Aspect Ratio of a Scatter Plot in its Delaunay Triangulation

NGUYEN Minh Thang TRAN Thanh Phu

April 30, 2015

Abstract

This article proposes an algorithm to find a scale factor $s \in R+$ from set of point $Q = (q_1, q_2, ..., q_n)$. We assume that Q will be compute to Delaunay Triangulation before it was scaled to a scale factor $P(s)$ without edges of Delaunay Triangulation. The topology changes of scatter plot due to scale factor are update on x-coordinates by s, each point of scatter plot $p_i \in P(s)$ with point coordinate(sx_i, y_i).

1 Introduction

Scatter plot is type of mathematical diagram using Cartesian coordinates to display values for a set of data. The data is displayed as a collection of points. This kind of plot is also called scatter chart, scattergram, scatter diagram. A scatter plot is used when a variable exists that is under control of the experimenter. A scatter plot can suggest various kinds of correlations between variables with a certain confidence interval. For example, weight and height, weight would be on x axis and height would be on the y axis.

2 Definitaions and objectives

In this section, we briefly introduce several definitions and properties related to implementation the algorithm. Scatter plot is a diagram that visualizes two-dimensional data as set of points in the plane. Delaunay Triangulation for a set P of points in a plane is a triangulation such that no point P inside the circumscribe of any triangle $DT(P)$. Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation. An aspect ratio is good is Delaunay triangulation of scatter plot at this aspect ratio has some nice geometric property. We present an exact algorithm for maximizing the minimum angle of the triangulation.

3 Description of the work

We read some main part of the article ?Selecting the Aspect Ratio of a Scatter Plot Based on Its Delaunay Triangulation? such as: Delaunay Triangulation, aspect ratio, the edge flipped, circumscribe triangle.

- With the Delaunay Triangulation, we find the algorithm to compute from a set of point in the plane to a Delaunay Triangulation graph.

- With aspect ratio, we find the function to scale the graph with scale factor
- With the edge flipped, we find a smallest scale factor in order to the corresponding quadrilateral becomes co-circular.

4 Methods and used tools

We use Tulip framework to implement by C++, we can visualize it on interface of Tulip. The input is the set of points $Q = q_1, q_2, .., q_n$on plan by x-coordinate and y-coordinate. The set of point will be compute to Delaunay Triangulation graph. We don?t mention about how to compute to Delaunay Triangulation in here, because we have reused an algorithm in Tulip framework. The problem is finding an event point for edge flipped. It?s hard to find exactly event point in quadrilateral, so we declare a sequence event point $s_1, s_2, ..., s_n$ with interval is ϵ(the value default is 0.05), we think it's better to check edge flipped. In our algorithm, we check each event point until the edge Delaunay was flipped. The second thing is selecting a good event point, in our implement, we get the last event point from priority queue in order to scale set of points.

It's complicated to find four points was scale that is concyclic. In below, We show the algorithm FindEventPoint() with an approximate result s by declare an sequence of event points increase one by one with a constant interval value in $[s_i, s_{i+1}]$.

input : G is the set of points without egdes
output: s is an aspect ratio

$s = 1$;
$G = (V, E) = D(P)$;
$Q = newPriorityQueue()$;
$mV = FindMapPointNeighborhood(G)$;
foreach $e \in E$ **do**
    $t = FindEventPoint(e, G)$;
    if $t \geq s$ **then**
        $Q.Insert(t, e)$;
    end
end
while $Q \neq \emptyset$ **do**
    $ptr = Q.ExtractMin()$;
    $s = ptr.eventpoint$;
    $e = ptr.edge$;
    $f = FlipAndUpdateGraph(e, G)$;
    $E' = FindSetOfEdgesOfQuadrilateral(f, G)$;
    foreach $g \in E'$ **do**
        $t = FindEventPoint(g, G)$;
        Update t and g in Q;
    end
end

Algorithm 1: Find Aspect Ratio

input : e is edge and G is set of points
output: an event point

$s = 1$;
$Graph * subGraph = FindSubGraph(e)$;
`/* sequence event point with interval is` eps `*/`
$eps = 0.05$;
find length of e and a, b, c, d;
$r1$ is radius cricle of (e, a, b);
$r2$ is radius cricle of (e, c, d);
Increase s with eps then e be flip;
if $s == 1$ **then**
  return -1;
end
return s;

Algorithm 2: Find Event Point

5 Results

In our implement, we also test for some the figure or some set of points in plane. With each point that have small value in each coordinate, the plugin can run it so fast, addition it can run with large points. But, with the larger value in each coordinate, the plugin run slowly and less than number of points. The selecting good aspect ratio depend on the way we need, our implement isn't optimize for select right aspect ratio, and of course, the last event point is right aspect ratio.

References

[Selecting the Aspect Ratio] IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS 2013 *Martin Fink, Jan-Henrik Haunert, Joachim Spoerhase, and Alexander Wolff*

[Circumscribed circle] Wikipedia

[Cyclic quadrilateral] Wikipedia

[Voronoi diagrams over dynamic scenes] Discrete Applied Mathematics *Thomas Roos*

[Distributed Kinetic Delaunay Triangulation] Division of Computer Science, Department of Electrical Engineering and Computer Science Korea Advanced Institute of Science and Technology *Taewon Yoo, Sunghee Choi, Hyonik Lee, Jinwon Lee*