Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computational Geometry 31 (2005) 166–178
Computational Geometry
Theory and Applications
www.elsevier.com/locate/comgeo

Fast reconstruction of Delaunay triangulations ☆

Christian Sohler

Heinz Nixdorf Institute and Department of Mathematics & Computer Science, University of Paderborn, D-33095 Paderborn, Germany

Available online 25 January 2005

Communicated by J. Snoeyink

Abstract

We present a new linear time algorithm to compute a good order for the point set of a Delaunay triangulation in the plane. Such a good order makes reconstruction in linear time with a simple algorithm possible. Similarly to the algorithm of Snoeyink and van Kreveld [Proceedings of 5th European Symposium on Algorithms (ESA), 1997, pp. 459–471], our algorithm constructs such orders in $O(\log n)$ phases by repeatedly removing a constant fraction of vertices from the current triangulation. Compared to [Proceedings of 5th European Symposium on Algorithms (ESA), 1997, pp. 459–471] we improve the guarantee on the number of removed vertices in each such phase. If we restrict the degree of the points (at the time they are removed) to 6, our algorithm removes at least $1/3$ of the points while the algorithm from [Proceedings of 5th European Symposium on Algorithms (ESA), 1997, pp. 459–471] gives a guarantee of $1/10$. We achieve this improvement by removing the points sequentially using a breadth first search (BFS) based procedure that—in contrast to [Proceedings of 5th European Symposium on Algorithms (ESA), 1997, pp. 459–471]—does not (necessarily) remove an independent set.

Besides speeding up the algorithm, removing more points in a single phase has the advantage that two consecutive points in the computed order are usually closer to each other. For this reason, we believe that our approach is better suited for vertex coordinate compression.

We implemented prototypes of both algorithms and compared their running time on point sets uniformly distributed in the unit cube. Our algorithm is slightly faster. To compare the vertex coordinate compression capabilities of both algorithms we round the resulting sequences of vertex coordinates to 16-bit integers and compress them with a simple variable length code. Our algorithm achieves about 14% better vertex data compression than the algorithm from [Proceedings of 5th European Symposium on Algorithms (ESA), 1997, pp. 459–471].

☆ Research supported by DFG grant Me872/7-1, and by the IST program of the EU under contract number IST-1999-14186 (ALCOM-FT). A preliminary version of this paper appeared in Proceedings of the 11th Canadian Conference on Computational Geometry (CCCG'99), 1999, pp. 136–141.

E-mail address: csohler@uni-paderborn.de (C. Sohler).

0925-7721/$ – see front matter
doi:10.1016/j.comgeo.2004.12.006

Keywords: Computational geometry; Algorithm; Delaunay triangulation; Data compression

1. Introduction

The Delaunay triangulation is an important structure in computational geometry and its application areas. One of its applications is to serve as a terrain model in geographic information systems and flight simulation, as explained below.

Suppose we want to store a map of a piece of the earth's surface—including the relief—on a computer, say for a flight simulation or a geographic information system. Typically, we are given a reasonable dense set of points on the surface and we want to have some mechanism to approximate the height of all other points. A well-known approach is to triangulate the points in two dimensions (ignoring the height of the points) and then lift the points to its height in the 3-dimensional space. Then the height of an arbitrary point of the surface is approximated by the height of the intersection of a vertical line through that point and the triangulation.

The Delaunay triangulation is well suited for this technique. It is canonical, that is, it is uniquely defined by the point set and it maximizes the minimal angle of the triangulation. This means it avoids long and skinny triangles resulting in a nice looking map.

Delaunay triangulations used in this context are usually large; sometimes they consist of several millions of triangles. A natural question is how to transmit such a triangulation over a slow communication link (e.g., the internet) from a server to a client. One possibility to do so is to use a compression algorithm for (more or less) general 3D triangle meshes [1–3,5–7,9,11–26,28–30]. Such an algorithm compresses both the topology of the triangulation and the geometry (the vertex positions). In our case, we do not need the topology of the triangulation, it is already defined by the point positions. Thus, such an algorithm will always result in unnecessary transmission overhead.

A second possibility is to transmit the point set only and to recompute the triangulation at the client. There is a wide variety of algorithms that compute a Delaunay triangulation in $O(n \log n)$ time [4,10]. It is also known that computing the Delaunay triangulation is at least as difficult as sorting and it therefore cannot be computed in $\omega(n \log n)$ time in general.

Recently, Snoeyink and van Kreveld [27] introduced a third possibility, which is also considered in this paper: compute a special order for the point set. If the point set arrives in this order at the client, the Delaunay triangulation can be recomputed in $O(n)$ time. Provided the server has already computed the Delaunay triangulation, it is possible to compute such an order in $O(n)$ time as shown in [27]. The order computation algorithm in [27] has $O(\log n)$ phases. In each phase an independent set I of $\Omega(n')$ points is removed from the current triangulation, where n' denotes the number of points in the current triangulation. Then the created holes are retriangulated and each removed point gets a reference pointer to its enclosing triangle. This is followed by a (canonical) DFS or BFS on the triangles of the current triangulation. The order of the point set is then given by the time the enclosing triangles for the points in I are visited.

The reconstruction algorithm also has $O(\log n)$ phases corresponding to the phases of the order computation algorithm. At the beginning of each phase, the points can be located by the same canonical DFS or BFS as in the order computation algorithm. This way, the point location for I can be done in $O(n')$