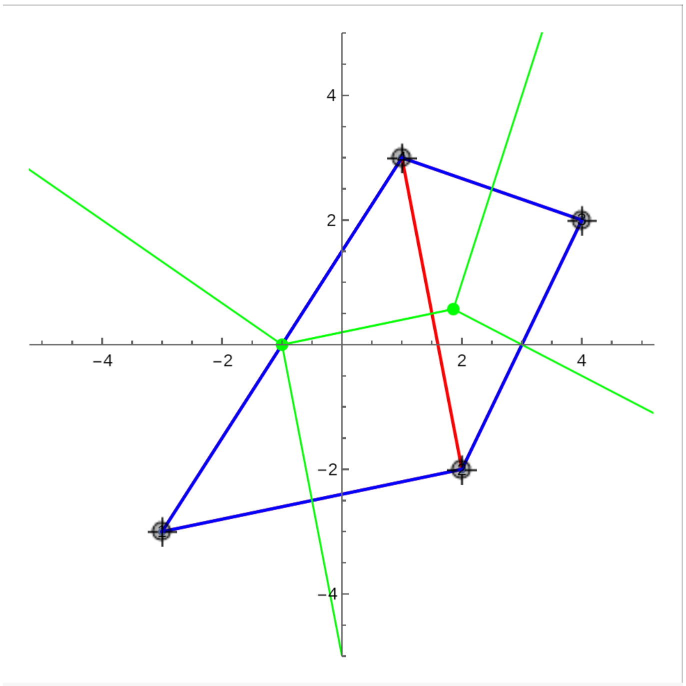
Table of Contents

Survey

A triangulation is a subdivision of an area (volume) into triangles (tetrahedrons). The Delaunay triangulation has the property that the circumcircle (circumsphere) of every triangle (tetrahedron) does not contain any points of the triangulation.[1] Every circumcircle of a triangle is an empty circle (Okabe et al. 1992, p. 94).

Delaunay triangulations, convex hulls and Voronoi diagrams have a connection between them. In the context of 2 dimensions, “The lifting map λ : R^2 → R^3 is defined by λ(x1, x2) = (x1, x2, x2 1 + x 2 2); Λ = λ(R 2) is a paraboloid of revolution about the vertical axis.”[2] If the lower faces(a face is lower if it has a supporting plane with inward normal having positive vertical coordinate) of a convex hull of the lifted sites orthogonally projected into R^2, Delaunay triangulation of the same point set is obtained.[2]

Connection between a delaunay triangulation and a Voronoi diagram is described in [2] as “...suppose that triangle λ(s)λ(t)λ(u) is a lower facet of H, and that plane P passes through λ(s)λ(t)λ(u). The intersection of P with Λ is an ellipse that projects orthogonally to a circle in R^2. Since all other lifted sites are above the plane, all other unlifted sites are outside the circle, and stu is a Delaunay triangle. The opposite direction, that a Delaunay triangle is a lower facet, is similar. For Voronoi diagrams, assign to each site s = (s1, s2) the plane Ps = {(x1, x2, x3) : x3 = −2x1s1 + s 2 1 − 2x2s2 + s 2 2 }. Let I be the intersection of the lower halfspaces of the plane's Ps. The Voronoi diagram is exactly the orthogonal projection into R^2 of the upper faces of I.”

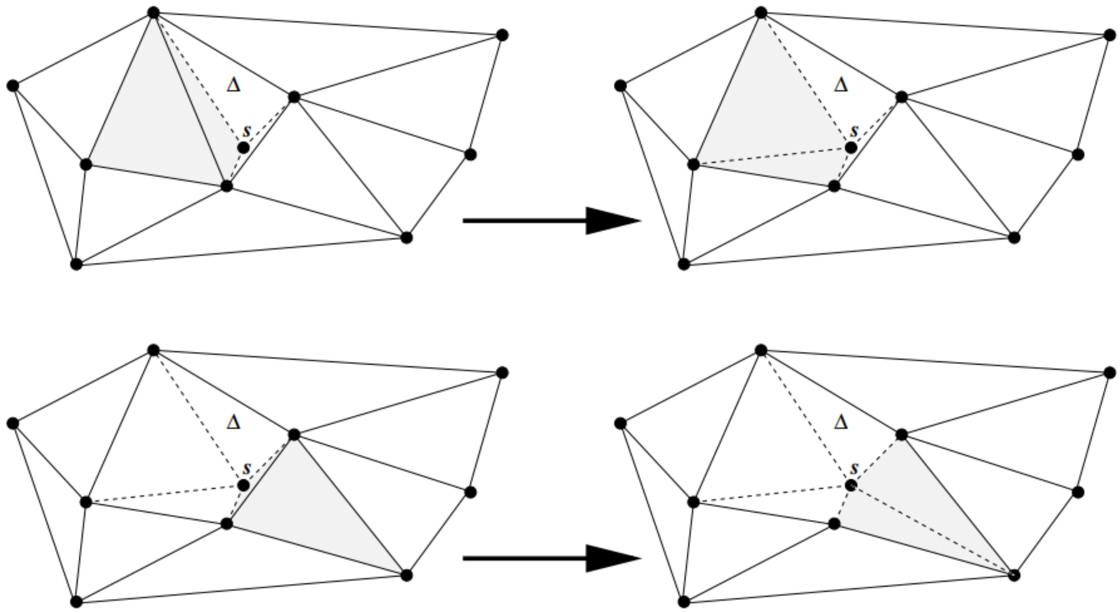


An illustration of Delaunay triangulation(blue and red), convex hull(blue) and Voronoi diagram(green).[3]

Algorithms and Data Structures

Convex hull algorithms(the divide-and-conquer, incremental, and gift-wrapping algorithms) can be used to compute Delaunay triangulations.
Randomized Incremental Algorithm and Plane sweep algorithms will be used in this project.

Randomized Incremental Algorithm
A randomized incremental algorithm adds vertices one by one. After each addition, triangulation is checked and updated. The update consists of discovering all Delaunay faces.[2]



An illustration of the update phase. s is the new added vertex.[5]

Initial Step
On an empty plane, think of 3 artificial points outside of all possible points. Add a random point from the point set.

Method
Find the triangle that contains the newly added point. Connect the vertices of the triangle with the newly added point.
Perform flips to create Delaunay triangulation. At first glance, it may seem like we need to search all the triangles and perform flips on them. However, if we sign the edges that need to be changed as “bad”, and that can stay the same as “good”;
“(a) In every flip, the convex quadrilateral Q in which the flip happens has exactly two edges incident to ps(new point), and the flip generates a new edge incident to ps.
(b) Only the two edges of Q that are not incident to ps can become bad after the flip.”
Considering these 2 rules, a queue of potentially bad edges can be maintained.
A good edge will be removed from the queue, and a bad edge will be flipped and replaced according to (b) with two new edges in the queue.

Implementation on Webgl

Initialize vertex and fragment shaders.
Initialize vertex and color buffers with a defined maximum number of vertices.
Initialize data sources.

```
function FRONTENDINPUTS
        //Take number of points, zoom in/out, rotate and translate inputs interactively

function Translate
        //Standard translate function for 2d
        render()

function Rotate
        //Standard rotate function for 2d
        render()

function ZOOM
        //Perform zoom in/out through a setted camera
        render()


function LEGALTRIANGULATION(T) //Taken from lesson slides
Input: Some triangulation T of a point set P.
while T contains an illegal edge pi pj
do begin (* edge flip *)
        Let pi pj pk and pi pj pl be the two adjacent triangles to pi pj
        Remove pi pj and add pk pl instead.
        Set buffer
        Render()
end
```

return T
end

Algorithm DELAUNAYTRIANGULATION(P) //Taken from lesson slides
Input: A set P of n points in the plane.
Output: A Delaunay triangulation of P.
1. Let p-1 , p-2 , and p-3 be a suitable set of three points such that P is contained in the triangle p-1 p-2 p-3.
2. Initialize T as the triangulation consisting of the single triangle p-1 p-2 p-3 .
3. Compute a random permutation p1, p2 , …., pn of P.
4. for r ← 1 to n
5 do (* insert pr into T *)
6 Find a triangle pi pj pk Є T containing pr .
7 if pr lies in the interior of the triangle pi pj pk
8 then Add edges from pr to the vertices of pi pj pk , thereby splitting pi pj pk into three triangles.
9 LEGALIZEEDGE(pr , pi pj , T)
10 LEGALIZEEDGE(pr , pj pk , T)
11 LEGALIZEEDGE(pr , pk pi , T)
12 else (* pr lies on an edge of pi pj pk , say pi pj *)
13 Add edges from pr to pk and to the third vertex pl of the other triangle that is incident to pi pj , thereby splitting the two triangles incident to pi pj to four triangles.
14 LEGALIZEEDGE(pr , pi pl , T)
15 LEGALIZEEDGE(pr , pl pj , T)
16 LEGALIZEEDGE(pr , pj pk , T)
17 LEGALIZEEDGE(pr , pk pi , T)
18 Discard p-1 , p-2 , and p-3 with all their incident edges from T.
19 return T

Algorithm LEGALIZEEDGE(pr , pi pj , T)
1. (* The point being inserted is pr , and pi pj is the edge of T that may need to be flipped *)
2. if pi pj is illegal
3. then Let pi pj pk be the triangle adjacent to pr pi pj along pi pj
4. (* Flip pi pj *) Replace pi pj with pr pk .
5. LEGALIZEEDGE(pr , pi pk , T)
6. LEGALIZEEDGE(pr , pk pj , T)

```
function Render
        for(var i=0; i<numOfTriangles; i++) { //Edges not part of the convex hull will be drawn
twice
      gl.drawArrays( gl.TRIANGLE_STRIP, 0, 3 );
  }
```

Plane Sweep Algorithm
Sort the points on an array according to ascending y coordinates. Let AS be the resulting array.

Variables
Q //A fifo queue
S
Count //Point count

Initialize first 2 elements of the array
Initialize S = AS[0];
Q.push(S);

S = AS[1]
Q.push(S);

Main Loop
```
int i = 2 //A loop counter
int qp //Queue index
While( S != AS[count-1]):
        S = AS[i]
        Q.push(S)
        Triangulate(&Q,S);
        i=i+1
```

References

[1] *Delaunay triangulation in two and three dimensions*. Delaunay triangulations. (n.d.). Retrieved March 26, 2022, from http://www.ae.metu.edu.tr/tuncer/ae546/prj/delaunay/?fbclid=IwAR0PC6VdEKvX6eDyHVbyczRgNs4Z0cl9qvMm3XFUEovzc6atiXW5-xZ2QwU

[2] *27 Voronoidiagrams Anddelaunaytriangulations*. (n.d.). Retrieved March 26, 2022, from https://www.csun.edu/~ctoth/Handbook/chap27.pdf

[3] *Wolfram Demonstrations Project*. Convex Hull and Delaunay Triangulation. (n.d.). Retrieved March 26, 2022, from https://demonstrations.wolfram.com/ConvexHullAndDelaunayTriangulation/

[4] *Delaunay triangulations: Properties, algorithms, and ...* (n.d.). Retrieved March 26, 2022, from https://members.loria.fr/MTeillaud/collab/Astonishing/2017_workshop_slides/Olivier_Devillers.pdf

[5] Theory of combinatorial algorithms, Institute of Theoretical Computer Science, Department of Computer Science, ETH Zürich. (n.d.). Retrieved March 26, 2022, from https://www.ti.inf.ethz.ch/ew/Lehre/CG13/