Journal of ELECTRICAL ENGINEERING, VOL. 59, NO. 1, 2008, 49–52

COMMUNICATIONS

DELAUNAY TRIANGULATION BENCHMARKS

Denis Špelič [*] — **Franc Novak** [**] — **Borut Žalik** [*]

In this communication we propose an initial set of 2D Delaunay triangulation benchmarks for checking the correctness of algorithms and discovering possible flaws. A tool for verification of the generated triangulation is provided. The tool reports typical errors like the existence of unused points, missing edges, non-Delaunay triangles or degenerated triangles. While the tool has been primarily conceived for the verification of the implemented algorithms it may also be used for their debugging in the early design phase.

K e y w o r d s: computational geometry, Delaunay triangulation, benchmarks

1 INTRODUCTION

Evaluation and comparison of different tools and algorithms is a difficult task. For commercial tool purchasers, it is important to understand how well a tool accomplishes the required task in order to select the one that works best for the kind of problems a user will face. For the tool developer it is vital to provide some means for evaluation of the efficiency of the implemented algorithms. Similarly, the academia needs to evaluate the speed and robustness of the developed algorithms on general case studies in order to detect possible programming flows and for further improvements.

In order to evaluate the software or hardware solutions, benchmarks are applied in different technical areas. While no single test (*ie*, benchmark) can fully characterize target system performance, the results of a set of representative benchmarks can give valuable insight into expected real performance. A benchmark set typically consists of a collection of items described in a common format representing a range of problem tasks for a given problem domain. It allows different parties (academia, tool developers and users) an unbiased evaluation of tools and algorithms in terms of speed, effectiveness and quality of result.

Surprisingly, computational geometry society has not set-up such benchmarks in spite of the fact that solutions are strongly oriented towards solving different engineering tasks. Strategic Directions in Computational Geometry Working Group Report [1] outlines the major achievements of the field and gives a list of accomplishments in individual areas that can be applied in practice. Although the theoretical bounds on time and space complexity of the proposed algorithms are known, practical aspects introduce other problems, in particular from the viewpoints of robustness, implementation and experimental evaluation. A prerequisite to deciding which algorithms to implement in a tool for a given engineering task is to know which ones perform well in practice. This could be effectively done if the corresponding benchmarks were available.

2 TOWARDS SETTING UP COMPUTATIONAL GEOMETRY BENCHMARKS

Setting-up a set of benchmarks for specific problem domain is a challenging issue. Diversity of the sub-fields of the computational geometry (as presented in Section 2 of the Computational Geometry Working Group Report [1]) indicates that the problem of defining general computational geometry benchmarks is almost certainly intractable. It is likely to be more effective to develop separate sets of benchmarks focusing on specific problems such that are reasonably representative for the class of applications dealing with individual sub-fields.

In this regard we propose the benchmark tool and data sets for 2D Delaunay triangulation as one of the basic subfields of the computational geometry attracting interest of both academia and commercial tool designers.

The solutions to problem of finding the Delaunay triangulation for a set of points in the plane represent the key issue in different areas of computational geometry [2], [3]. In the last decades, different algorithms for constructing Delaunay triangulation have been proposed [4–10][1]. They are classified in different groups as for example: incremental insertion algorithms, gift-wrapping algorithms, divide and conquer algorithms, convex-hull based algorithms and sweep-line algorithms. These algorithms differ in the elapsed CPU time, consumed computer memory, implementation demands, numerical stability, and the sensibility to different distributions of input points.

[1] Notice, however, that the above references are given just to illustrate the research field and applications, the list is by no means exclusive.

[*] Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova ulica 17, 2000 Maribor, Slovenia; zalik@uni-mb.si [**] Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

ISSN 1335-3632

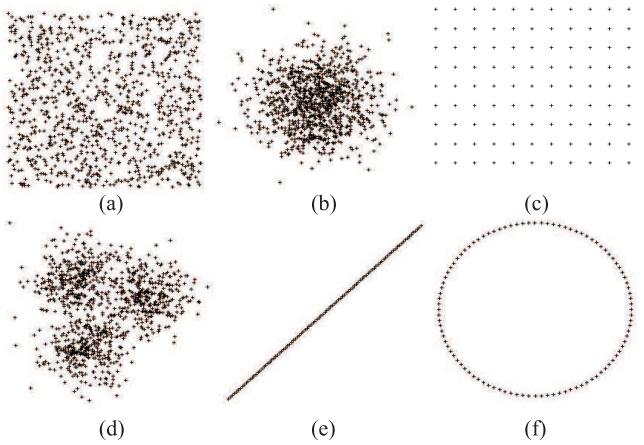

Fig. 1. Artificially generated benchmark data sets: (a) normal, (b) Gaussian, (c) grid, (d) clusters, (e) line, (f) circle.

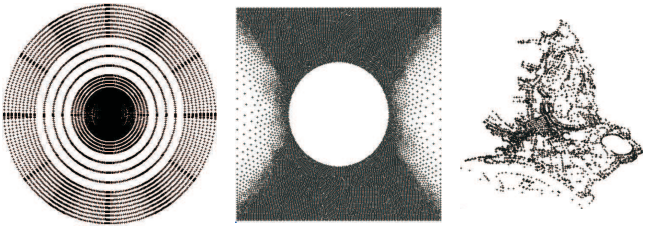

Fig. 2. Engineering benchmark data sets: (a) electromagnetic data, (b) mechanical data, (c) GIS data.

Table 1. Artificial benchmark data sets.

name	type of distribution	type of coordinates	Range
U_1K_I.pnt	Normal	integer	$[-10^3, -10^3, 10^3, 10^3]$
U_1K_F.pnt	Normal	floating-point	$[-10^{15}, -10^{15}, 10^{15}, 10^{15}]$
G_1K_I.pnt	Gaussian	integer	$[-10^3, -10^3, 10^3, 10^3]$
G_1K_F.pnt	Gaussian	floating-point	$[-10^{15}, -10^{15}, 10^{15}, 10^{15}]$
Gr_1K_I.pnt	grid	integer	$[-10^3, -10^3, 10^3, 10^3]$
Gr_1K_F.pnt	grid	floating-point	$[-10^{15}, -10^{15}, 10^{15}, 10^{15}]$
C_1K_I.pnt	clusters	integer	$[-10^3, -10^3, 10^3, 10^3]$
C_1K_F.pnt	clusters	floating-point	$[-10^{15}, -10^{15}, 10^{15}, 10^{15}]$
L_1K_I.pnt	line	integer	$[-10^3, -10^3, 10^3, 10^3]$
L_1K_F.pnt	line	floating-point	$[-10^{15}, -10^{15}, 10^{15}, 10^{15}]$
Ci_1K_I.pnt	circle	integer	$[-10^3, -10^3, 10^3, 10^3]$
Ci_1K_F.pnt	circle	floating-point	$[-10^{15}, -10^{15}, 10^{15}, 10^{15}]$

Delaunay triangulation finds application in different fields such as communications, GIS, signal processing, multimedia, micromechanics, etc, [11–16][1]. The implementation of a stable and fast Delaunay triangulator has always been a challenge.

In practice, there is a frequent need of verification if a generated triangulation of a given data set is correct. Besides, implementations of algorithms may have quite different performance characteristics on different types of data sets, which is of crucial importance for the target applications. Individual attempts to test the efficiency of triangulation algorithms on various set of points have been reported (*eg*, Su and Drysdale [17]) yet they have not resulted in a general verification platform.

We propose a set of benchmark data that would provide a means of comparison amongst different implementations of algorithms for constructing 2D Delaunay triangulation. In addition, a tool for verification of the correctness of the generated triangulation of the given benchmark data set is given.

3 BENCHMARK DATA SETS

Proposed benchmarks are collections of:

- artificial data sets,
- engineering data sets.

Artificial data sets are generated using different rules and distributions as shown in Figure 1. These data sets differ in number of points (we use 1 000, 10 000, 100 000 and 1 000 000 points), type of coordinates and the range of the bounding box surrounding the points (Table 1). For denoting the data sets, we use the following convention:

(type of distribution)_(number of points)_
_(type of coordinates).pnt.

For example, the data sets for the normal distribution with integer coordinates 10 000, 100 000 and 1 000 000 points are denoted U_10K_I.pnt, U_100K_I.pnt and U_1M_I.pnt.

Engineering benchmark data (Table 2) are obtained from different engineering disciplines (*ie*, GIS, mechanics, electromagnetics). Characteristic examples are shown in Figure 2.

3.2 Benchmark file template

The benchmark data sets are stored in ASCII files. The first line contains an integer determining the number of points being stored in the file. The remaining part includes x and y coordinates of points. The file template is given below.

$$
\begin{array}{l}
n \\
x_1 y_1 \\
x_2 y_2 \\
\quad \vdots \\
x_n y_n
\end{array}
$$

Benchmark data sets can be downloaded from the WEB side http://gemma.uni-mb.si/dtt.

4 VERIFICATION OF TRIANGULATIONS

4.1 Verification procedure

Figure 3 shows the verification procedure of Delaunay triangulation.

1. A benchmark file is selected from the available Benchmark datasets.

Table 2. Engineering benchmark data sets.

name	description	type of coordinates	range
	Electromagnetic data:		
EM_46625_F.pnt	electrical field around high-power lines	floating-point	$[-10^{15}, -10^{15}, 10^{15}, 10^{15}]$
	Mechanical data:		
MD_15559_F.pnt	measurement of forces in a sheet-metal plate	floating-point	$[-10^{15}, -10^{15}, 10^{15}, 10^{15}]$
	GIS data:		
GIS_4897_F.pnt	points from irregular triangular network representing terrain	floating-point	$[-10^{15}, -10^{15}, 10^{15}, 10^{15}]$
GIS_193360_F.pnt	points from irregular triangular network representing Slovenian city of Maribor	floating-point	$[-10^{15}, -10^{15}, 10^{15}, 10^{15}]$

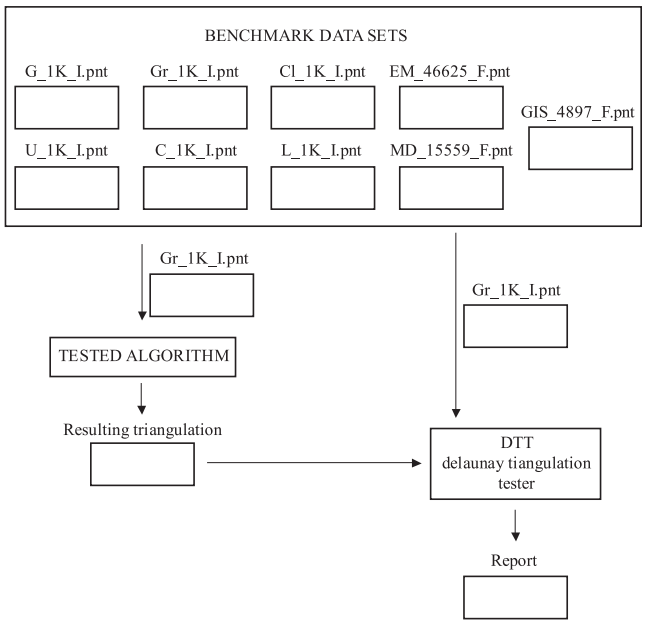

Fig. 3. Verification procedure.

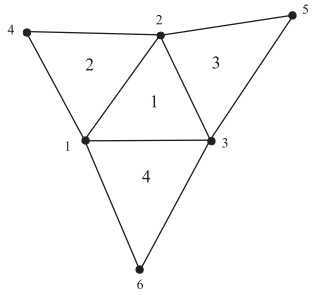

Fig. 4. Example of Delaunay triangulation.

2. Delaunay triangulation is performed by the algorithm that we want to test.
3. The result is stored in the form described in Section 4.2.
4. The resulting triangulation is checked by the DTT program (Delaunay Triangulation Tester). The inputs to DTT program are the original benchmark and the resulting triangulation file. DTT program checks the correctness of the resulting triangulation and reports possible errors. Error messages are described in Appendix.

DTT is available free for download at http://gemma.uni-mb.si/dtt. The compressed package includes DTT program and user manual.

4.2 Interchange format

The structure of the ASCII file storing the constructed Delaunay triangulation is given below:

m
$x \;\; i \;\; j \;\; k \;\; n_{ij} \;\; n_{jk} \;\; n_{ki}$

The first line in the file defines the number of generated triangles: m. Next, the description of triangles is given. Each triangle is denoted by a unique index x: $x = 1, 2, 3, \ldots, m$. A triangle is described by the indices of its vertices $i, j, k \in [1, n]$. Next, for each triangle, its neighboring triangles are given $n_{ij}, n_{jk}, n_{ki} \in [1, m]$. If the neighboring triangle does not exist it is denoted by 0.

For clarity, the ASCII file of the triangulation in Figure 4 is given below.

```
4
1: 1 2 3 2 3 4
2: 1 4 2 0 0 1
3: 2 5 3 0 0 1
4: 3 6 1 0 0 1
```

5 CONCLUSION

Proposed benchmark data sets for 2D Delaunay triangulation represent an initial step towards establishing a general verification platform supporting the strategic directions in computational geometry [1]. We are soliciting feedback on the usefulness of the benchmark as well as proposals for additional data sets that would contribute to the efficiency of the verification process.

References

[1] Strategic Directions in Computational Geometry, ACM Computing Surveys 28 No. 4 (1996), 591–606, Working Group Report (Tamassia R., ed.).

[2] de BERG, M.—van KREVELD, M.—OVERMARS, M.—SCHWARZKOPF, O.: Computational Geometry, Algorithms and Applications, Springer-Verlag, Berlin, 2nd rev. ed. 2000.

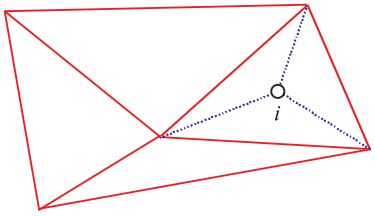

Fig. 5. Point i is not used.

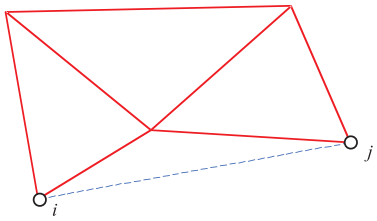

Fig. 6. Missing edge i, j.

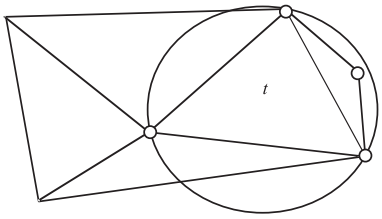

Fig. 7. Non-Delaunay triangle t.

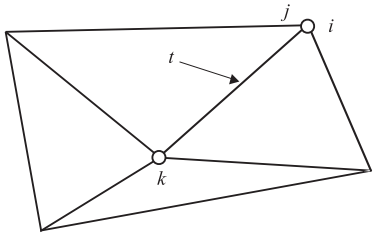

Fig. 8. Degenerated triangle t: duplicated vertices i, j.

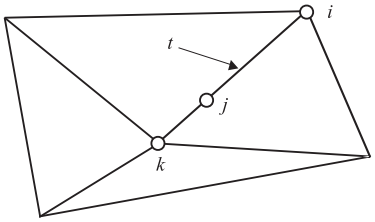

Fig. 9. Degenerated triangle t: collinear vertices i, j, k.

[3] O'ROURKE, J.: Computational Geometry in C, Cambridge Tracts in Theoretical Computer Science, 2001.

[4] KOLINGEROVÁ, I.: Modified DAG Location for Delaunay Triangulation, Computational Science – ICCS 2002, Part III, Amsterdam - The Netherlands, LNCS 2331, Springer-Verlag, 2002, pp. 125–134.

[5] FANG, T. P.—PIEGL, L.: Delaunay Triangulation Using a Uniform Grid, IEEE Computer & Graphics Applications **13** No. 3 (1993), 36–47.

[6] CIGNONI, P.—MONTANI, C.—SCOPIGNO, R. De Wall: a Fast Divide & Conquer Delaunay Triangulation Algorithm in Ed: Computer-Aided Design **30** No. 5 (1998), 333–341.

[7] EDELSBRUNNER, H.—SEIDEL, R.: Voronoi Diagrams in Linear Expected Time, Discrete Computational Geometry **1** No. 1 (1986), 25–44.

[8] FORTUNE, S. J. A Sweepline Algorithm for Voronoi Diagrams: Algorithmica **2** No. 1 (1987), 153-174.

[9] ŽALIK, B.: An Efficient Sweep-Line Delaunay Triangulation Algorithm, Computer-Aided Design **37** No. 10 (2005), 1027–1038.

[10] KOHOUT, J. KOLINGEROVÁ, I.—ŽÁRA, J. Parallel Delaunay Triangulation in E2 and E3 for Computers with Shared Memory: Parallel Computing **31** No. 5 (2005), 491–522.

[11] MEGUERDICHIAN, S.—KOUSHANFAR, F.—QU, G.—POTKONJAK, M.: Exposure In Wireless Ad-Hoc Sensor Networks, ACM SIGMOBILE 7/01, Rome, 2001, 139–150.

[12] LIEBEHERR, J.—NAHAS, M.—SI, W.: IEEE Journal on Selected Areas in Communications.

[13] PARK, D. G.—CHO, H. G.—KIM, Y. S.: A TIN Compression Method using Delaunay Triangulation, Int. Journal of Geographical Information Science (IJGIS) **15** No. 3 (2001), 255–269.

[14] PLANTIER, J.—BOUTTÉ, L.—LELANDAIS, S.: Defect Detection on Inclined Textured Planes Using the Shape from Texture Method and the Delaunay Triangulation, EURASIP Journal on Applied Signal Processing No. 7 (2002), 659–666.

[15] YAOPING, Y.—CHENGKE, W.: A Novel Video Coding Scheme Using Delaunay Triangulation, Journal of Visual Communication and Image Representation **9** No. 1 (1998), 80–86.

[16] ZHAO, Y.—TAY, F. E. H.—CHAU, F. S.—ZHOU, G. A Nonlinearity Compensation Approach based on Delaunay Triangulation to Linearize the Scanning Field of Dual-Axis Micromirror: Journal of Micromechanics and Microengineering **15** No. 10 (2005), 1972–1978.

[17] SUP.—DRYSDALE, R. L. S.: A Comparison of Sequential Delaunay Triangulation Algorithms, Proceedings of 11th Annual Symposium on Computational Geometry, ACM Press, Vancouver, 1995, pp. 61–70.

Received 9 June 2007

APPENDIX: Description of error messages

Delaunay triangulation tester (DTT) detects four types of errors:

- *Unused point.* In this case, at least one non-doubled input point is unused in the final triangulation. The situation is shown in Figure 5. DTT outputs the error message *Error: point i is not used*, where i is the index of the unused point.
- *Missing edge.* This error occurs if the convex hull property of the Delaunay triangulation is violated. The situation is shown in Figure 6. In the example, edge ij is missing. The error message *Error: missing edge ij* is reported.
- *Non-Delaunay triangle.* If non-Delaunay triangle t exists in the triangulation, the empty circle property is violated (see Figure 7). The error message *Error: non-Delaunay triangle t* is generated in this case.
- *Degenerated triangles*: two possible cases of degenerated triangles are detected:

a) Duplicate vertices: vertices i and j coincide (*ie*, they are located in the same point). The error message *Error: triangle t is degenerated because of duplicated vertices i, j.* is reported. The situation is shown in Figure 8.

b) Collinear vertices: If triangle t is determined by three collinear vertices i, j, k, the error message *Error: triangle t is degenerated because of collinear vertices i, j, k* is reported. The situation is shown in Figure 9.