Research Article Impact Factor: 0.621 ISSN: 2319-507X
PN Mulkalwar, IJPRET, 2014; Volume 2 (9): 139-144 IJPRET

INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

DELAUNAY TRIANGULATION AND MESH GENERATION

MULKALWAR P N[1], MULKALWAR AKSHAY[2], KHANDELWAL SHUBHAM[2]

1. Department of Computer Science, Amolakchand Mahavidyalaya, Yavatmal.
2. School of Computing Science and Engineering, VIT University, Vellore (Tamilnadu).

Accepted Date: 27/02/2014 ; Published Date: 01/05/2014

Abstract: The Delaunay triangulation is one of the most popular and most often used methods in problems related to the generation of meshes. The aim of this paper is to implement the algorithm for computing Delaunay Triangulation in Euclidean space(E^d). One of the main characteristics of this algorithm is its generality: it can be simply extended to triangulate point sets in any dimension. Also the methodology used for implementation is discussed.

Keywords: Delaunay triangulation, De Wall algorithm,

Corresponding Author: MR. MULKALWAR P N

Access Online On:

www.ijpret.com

How to Cite This Article:

PN Mulkalwar, IJPRET, 2014; Volume 2 (9): 139-144

PAPER-QR CODE

INTRODUCTION

The Delaunay triangulation of a set of points is one of the classical problems in computational geometry. It was discovered in 1934 by the French mathematician Boris Nikolaevich Delone or Delaunay [1]. Many algorithms have since been proposed by computer scientists as well as mathematicians. In 1985, Guibas and Stolfi [2] proposed a Divide and Conquer algorithm which achieves the optimal bound of O(n log n). Later on, Steve Fortune [3] proposed a Sweepline algorithm for Voronoi diagrams which also achieves this bound. Unfortunately, they are hard to implement, so that engineers instead use incremental insertion algorithms because of their simplicity and the fact that they are not hard to implement. Guibas and Stolfi [2] proposed an incremental algorithm using a walking strategy to locate points which achieves O(n1/2) per point location, and it was improved by Mucke [4] to an O(n1/3) per point location. Although these algorithms provide good implementations, they are still sub-optimal algorithms.

Meshes can be divided into structured (points are distributed regularly) and unstructured (irregularly placed points). Structured meshes are not the topic of this paper. Structured meshes are generated by three methods in general [5]:

- The Delaunay triangulation- first, Delaunay triangulation of the existing points is made; if the mesh is not sufficiently fine, new points are generated, triangles are sub divided and the mesh is refined to make the approximation more precise,
- advancing more techniques – the point generation goes in the direction from the boundary of the domain inwards,
- finite octree – the sparse regular grid over the domain is subdivided so that at the boundaries the cell size is consistent with the boundary point spacing; the mesh is then further modified by the boundaries.

Computing meshes for surfaces dynamically was considered impossible a decade ago. The present implementations have large code for implementation. The code is designed in modules reusable such that it can be used to generate mesh as well as track other aspects of a given point. Triangulation can be Constrained, Regular and Delaunay. In this project we present implementation of 2D Delaunay triangulation, provide an algorithm for Delaunay Triangulation in 3-Dimensional space and present an overview of difference between 2D and 3D Delaunay triangulation. Delaunay triangulation is named after Boris Delaunay. The initial inspiration was to form a mesh where each triangle’s three points lie on the edge of a circle that does not contain any other point. This idea was further developed for d-dimensions which is currently used in computing 3D convex hulls. This algorithm was recently used for fitting triangular faceted surfaces to digital terrain data.

The paper is organized as follows. Definitions and a taxonomy of Delaunay triangulation algorithms are presented in Section 2. The proposed algorithm is described in detail in Section 3, together with some optimization techniques. The performances of the proposed solution are evaluated on a number of datasets and compared with other solutions in Section 4. Conclusions are drawn in the last section.

2. Delaunay Triangulation

For a set P of points in the (d-dimensional) Euclidean space, a Delaunay triangulation is a triangulation DT(P) such that no point in P is inside the circum-hypersphere of any simplex in DT(P). It is known[6] that there exists a unique Delaunay triangulation for P, if P is a set of points in general position; that is, there exists no k-flat containing k + 2 points nor a k-sphere containing k + 3 points, for $1 \leq k \leq d - 1$.

The duality between DTs and Voronoi diagrams is well known [7] and therefore algorithms are given for the construction of DT from Voronoi diagrams. However, direct construction methods are generally more efficient because the Voronoi diagram does not need to be computed and stored. There are several types algorithms for creating Delaunay triangulation. According to [8] in general it can be classified as follows:

- local improvement - starting with an arbitrary triangulation, these algorithms locally modify the faces of pairs of adjacent simplices according to the circum-sphere criterion;
- on-line (or incremental insertion) - starting with a simplex which contains the convex hull of the point set, these algorithms insert the points in P one at a time: the simplex containing the currently added point is partitioned by inserting it as a new vertex. The circum-sphere criterion is tested on all the simplices adjacent to the new ones, recursively, and, if necessary, their faces are fliipped;
- incremental construction - the DT is constructed by successively building simplices whose circum-hyperspheres contain no points in P;
- higher dimensional embedding - these algorithms transform the points into the E^{d+1} space and then compute the convex hull of the transformed points; the DT is obtained by simply projecting the convex hull into E^d; for a comparison of the different approaches;
- divide & conquer (D&C) - this is based on the recursive partition and local triangulation of the point set, and then on a merging phase where the resulting triangulations are joined.

3. METHODOLOGY

Delaunay Triangulation (3D) is a triangulation such that the circumsphere of every d-simplex is empty. It should not contain any of the given points in its interior. To summarize, Delaunay triangulation minimizes the maximum radius of the semicircle and maximizes the minimum angle. Loosely put, the Delaunay triangulation is the most efficient way to draw triangles between pairs of points. Each point is connected by lines to its closest neighbours, in such a way that all line parts form triangles, and do not intersect otherwise. No triangles overlap; in fact, the surface is completely covered with one nice layer of different triangular tiles. Its 3D-variant is important in creating virtual worlds for video games, among many other things. Although at first glance, obtaining the Delaunay triangulation seems to be almost trivial, in fact it's a quite complicated task, the more so if you want to do it efficiently for greater numbers of points. Quite a lot of ingenious algorithms have been devised, and the field is the subject of on-going research.

4. The Algorithm and Its Implementation

In this section a De Wall algorithm is presented which uses a claver idea to avoid the complications of joining phase in 3D. The algorithm is based on the D&C paradigm, but this paradigm is applied in a different way with respect previous DT algorithms [9] [10]. The general structure of D&C algorithms is: divide the input data into subset P1 and P2; recursively solve on P1 and P2; and merge the partial results S1 and S2 to build solution S.

The DeWall (Delaunay Wall) algorithm consists of the following steps:

- Select the dividing plane α , split P into the two subsets P1 and P2 and construct Σ_α ;
- starting from Σ_α, recursively apply DeWall on P1 and P2 to build Σ_1 and Σ_2 ;
- return the union of Σ_α, Σ_1 and Σ_2.

The whole De Wall algorithm can be written as follows:

Select the dividing plane α

Split P into two subsets P1 and P2 and construct trangulation Σ_α

recursively apply DEwall on P1 and P2 to built trangulation Σ_1 and Σ_2

return the union of Σ_α, Σ_1 and Σ_2

Explanation of each step in more detaisdfgbgl.

The splitting plane α is cyclically selected as a plane orthogonal to the axes of the E^3 space. Two subsets of input points obtained by the splitting should be of comparable cardinality.

The tetrahedral intersected by α i.e. simplex wall is constructed by simple incremental algorithm.

Then the adjacent simplices are constructed according to the Delaunay definition by searching for the Delaunay points. The point which is selected as Delaunay one is the which minimizes the fuction dd (Delaunay distance). To speed up the searching for Delaunay points, the uniform grid is used. The space input points is divided into a uniform grid where the number of cells is equal to the number of points. The search is then restricted to those cells contained in the bounding box of the circumscribed sphere with minimal radius.

For each tetrahedron from the simplex wall its triangles are stored in the suitable face list. The simplex wall is built by removing and expanding the triangles from AFL_α(the triangles intersected by the plane α). The wall is finished when the AFL_α is empty. The algorithm is then recursively applied to the pair (P1,AFL_1) and (P2, AFL_2), unless all active lists are empty(AFL_1 and AFL_2 are the triangles with all the points in P1 and P2 respectively)

4. CONCLUSION

We have presented an implementation of the well-known incremental algorithm for constructing Delaunay triangulations in any dimension. The New DT code is fully robust and outperforms the existing implementations. We believe that New DT can be used in real applications in spaces of 4- dimensions. In these dimensions, the simple filtering mechanism that we use for evaluating the geometric predicates may not be sufficiently efficient, and revert too often to exact computations. Solutions to this problem exist but have to be implemented into our d-dimensional kernel. We have shown empirical evidence that the algorithm satisfies its balance conditions and that it performs like a randomized incremental algorithm that is output sensitive to the number of vertices.

REFERANCES

1. Delone, B. N. Sur la sph´ere vide. Bul. Acad. Sci. URSS, Class. Sci. Nat. (1934), 793–800.

2. Guibas, L. J., and Stolfi, J. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. ACM Trans. Graphics 4 (1985), 74–123.

3. Fortune, S. A sweepline algorithm for the Voronoi diagrams. Algorithmica 2 (1987), 153–174.

4. E.P., M., and I. Saias I., Z. B. Fast Randomized Point Location Without Preprocessing in Two- and Three dimensional Delaunay Triangulations. CompGeom'96, Philadelphia (1996),274–283.

5. Handbook of Grid Generation. Edited by Thompson, J.F.,Soni, B.K., Weatherill,N.P., CRC Press, 1999

6. de Berg, Mark; Otfried Cheong, Marc van Kreveld, Mark Overmars (2008). Computational Geometry: Algorithms and Applications. Springer-Verlag. ISBN 978-3-540-77973-5.

7. F.P. Preparata and M.I. Shamos. Computational Geometry: an Introduction. Springer-Verlag, 1985.

8. Cignoni, P., Montani , C., Scopigno, R.:DeWall: A Fast Divide & Conquer Delaunay Triangulation Algorithm in E^d Computer-Aided Design, 30, 1989, pp. 333-341.

9. D. T Lee and B.J. Schachter. Two algorithms for constructing a Delaunay triangulation. Int.J. of Computer and Information Science, 9(3):219-242, 1980.

10. R.A. Dwyer. A faster divide and conquer algorithm for constructing Delaunay triangulations. Algorithmica, 2:137-151, 1987.