Pergamon

Computers & Geosciences Vol. 24, No. 2, pp. 193–196, 1998

Printed in Great Britain
PII: S0098-3004(97)00136-2
0098-3004/98 $19.00 + 0.00

SHORT NOTE

IMPROVEMENTS ON SLOAN'S ALGORITHM FOR CONSTRUCTING DELAUNAY TRIANGULATIONS

CHONG-WEI HUANG, and TIAN-YUAN SHIH
Department of Civil Engineering, National Chiao-Tung University, Hsin-Chu, Taiwan R.O.C.
(*e-mail*: tyshih@cc.nctu.edu.tw)

(Received 12 June 1997; revised 18 September 1997)

INTRODUCTION

Delaunay triangulation has been applied widely in a number of fields. Its definitions and properties have been fairly assembled and well described in Okabe, Boots, and Sugihara (1992). By aiming for better running efficiency, a number of TIN construction algorithms have been developed along with different approaches. Extensive reviews can be found in Lee and Preparata (1984), Aurenhammer (1991), Okabe, Boots and Sugihara (1992) and Tsai (1993). The $O(N \log N)$ algorithms are asymptotically optimal (Preparata and Shamos, 1985) in the worst situation. For engineering applications, the average performance of a triangulation algorithm is generally more important than its worst-case performance, because the latter tends to occur rarely in practice (Sloan, 1987).

Because Sloan's (1987) algorithm is efficient and the accompanying program is handy to use, it has been widely adopted. Nevertheless, some improvements could be made:

(1) Some calculations in the searching procedure can be replaced by a simple comparison.

(2) Because some triangles related to the supertriangle are deleted in the final triangulation, the empty-circle-criterion checks for these triangles are unnecessary.

(3) The efficiency may deteriorate if the supertriangle's selected size is too large. Even so, after removing the supertriangle-related data, the final triangulation generated with Sloan's (1987) algorithm is not guaranteed to be a convex hull.

(4) Other types of enclosure polygons may provide better efficiency than a supertriangle.

(5) To reduce the number of empty-circle-criterion checks, the domain is subdivided into m by m bins, where m is the nearest integer of $N^{1/4}$. This value can be modified to speed up the triangulation in terms of a strip-function.

The improvements are evaluated numerically in terms of the number of primitive operations invoked. The first primitive is the computation of a rank-three determinant, for determining whether the specified three points are in clockwise (CW) or counterclockwise (CCW) order. This information is provided to locate the triangle which encloses a newly inserted point (Guibas and Stolfi, 1985; Preparata and Shamos, 1985). This primitive is denoted herein as *CCW*.

The second primitive, denoted as *InCircle*, is the computation of a rank-four determinant. This operation is based on the empty-circle criterion, to check whether a point is located inside a triangle's circumcircle. If this test fails, the diagonal of the quadrilateral formed with three vertices of the triangle and the point is then swapped. The corresponding procedure is referred to as *Swap*. Guibas and Stolfi (1985) applied *CCW* and *InCircle* for the efficiency evaluation, while the *InCircle* and *Swap* primitives are grouped as a single primitive. Because the number of *Swap* invocations is different from that of *InCircle*, these two primitives should be observed separately. Therefore, *Swap* is used as the third primitive in this study.

Regarding the running time for these three primitives, two million tests indicate that the average cost of an *InCircle* is around 2.62 times that of a *CCW* for computation with 4-byte real numbers; meanwhile, the unit cost for *Swap* is about 0.85 times of a *CCW*. The overall quantity in expression of (2.62**InCircle* + 0.85**Swap* + *CCW*) is nearly identical to the running time complexity of Sloan's (1987) algorithm, excluding sorting.

The coordinates of all sampling points in the simulated datasets were generated by the linear congruential random-number generator developed by Wichmann and Hill (1987). The three initial seeds were set as 11, 10001, 3001, respectively. The sizes of tested datasets are of twelve different values, ranging from 20000 to 500000. A unit square domain was defined for each dataset.

PERFORMANCE MEASUREMENT OF SLOAN'S (1987) ALGORITHM

The number of invocations and complexities of Sloan's (1987) algorithm for triangulating the twelve datasets are listed in Table 1. The invocations of *CCW* contribute most to the complexity of Sloan's algorithm.

IMPROVEMENTS IN SLOAN'S (1987) ALGORITHM

Corresponding to the observations made on Sloan's algorithm described in the introduction, five revisions are devised.

The usage of previous CCW result in range searching

For each insertion, *range searching* is performed with the *TRILOC* algorithm (Fig. 1). Because both vertices and edges of a triangle are arranged in a counterclockwise order, if a point is at the right side of an edge in triangle T_i, this point is certainly at the left side of this edge in the adjacent triangle T_{i+1}. A *CCW* calculation in T_{i+1} for the shared edge could be saved. This common edge is defined as *DUMMY* in the following *TRILOC* algorithm.

The saving of *CCW* operations in *TRILOC* could account for each triangle searched. For T_i, where $k > i > 0$, the *CCW* operations required are at most 2, compared with 3 for the original algorithm. The expected number of *CCW* is thus equal to $(1 + 2)/2 = 1.5$, compared with $(1 + 2 + 3)/3 = 2$, assuming equal probabilities. For T_k, only two, instead of three, *CCW* operations are sufficient. For T_0, the expected number of *CCW* is still two.

Convexity and the size of supertriangle

Sloan (1987) stated that a supertriangle's size could be arbitrarily chosen. However, even with a large supertriangle, the convexity cannot be ensured. To maintain the convexity of the final triangulation, a sub-algorithm based on Graham's scan (Graham, 1972) is introduced. This process involves both *InCircle* testing and *Swap* procedures simultaneously on the boundary triangles. Additional complexity is introduced, that is $O(h)$ for *CCW* and $O(h^2)$ for *InCircle*, where h is the number of boundary points. Because h is usually small in practice, this addition is insignificant in most situations.

```
Algorithm TRILOC(P,DUMMY,T);
Input :
        P : the new point to be inserted
        DUMMY : edge in T not to be checked and initialized as zero
        T : a triangle to be determined if P is enclosed in it
Output :
        T : the triangle which P is found to be inside of it
begin
  for i:=3 downto 1 do          {loop clockwise over the three edges of T}
    if i<> DUMMY then  {CCW operations can be neglected for this edge}
      j ← i mod 3 + 1
      V1 ← the i-th vertex in T
      V2 ← the j-th vertex in T     {assign the two endpoints of an edge}
      if CCW(V2,V1,P) then                          {P is outside of T}
        Ta ← the i-th edge in T
        DUMMY ← EDG(Ta,T)             {find edge in Ta adjacent to T}
        T ← Ta                  {keep searching on its adjacent triangle}
        TRILOC(P,DUMMY,T)                                 {until found}
        return T
        stop
  return T    { T is thus found which passed over three consecutive CCW }
end;
```

Table 1. Performance measures of Sloan's (1987) algorithm

N	*InCircle*	*Swap*	*CCW*	Overall
20000	279883	110164	341805	1168738
40000	571986	226298	769707	2460664
60000	874701	347728	1230169	3817454
100000	1481890	591432	2224529	6609798
150000	2269786	910495	3593755	10314515
200000	3062822	1232094	5127269	14199143
250000	3858760	1555160	6685379	18117216
300000	4658556	1880112	8293699	22097211
350000	5464612	2208221	9938671	26132942
400000	6269434	2535685	11568281	30149530
450000	7064454	2858209	13206254	34144601
500000	7881517	3191808	14873314	38235925
Complexity	$10.5N^{1.0313}$	$3.85N^{1.0385}$	$3.78N^{1.1573}$	$27.0N^{1.0794}$

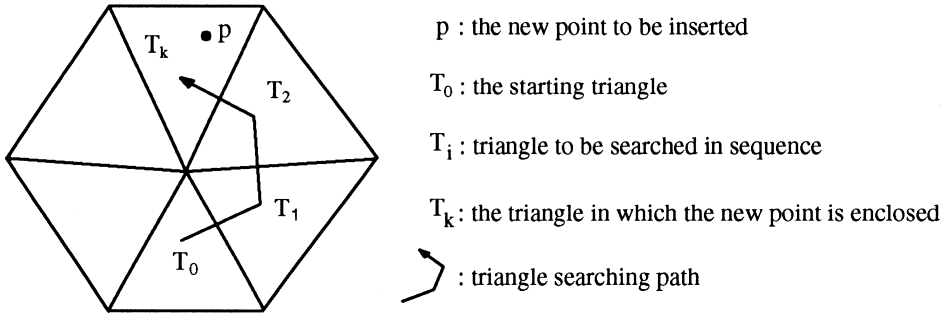

Figure 1. Triangle search algorithm.

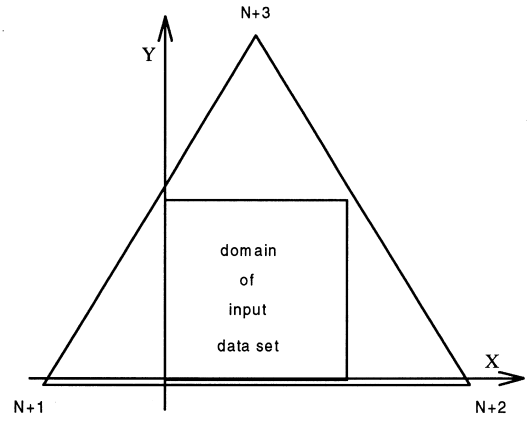

Figure 2. Redefined supertriangle.

Numerical experiments have demonstrated that the computation is reduced when the corners of supertriangle are approaching the enclosing boundary of points. The coordinates corresponding to the three vertices of the redefined supertriangle are set as (−1.25, −0.025), (2.250, −0.025) and (0.5, 1.5) (Fig. 2) during testing.

Incircle reduction

According to Sloan's (1987) algorithm, the triangles that contain one or more of the supertriangle vertices will be deleted in the end. Therefore, the Delaunay properties of these triangles do not need to be checked. It can be done by simply adding a statement "IF V3 < N + 1 THEN" before *InCircle* test.

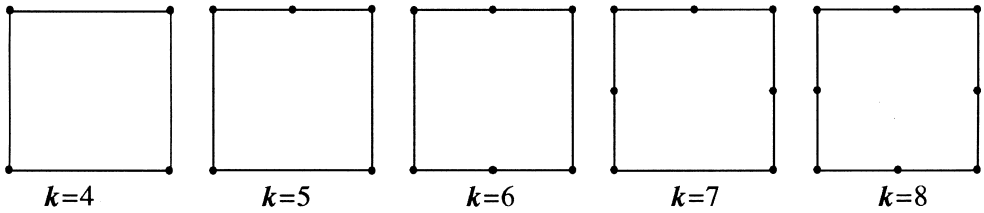

Figure 3. Five super-k-gons and allocation of vertices.

Table 2. Performance measures of the improved algorithm

N	*InCircle*	*Swap*	*CCW*	*Overall invocations*		
				improved	original	reduction rate %
20000	219780	82685	130688	776794	1168738	33.5
40000	458497	173721	271803	1620728	2460664	34.1
60000	700328	266334	415713	2476956	3817454	35.1
100000	1197964	457815	711884	4239692	6609798	35.9
150000	1832718	703030	1091718	6491015	10314515	37.1
200000	2479502	954502	1480004	8787626	14199143	38.1
250000	3126654	1205473	1866975	11083461	18117216	38.8
300000	3776679	1458097	2265838	13400119	22097211	39.4
350000	4433563	1714111	2659575	15732504	26132942	39.8
400000	5096932	1972967	3065706	18096690	30149530	40.0
450000	5753890	2228897	3462349	20432103	34144601	40.2
500000	6429787	2494069	3873410	22839411	38235925	40.3
Complexity	$7.50N^{1.0410}$	$2.59N^{1.0498}$	$3.96N^{1.0510}$	$25.8N^{1.0435}$	$27.0N^{1.0794}$	

Supertriangle and super-k-gon

For a square-shaped domain, polygons with four to eight vertices (Fig. 3) are compared with supertriangle in terms of computational efficiency. From numerical experiments, the case of $k = 8$ performs the best. However, the improvement decreases as N increases.

Stripsort

Sloan's (1987) utilizes *binsort* to reduce the number of searches required. Instead of dividing the domain into bins, *stripsort* could seemingly produce the same situation. A strip is the bins in the same row. Following Sloan's (1987), the number of strips can be defined as the nearest integer of $N^{1/4}$. In this study, $0.5\sqrt{N} + \sqrt[4]{N} + 15$ is found to be the most efficient definition for a uniformly distributed unit square.

The performance measures of the final version is listed in Table 2. The overall complexity is reduced from $27.0N^{1.0794}$ to $25.8N^{1.0435}$, which is due mainly to the reduction of *CCW* invocations from $3.78N^{1.1573}$ to $3.96N^{1.0510}$. For the 500000-points example, the overall invocations of the improved version is around 40.3% reduced from the original. This reduction seems to be scalable for $N > 500000$.

CONCLUSIONS

Five modifications to Sloan's (1987) algorithm have been described. Numerical experiments indicate that the running time is reduced to about 59.7% of the original for up to 500000 points randomly distributed in a unit square. Furthermore, the final triangulation resulting from the revised algorithm will be convex. Code may be downloaded by anonymous FTP from the server FTP.IAMG.ORG.

REFERENCES

Aurenhammer, F. (1991) Voronoi diagrams — A survey of fundamental geometric data structure. *ACM Computing Surveys* **23**(3), 345–405.

Graham, R. L. (1972) An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters* **1**, 132–133.

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

Lee, D. T. and Preparata, F. P. (1984) Computational geometry — a survey. *IEEE Transaction on Computers* **C-33**(12), 1072–1101.

Okabe, A., Boots, B. and Sugihara, K. (1992), *Spatial Tessellation Concept and Applications of Voronoi Diagrams*. John Wiley & Sons, New York, 532 pp.

Preparata, F. P. and Shamos, M. I. (1985) *Computational Geometry — An Introduction*. Springer-Verlag, New York, 398 pp.

Sloan, S. W. (1987) A fast algorithm for constructing Delaunay triangulations in the plane. *Advances in Engineering Software and Workstations* **9**(1), 34–55.

Tsai, V. J. D. (1993) Fast topological construction of Delaunay triangulations and Voronoi diagrams. *Computers & Geosciences* **19**(10), 1463–1474.

Wichmann, B. and Hill, D. (1987) Building a random-number generator. *Byte* **2**(3), 127–128.