ELSEVIER

Comput. Methods Appl. Mech. Engrg. 128 (1995) 153–167

Computer methods in applied mechanics and engineering

Fast Delaunay triangulation in three dimensions

H. Borouchaki[a], S.H. Lo[b,*]

[a]*Institut National de Recherche en Informatique et en Automatique, Domaine de Voluceau – Rocquencourt B.P. 105, 78153 Le Chesnay Cedex, France*

[b]*Department of Civil and Structural Engineering, The University of Hong Kong, Hong Kong*

Received 4 December 1994; revised 7 April 1995

Abstract

An efficient algorithm for Delaunay triangulation of a given set of points in three dimensions based on the point insertion technique is presented. Various steps of the triangulation algorithm are reviewed and many acceleration procedures are devised to speed up the triangulation process. New features include the search of a neighbouring point by the layering scheme, locating the containing tetrahedron by random walk, formulas of important geometrical quantities of a new tetrahedron based on those of an existing one, a novel approach in establishing the adjacency relationship, the use of adjacency table and the management of memory. The resulting scheme is one of the fastest triangulation algorithms known to the authors, which is able to generate tetrahedra generation rate of 15 000 tetrahedra per second for randomly generated points on a HP 735 machine.

1. Introduction

A triangulation of a set of points in three-dimensional space is a decomposition of the convex hull of the points into non-overlapping tetrahedra. The Delaunay triangulation, in which the circumsphere of every tetrahedron in the triangulation contains no point in its interior, is of particular interest to mathematicians as well as to engineers. It has been used extensively in the design of efficient algorithms and in many direct applications such as interpolation [1], contouring [2] and mesh generation [3–5]. Since the Delaunay triangulation has some optimal properties in two dimensions and efficient global and incremental algorithms exist for their construction, they have been used in finite element mesh generation to produce well-proportioned elements [6, 7].

In a recent paper by Rajan [8], some new optimality results for Delaunay triangulation in higher dimensions are discussed. It is suggested that the Delaunay triangulation is the most compact one in the sense that:

(1) the min-containment sphere is the smallest,

(2) the circumspheres of the tetrahedra incident on an interior point is closest to the point, and

(3) the weighted average of the square of edge lengths is the smallest.

Algorithms for the construction of Delaunay triangulations in n-dimensional space for $n \geq 2$ are given by Bowyer [9], Watson [10] and Avis and Bhattcharya [11]. The incremental algorithms of Bowyer and Watson is especially popular in the large scale three-dimensional finite element generation [12, 13]. The popularity of the method may be accounted for by the speed with which a large number of points can be

* Corresponding author.

SSDI 0045-7825(95)00854-3

processed, and the existence of triangulation for any distribution of points which can later be corrected to retrieve the boundary of an arbitrarily shaped three-dimensional object [14].

For Delaunay triangulation in three dimensions, the estimated time complexity of triangulating n points is $\mathcal{O}(n^{4/3})$ for the algorithm of Bowyer, and higher for the algorithms of Watson and Avis et al. Joe [19] presented an algorithm which makes use of local transformations to construct a Delaunay triangulation of a set of 3D points. The empirical time complexity of the algorithm is $\mathcal{O}(n^{4/3})$, and $\mathcal{O}(n^2)$ in the worst case. Edelsbrunner et al. [20] proposed a scheme in the worst case time of $\mathcal{O}(n^2)$ for the construction of 3D Delaunay triangulation by projecting the given 3D points onto a paraboloid in four dimensions. The convex hull of the 4D points on the paraboloid is constructed, and the 3D triangulation is then obtained from an appropriate portion of the convex hull.

In the incremental algorithm of Bowyer or Watson, the points are processed one at a time. In a typical step of point insertion, the tetrahedra whose circumsphere contains the insertion point are identified and deleted. New tetrahedra are constructed in the cavity left behind by the tetrahedra removed. Hence, the efficiency of the triangulation algorithm depends on how fast we can identify the tetrahedra to be removed and determine correctly the cavity for insertion, and the speed with which the circumcentres, the circumradii and adjacency relationship of the new tetrahedra are calculated.

In this paper, we propose a fast and reliable method to determine the tetrahedra to be removed and how the insertion cavity has to be corrected before triangulation. It is interesting to note that the circumcentres and circumradii of the new tetrahedra can all be calculated from the tetrahedra to be deleted, thus saving the trouble of solving a 3×3 linear system for each new tetrahedron created. An intelligent scheme is also devised in the determination of the adjacency relationship, which requires no searching or matching at all, enabling the entire implementation to have a linear time complexity for reasonable even point distribution up to more than one million tetrahedra.

2. Delaunay triangulation

The Delaunay triangulation of a set of points is defined to be a triangulation such that the circumsphere of every tetrahedron in the triangulation contains no point from the set in its interior. Such a triangulation exists for a given set of points in three dimensions, and it is the dual of the Voronoi diagram [15]. The triangulation is unique if the points are in general positions, i.e. no five points are cospherical.

A tetrahedron T is said to be Delaunay valid with respect to a point P if P does not lie inside the circumsphere of T. A tetrahedron T in a triangulation of a set of points is called Delaunay tetrahedron if T is Delaunay valid with respect to every point in the set. A triangulation of a set of points in three dimensions is called Delaunay triangulation of the point set if every tetrahedron in the triangulation is a Delaunay tetrahedron. The following lemma provides the basis for many algorithms in the construction and verification of Delaunay triangulations.

Lemma of Delaunay [16]

Let $\mathbf{T}(S)$ be a triangulation of the point set S. The necessary and sufficient condition that no point of S is contained in the circumsphere of any tetrahedron in the triangulation is that any two adjacent tetrahedra in the triangulation are Delaunay valid with respect to each other's vertices.

The property of circumsphere containment is the key to the various algorithms that construct Delaunay triangulation for a given set of points. In three dimensions, Watson's algorithm starts with a tetrahedron containing all points to be inserted, and new internal tetrahedra are formed as the points are entered one at a time. At a typical stage of the process, a newly inserted point is tested to determine which circumsphere of the existing tetrahedra contains the point. The associated tetrahedra are removed leaving an insertion polyhedron containing the newly inserted point. New tetrahedra are created by connecting the new point to all triangular facets on the surface of the insertion polyhedron. Combining these with the tetrahedra outside the insertion polyhedron produces a new Delaunay triangulation which contains the newly added point. Triangulation is done when all the points are inserted and processed sequentially.

We follow the Watson's procedure except that we start with a cube of five tetrahedra which is large enough to contain all the points. The triangulation is achieved by a point insertion algorithm, in which each cycle of point insertion can be divided into three distinct stages.

(i) For a newly inserted point, identify all the tetrahedra whose circumsphere contains the point. The cavity left behind upon the removal of these tetrahedra forms the insertion polyhedron, which will be referred to as the CORE.

(ii) Owing the finite precision arithmetics, the triangulation facets on the boundary of the core have to be verified and corrected before they are connected with the inserted point to form tetrahedra.

(iii) The triangulation of the core should be a trivial step. However, the adjacency relationship of the tetrahedra has to be available, and will be frequently referred to, throughout the triangulation process. Hence, when tetrahedra are generated, we have to establish the adjacency relationship for the new tetrahedra and modify that of the old tetrahedra attached to the surface of the core.

2.1. Determination of the core

When a new point P is inserted in a Delaunay triangulation, it is required to find all the tetrahedra whose circumsphere contains the point P. A simple method to determine these non-Delaunay tetrahedra is to scan through all the existing tetrahedra for those whose circumsphere contains the point P. However, a more efficient approach is to start with the tetrahedron which contains the inserted point P and find the others by means of the adjacency relationship. In this way, the boundary of the core is given by the common faces of two tetrahedra for which one is positive in the sphere inclusion test while the other fails.

The tetrahedron which contains the insertion point P is called the BASE, which is an integral part of the core. It seems that finding a tetrahedron whose circumsphere contains P is easier than finding the base. However, it does not always work in case a wrong decision is made in the sphere inclusion test due to numerical errors. The difficulty can be explained in terms of an example in a two-dimensional situation for the inclusion of a new point P.

Let AFB be the triangle which contains P in a Delaunay triangulation, and A, B, C, D, E are five points roughly on the circumference of a circle as shown in Fig. 1. In the sphere inclusion test, point P is found in the circumcircles of triangles ABC and ADE, but not in that of ACD. In this case the core is a disconnected piece if we have started building the core with triangle ADE. However, if we insist that the core has to be started with the containing triangle, then we can delete triangle ADE and yet obtain a valid core composed of triangles AFB and ABC.

Search for the base

In principle, we would like to start the searching process for the containing tetrahedron of the inserted point P from a point as close to point P as possible. However, when there is no better

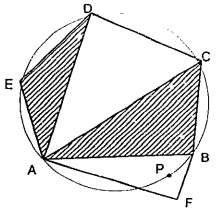


Fig. 1. Insertion of point P.

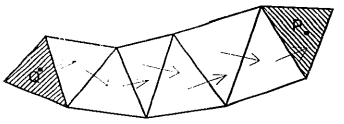


Fig. 2. Search for the containing tetrahedron of point P.

information as to where the point P is with respect to the existing tetrahedra, we can only arbitrarily start the search from the last constructed tetrahedron. Suppose that we start with a point Q at the centroid of the last constructed tetrahedron. Q moves across a face which separates the moving point Q and the destination point P as shown in Fig. 2.

Since the neighbours of a tetrahedron are known, the position of point Q can be updated easily. The process is repeated until point P is reached, and the containing tetrahedron is found. Nevertheless, there is a weakness in this scheme as when point Q moves into the triangle from position 1 as shown in Fig. 3, it will go round the triangle in counter-clockwise order and will loop indefinitely.

To avoid looping round a closed circuit, there is a suggestion that all the triangles visited should be flagged so that point Q will not go into those triangles again. However, from the same figure, if point Q starts its journey from position 3, it will go to positions 4, 5, 6, 7, 2, and then since 3 is already visited it is forced to go to 1. Thereafter, point Q will be separated from point P by the visited triangles which effectively form a closed ring.

Instead of recording the elements visited, recording the directions (faces) visited may help but still cannot solve the problem, as you do not know if you should go to the alternative direction when you are given the choice. When you have started from position 1 and completed a circuit and go back to triangle 2, all directions have been visited, again you do not know what to do and where to go. A solution to overcome the difficulty of looping is by direct crossing from point Q to point P following the line of intersection as shown in Fig. 4. The triangle T_1 containing Q is known. The face of T_1 which is hit by ray QP is determined. Go across the face along ray QP to triangle T_2. Repeat the process until we reach T_4 and point P.

A more economic solution both in terms of practical implementation and CPU time is the approach by random walk. Instead of following a systematic rule in approaching P, whenever more than one choice is given, invoke the random number generator to decide which way to go. Owing to the random nature of the approach, point Q will not loop indefinitely in a closed circuit. Implementation experience with work examples of triangulation up to one million tetrahedra demonstrates that it is efficient and reliable.

Boundary of the core

Starting from the base, the core can be easily established by applying the sphere inclusion test to the neighbouring tetrahedra. Continue applying the test to more adjacent tetrahedra until all non-Delaunay tetrahedra are determined. The boundary of the core is made up of triangular facets which are the faces between two tetrahedra that give positive and negative response to the sphere inclusion test.

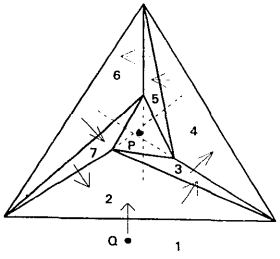


Fig. 3. Looping round a triangle.

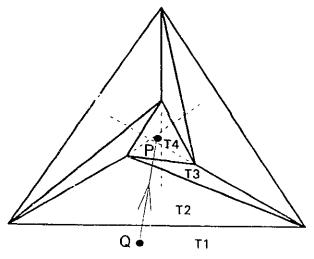


Fig. 4. Search along the line of intersection.

2.2. Origin of inconsistency

At this preliminary stage, the core consists of all the non-Delaunay tetrahedra with respect to the insertion point P. One of the major difficulties in the implementation of the Delaunay triangulation algorithm by point insertion is to ensure consistency in the sphere inclusion test based on finite arithmetic calculations. An inconsistent decision in the sphere inclusion test may result in a disconnected core as explained in Section 2.1. Imprecise numerical calculation may also introduce tetrahedra with negative volumes.

Delaunay triangulation is not unique unless all the points are in general positions. On a two-dimensional plane, a degeneracy occurs when four points are cyclic giving rise to two different Delaunay triangulations that can be formed by the four points as depicted in Fig. 5.

When a fifth point P is introduced, difficulties can arise unless both existing triangles are removed. In Fig. 5, the algorithm will generate a consistent triangulation if both triangles ABC and ACD are regarded as non-Delaunay with respect to point P. A consistent triangulation will also be generated if neither triangle is deleted upon the introduction of P. However, if triangle ACD is deleted but not triangle ABC, then new triangulation would be structurally inconsistent. This situation may arise when four points are cyclic or almost cyclic, because finite precision arithmetic that is used in the Delaunay test may cause one of the two triangles to be accepted into the cavity list. In three-dimensional space, five points lying on the surface of a sphere can be connected up to form either two or three tetrahedra. When a sixth point is introduced, a similar algorithm will fail unless all of these tetrahedra are either deleted or retained.

Another type of degeneracy in three space, which is due to the nature of Delaunay triangulation rather than the result of numerical error, can occur when four points are coplanar and cyclic. It is possible for these four points to become the vertices of a tetrahedron of zero volume, which is known as *sliver*.

Using higher precision arithmetics can only postpone the problem to solve more cases which are near to degeneracy, but not solve it entirely. Baker [6] proposed a solution in which tolerances depending on the values of the data and the machine precisions are applied to all real number calculations. In the present implementation, the sphere inclusion test is done based on double precision arithmetics. This allows us to obtain a triangulation which is as close to Delaunay triangulation as possible.

2.3. Correction of the core

Inconsistency of the sphere inclusion test is to be supplemented and corrected by the visibility check or the positive volume test. In Fig. 6, a triangular facet ABC is visible to point P if $\boldsymbol{a}\cdot\boldsymbol{n}>0$, where $\boldsymbol{a}$ is a vector joining a point of triangle ABC to point P and $\boldsymbol{n}$ is a vector normal to the triangle. Volume of tetrahedron $ABCP$, $\mathrm{vol}(ABCP) = AP\cdot(AB\times AC) = \boldsymbol{a}\cdot\boldsymbol{n}$.

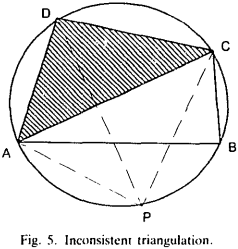
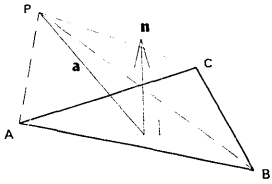


Fig. 5. Inconsistent triangulation.



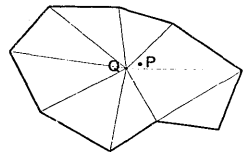


Fig. 6. Visibility test.

Fig. 7. An existing point Q is deleted by the inserted point P.

'Poor shaped' tetrahedra can also be avoided at the same time if we insi. that their volumes have to be greater than a certain threshold value. The condition that every facet on the boundary of the core is visible to point P also guarantees that the core is a single connected piece. The visibility check, although simple, is a necessary and sufficient test to ensure that the core is a star-shaped region with respect to the insertion point P. In fact, we can adopt other inclusion rules as which tetrahedra are to be deleted by the insertion point P to obtain different triangulations; the closer to the empty sphere criterion, the closer is the resulting triangulation to Delaunay triangulation. In the extreme case, upon the introduction of point P only one tetrahedron is deleted, the tetrahedron which contains P. Four new tetrahedra are created by joining P to the four faces of the deleted tetrahedron. Of course, by this simple rule, we are far away from the Delaunay triangulation. Though we can obtain a fast and valid triangulation of the given set of points, the tetrahedra so generated are flat and almost degenerated. In summary, the visibility check ensures the validity of the triangulation, and using higher precision arithmetics in the sphere inclusion test guarantees that the resulting triangulation is close to a Delaunay one.

In some rare occasions, an existing point may be found at the interior of the core as shown in Fig. 7. As all the tetrahedra connected to this point are removed, the point will also be removed by the newly inserted point. A remedy to this situation is to reinstate a tetrahedron linked to this point. The visibility test that follows will reinstate more tetrahedra if necessary to ensure the validity of the core and that each tetrahedron would have a positive volume.

2.4. *Triangulation of the core*

The triangulation of the core can be easily constructed by connecting point P to each of the facets on the boundary of the core. Such a triangulation exists as each facet is visible to P. This local triangulation of the core together with the existing tetrahedra outside the core form a new triangulation of all the inserted points including point P. A more delicate issue of triangulation is to determine the element adjacency relationship, i.e. for each tetrahedron we have to identify its four neighbouring elements. There is no problem in establishing the adjacency relationship between the old tetrahedra attached to the boundary facets of the core and the new tetrahedra that fill up the interior of the core. However, to determine the adjacency relationship between the new tetrahedra inside the core is less straightforward.

Each common face between two tetrahedra is identified with an edge on the boundary of the core as shown in Fig. 8. Tetrahedra $ABCP$ and $BADP$ are neighbours if they share common edge AB on the boundary of the core. Hence, the adjacency relationship for each tetrahedron can be established by identifying the three edges on the boundary of the core and determining the three neighbours through a matching process of common edges.

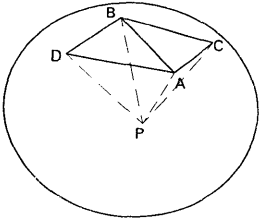


Fig. 8. Tetrahedra $ABCP$ and $BADP$ are neighbours.

3. Efficiency considerations

The efficiency of the triangulation scheme is greatly enhanced by the introduction of many relatively new and theoretically sound accelerating procedures in the three distinct phases of triangulation.

3.1. Search for the base by means of a background grid

The CPU time required in the search for the base of an inserted point will be excessive if we are dealing with a large system of points. The scale of the search can be much reduced with the help of a background grid. In a two-dimensional triangulation, suppose we are equipped with a background grid of regular cells. Before triangulation, all the insertion points are swept through once to determine which cells they are belonging to. When a triangle is constructed, the points which are the vertices of the triangle are given a flag of this triangle. Hence, unconnected points will be flagged zero, and a non-zero flag represents the triangle to which the point attaches. During triangulation, point P is inserted and the cell which contains P is determined. We have to devise a scheme to search for a nearby cell which contains a point with non-zero flag. A sequence in which cells close to point P will be scanned first is to conduct the search in a spiral fashion until the target point is found as shown in Fig. 9.

In three dimensions, the searching path is not a spiral, but something like the enveloping structure of an onion. When no point can be found in an inner surface, go one level out to an outer surface and search again until the first point with nonzero flag is encountered.

Let (i_0, j_0, k_0) be the address measured along three orthogonal axes of the cell which contains the

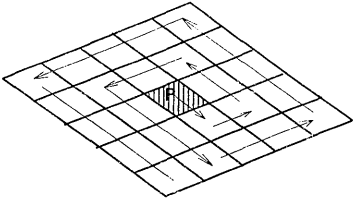


Fig. 9. Search in the form of a spiral on a 2D plane.

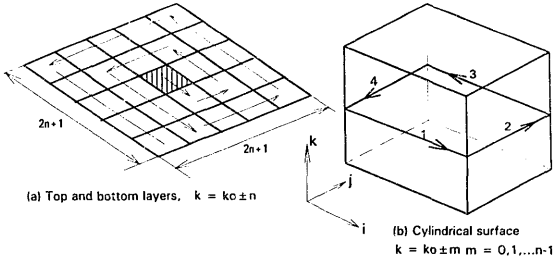


Fig. 10. Search of the surface at level n.

inserted point P, then the surface at level n, $n = 1, 2, 3, \ldots$ etc., is composed of a top layer and a bottom layer and four vertical faces which is in the form of an open cylinder as shown in Fig. 10.

The search on the top and the bottom layers is similar to that of the two-dimensional case. As for the cylindrical part, we have to go round the cylinder in four straight line segments **1**, **2**, **3** and **4** as shown in Fig. 10(b). At a typical level $k = k_0 \pm m$, with $m = 0, 1, n-1$, we have

(1) $(i_0 - n, j_0 - n) \rightarrow (i_0 + n - 1, j_0 - n)$

(2) $(i_0 + n, j_0 - n) \rightarrow (i_0 + n, \quad j_0 + n - 1)$

(3) $(i_0 + n, j_0 + n) \rightarrow (i_0 - n + 1, j_0 + n)$

(4) $(i_0 - n, j_0 + n) \rightarrow (i_0 - n, \quad j_0 - n + 1)$

The use of regular grid is especially efficient for even distribution of points, which makes the searching process a local one with constant time complexity independent of the number of points in the entire triangulation [17]. The use of a background grid is particularly effective for large-scale problems, say more than 10 000 points. It allows the triangulation algorithm to have a linear time complexity for random point distributions up to more than one million points. For very irregular point distributions, the use of octree data structure may be more advantageous [18]. However, a fair comparison between the two schemes is rather difficult due to the variations in the size of the problem, the implementation techniques, the characteristics of the point distribution, the sequence of insertion, the searching method, etc. In fact, the searching by a layering scheme in three-dimensional triangulation may not be so easily implemented in a spatial subdivision based on octree technique.

3.2. *Heredity of geometrical quantities*

Many geometrical quantities of a new tetrahedron such as the circumcentre, the circumradius and the normals of the faces can be easily calculated based on the same quantities of an old tetrahedron. It is reminded that a new tetrahedron is constructed by connecting the insertion point P with a triangular facet on the boundary of the core. However, the triangular facet to which P is connected is a face of a deleted tetrahedron: and it is from this old tetrahedron that the relevant geometric quantities are derived for the new tetrahedron.

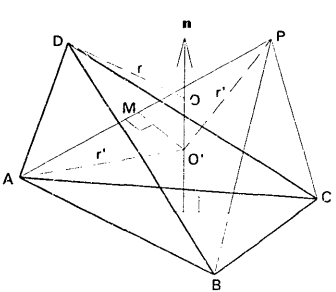
Let (O', r') be the circumcentre and circumradius of tetrahedron $ABCP$ and (O, r) be those of an old tetrahedron $ABCD$ as shown in Fig. 11, then new circumcentre O' is given by

$$O' = O + \lambda \boldsymbol{n} \quad \lambda \in \mathcal{R}$$

where $\boldsymbol{n} = AB \times AC$ is a normal vector to the face ABC.

$$\|O'A\| = \|O'P\| \Rightarrow MO' \perp AP$$

where M is the mid-point of AP.



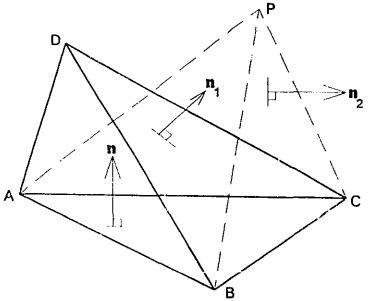


Fig. 11. Transfer of geometric quantities.

Fig. 12. Transfer of normal to tetrahedron $ABCP$.

$$MO' \perp AP \;\Rightarrow\; AP \cdot (MO + \lambda \boldsymbol{n}) = 0 \;\Rightarrow\; \lambda = \frac{AP \cdot OM}{AP \cdot \boldsymbol{n}}$$

$$AP \cdot OM = (OP - OA) \cdot \frac{OA + OP}{2} = \frac{\|OP\|^2 - \|OA\|^2}{2} = \frac{d^2(O,P) - r^2}{2}$$

where $d(O, P)$ is the Euclidean distance between centre O and point P. The value of $d^2(O,P) - r^2$ is already calculated in the sphere inclusion test, and $AP \cdot \boldsymbol{n}$ is the volume of tetrahedron $ABCP$ which is also available from the visibility test. λ and hence the circumcentre O' of the new tetrahedron can be readily calculated without forming and solving a linear 3×3 system. Knowing O', r' is given simply by the Euclidean distance between O' and P. Usually, the square of distance, r^2, is computed and stored as almost all calculations are based on the square of the distance rather than the distance itself.

Let $\boldsymbol{n}$, $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ be the normals to the faces ABC, BCD and BCP, respectively as shown in Fig. 12. Since $\boldsymbol{n}$, $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are normal to the line segment BC, $\boldsymbol{n}$, $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are contained on the plane normal to BC. Hence, $\boldsymbol{n}$, $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are not linearly independent vectors, and a linear relationship of the vectors exists which can be written as

$$\boldsymbol{n}_2 = \boldsymbol{n}_1 + \mu \boldsymbol{n}$$

$$CP \cdot \boldsymbol{n}_2 = 0 \;\Rightarrow\; CP \cdot (\boldsymbol{n}_1 + \mu \boldsymbol{n}) = 0 \;\Rightarrow\; CP \cdot \boldsymbol{n}_1 + \mu (CP \cdot \boldsymbol{n}) = 0$$

$$\Rightarrow\; \mu = \frac{-\mathrm{vol}(BCDP)}{\mathrm{vol}(ABCP)}$$

The volume function for each facet is available and $\mathrm{vol}(ABCP)$ is already calculated in the visibility test, hence μ and $\boldsymbol{n}_2$ can be readily calculated. The normals on the other faces of tetrahedron $ABCP$ are obtained in a similar way.

3.3. Adjacency relationship

In two-dimensional triangulations, the insertion core is a polygon, and there is no difficulty in establishing the adjacency relationship between triangles by constructing elements following the boundary contour of the polygon. In three and higher dimensions, the situation is quite different. The boundary of the insertion core is a surface, and there is no obvious order for the boundary faces following which the adjacency relationship of the tetrahedra so generated could be established in a more or less natural manner without much calculations.

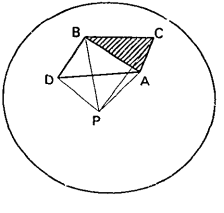
In Section 2.4, the adjacency relationship for the new tetrahedra by means of a matching process based on the edges on the boundary surface of the core is described. In a typical point insertion, the number of tetrahedra in a core is in the order of a hundred, and the searching and matching of tetrahedra to establish the adjacency relationship can be time-consuming compared to the other steps of the triangulation.

The adjacency relationship of the new tetrahedra inside the core is closely related to the adjacency relationship of the triangular facet on the boundary surface of the core. In fact, a one to one correspondence exists between the two topological structures, i.e. the three neighbours of a new tetrahedron can be identified with the three neighbouring triangles of the triangular facet to which the tetrahedron is attached. The fourth neighbour is, of course, the one opposite to the insertion point P, joining to the same boundary face from outside the core. As a result, search for neighbours for new tetrahedra can be avoided if the neighbours of a triangular facets on the boundary of the core can be directly determined.

The answer to this question is quite positive if we can take a closer look at the existing triangulation outside the core. The neighbouring triangle of a triangular facet on the core boundary can be determined by rotating about the common edge between the triangles through the tetrahedra connected to that edge. Since the neighbours of the tetrahedra are known, the determination of neighbouring triangles by this method is much faster, and no searching and matching is required. A better solution exists if we can make use of the tetrahedra to be deleted inside the core. The use of tetrahedra inside the core rather than those outside the core offers at least two advantages: (i) the path of rotation is shorter as the angle of turn is usually smaller, and (ii) the interior of the core is a continuous piece and there is no void inside whereas there may be voids outside the core.

In the present implementation, the tetrahedra are constructed directly on the boundary surface of the core using a technique similar to the advancing front approach. The construction process can be initiated by taking any triangular facet ABC on the boundary of the core. Tetrahedron $ABCP$ is formed by joining point P to triangular facet ABC. The construction front for this initial tetrahedron construction consists of three edges AB, BC and CA. To advance across the edge AB, the neighbouring triangle connected to AB has to be determined. Neighbouring triangle BAD can be determined by rotating about edge AB through the tetrahedra connected to this edge inside the core. The adjacency relationship between tetrahedra $ABCP$ and $BADP$ is established and the construction front is updated to four segments as shown in Fig. 13.

The construction process can be repeated until all the triangular facets on the boundary of the core are processed and the construction front will be reduced to zero. It is noted that the construction front can be closed in a fairly natural manner. An edge will be deleted from the front if a tetrahedron has already been generated in that direction. For instance, edge AB will be deleted if element $BADP$ has already been constructed. To check whether face BAD has already been used, simply refer to the element connected to the face BAD, tetrahedron $ABDQ$ on the opposite side of the core as shown in Fig. 14. Element $BADP$ is the neighbour of element $ABDQ$ sharing common face BAD. If the element



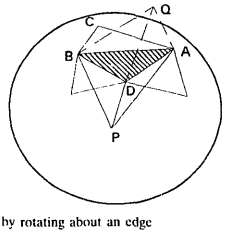
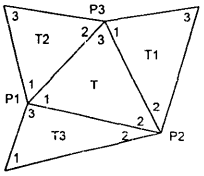


Fig. 13. Construction of tetrahedra by rotating about an edge

Fig. 14. Tetrahedra $BADP$ and $ABDQ$ are neighbours.



	P1	P2	P3
T1	3	2	1
T2	1	3	2
T3	3	2	1

Fig. 15. Adjacency table of triangle T.

number of this neighbour of $ABDQ$ is in the list of the newly constructed elements, then face BAD has already been treated and element $BADP$ has already been constructed and face BAD has to be ignored and no element should be constructed.

3.4. Adjacency table

For each tetrahedron, apart from the four vertices, the four neighbours are also stored. The element adjacency relationship is important in searching for the base of an insertion point, in confining the sphere inclusion test to a small number of elements near the base, and in the triangulation of the core. With a little additional memory, a more refined adjacency relationship and an adjacency table can be set up which allows us to have direct access to the neighbours. The structure of the adjacency table for triangle T with neighbours T_1, T_2 and T_3 opposite to vertices P_1, P_2 and P_3 is shown in Fig. 15.

The numbers on the diagonal are the node numbers of the neighbouring elements opposite to element T, whereas the off-diagonal coefficients a_{ij} indicate which node of element T_i is connected to node j of element T. From the adjacency table, we know that T is the third neighbour of T_1, the third neighbour of T_2 and the first neighbour of T_3. We also know which node of the first neighbour T_1 is connected to the second node P_2 of T, etc. In three dimensions, the adjacency table of a tetrahedron is a 4×4 matrix defined in a similar way. As the coefficients of the table only take values 1, 2, 3 or 4, by a compact storage scheme, four bytes is enough to store the entire table.

3.5. Memory management

For each tetrahedron, four vertices, four neighbours, an adjacency table, the circumcentre and the circumradius squared and the normals to its faces are stored. The newly created tetrahedra in an insertion process are put into the positions left behind by the deleted tetrahedra, and once these have been filled are added to the end of the list of tetrahedra. Similarly, the entries of the associated attributes of the tetrahedra are updated in the other data structures. In rare occasions, the newly created tetrahedra are fewer in number than those deleted, the unfilled positions of the old tetrahedra will be flagged. These positions can be taken up by the new tetrahedra created in the subsequent point insertions.

4. Numerical examples

The Delaunay triangulation algorithm was tested with examples with different number of points and distribution characteristics. The sets of points were taken from the boundary surface of practical engineering objects. The points were processed one at a time by the insertion algorithm, and the resulting mesh is the convex hull of the set of points. Six examples ranging from 2034 to 160 008 points were triangulated to give meshes of 12 750 to 107 7107 tetrahedra. These examples were done on a workstation HP735 which runs at a speed of about 150 MIP.

The first example as shown in Fig. 16 is a model of the bust of Victor Hugo, whose triangulated

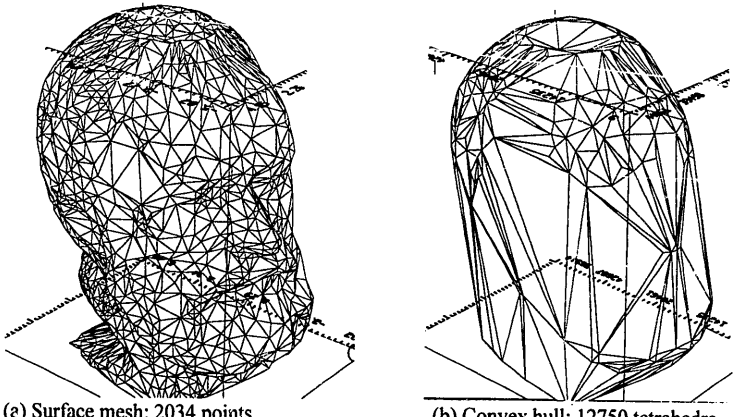


Fig. 16. Example 1—Bust of Victor Hugo (Courtesy of ENST).

surface consists of 2034 points. The second example is a car body, which is modelled using 4565 points as depicted in Fig. 17. Fig. 18 is a picture of a machine component, whose surface mesh consists of 3914 points. The model of a shuttle plane is shown in Fig. 19, the surface is discretized into 12 834 triangular facets using 6427 points. Example 5 is an object of quite an arbitrary shape modelled by 8505 points as shown in Fig. 20. The last example is the largest in scale of all, which consists of 160 008 randomly generated points. Table 1 gives a summary of these examples and the performance of the triangulation algorithm. In the last column of Table 1, the average number of tetrahedra that had to be visited from the starting point to the base is seven. This very short searching path reflects the effectiveness of using the grid in conjunction with the searching scheme by layers.

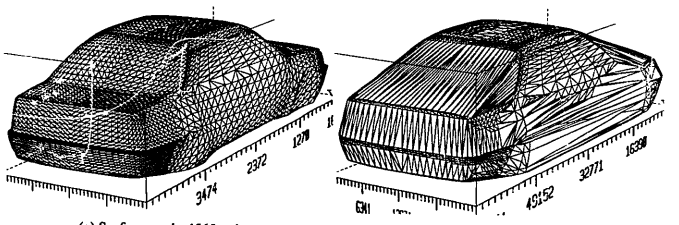
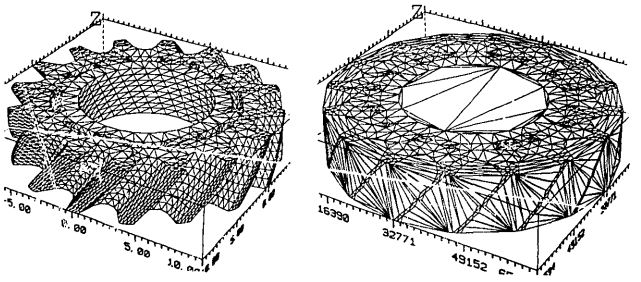
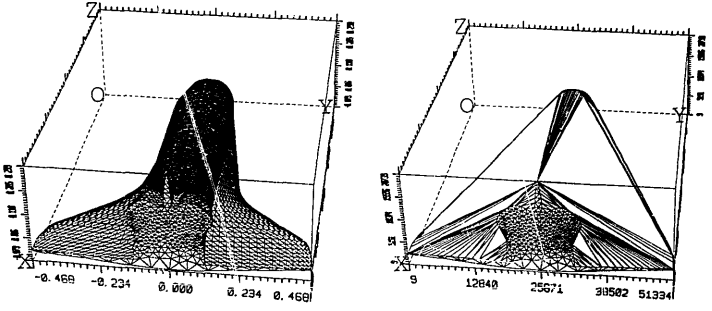


Fig. 17. Example 2—Peugeot, a car body.



(a) Surface mesh: 3914 points (b) Convex hull: 24942 tetrahedra

Fig. 18. Example 3—Engrenage, a machine part.



(a) Surface mesh: 6427 points (b) Convex hull: 43710 tetrahedra

Fig. 19. Example 4—Columbia, a shuttle plane (Courtesy of ONERA).

5. Conclusions and discussions

The detailed procedures for the construction of Delaunay triangulation by the point insertion algorithm are described and discussed. Virtually all stages of the triangulation have been reviewed and acceleration schemes have been devised in many parts of the algorithm to improve its general efficiency. The use of background grid is not new, but the search for a nearby tetrahedron following a layer by layer procedure away from the inserted point is proposed and formulated in details for the first time. The random walk approach in locating the containing tetrahedron has been proved to be efficient and reliable. Formulas of important geometrical quantities of a new tetrahedron related to those of an existing one are given, which greatly reduce the amount of arithmetic computations by completely avoiding the forming and solving of 3×3 linear systems.

However, the most important contribution of the paper is the introduction of a novel approach in

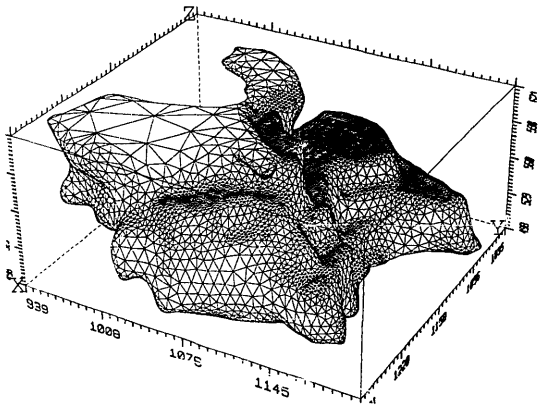


Fig. 20. Example 5—Ostab, arbitrarily shaped object, 8505 points and 54 698 tetrahedra (Courtesy of ENSMP (CGES)).

Table 1
Summary and statistics of work examples

	Points	Tetrahedra	Tetrahedra/s	Path
Victor Hugo	2034	12 750	16 139	6
Peugeot	4565	27 669	15 372	7
Engrenage	3914	24 942	15 687	12
Columbia	6427	43 710	15 837	7
Ostab	8505	54 698	14 893	6
Random	160 008	107 7107	14 724	6

establishing the adjacency relationship, in which no searching and element matching is required. Other techniques including the use of adjacency table and the management of memory further optimize the speed of the triangulation algorithm. It is interesting to note that all these ideas and the simplified formulas for the calculation of circumcentre and normals can be directly generalised to higher dimensional space. In fact, the three-dimensional triangulation algorithm reported here is an adaptation and refinement of a general n-dimensional triangulation scheme developed earlier.

As shown by the results of the numerical examples, the triangulation algorithm bears a linear time relationship with the number of insertion points or tetrahedra generated. This is in consistency with the theoretical predication of the triangulation time for even distribution of points. The present implementation is one of the most efficient three-dimensional triangulation schemes known to the authors, which is able to generate 15 000 tetrahedra per second on a HP735 machine. This rate can be maintained within the memory capacity of the machine used, and up to more than one million tetrahedra.

There is one more part in which considerable improvement could be made. The triangulation scheme will run faster if the optimal insertion sequence for the given set of points can be determined. The idea is to devise the proper order of point insertion for which the number of tetrahedra deleted would be minimized. More theoretical thinking will be needed and this will be a subject for future research

Acknowledgment

The second author is grateful to the Institut National de Recherche en Informatique et en Automatique of Paris for his kind invitation to participate in a mesh generation project over a period of more than three months. Particular thanks are due to Prof. M. Bernadou, Prof. P.L. George and Prof. E. Saltel for their cordial hospitality and help throughout the stay in INRIA.

References

[1] J.L. Brown, Vertex based data dependent triangulations, Comput. Aided Geometric Des. 8 (1991) 239–251.
[2] C.S. Petersen, B.R. Piper and A.J. Worsey, Adaptive contouring of a trivariate interpolant, in: G.E. Farin, ed., Geometric Modelling: Algorithms and New Trend (SIAM, Philadelphia, PA, 1987) 385–395.
[3] J.C. Cavendish, D.A. Field and W.H. Frey, An approach to automatic three-dimensional finite element mesh generation, Int. J. Numer. Methods Engrg. 21 (1985) 329–347.
[4] P.L. George, F. Hecht and E. Saltel, Fully automatic mesh generator for 3D domains of any shape, Impact Comput. Sci. Engrg. 2 (1990) 187–218.
[5] J.P. Wright and A.G. Jack, Aspect of three-dimensional constrained Delaunay meshing, Int. J. Numer. Methods Engrg. 37 (1994) 1841–1861.
[6] T.J. Baker, Tetrahedral mesh generation by a constrained Delaunay triangulation, in: E.N. Houstis and J. R. Rice, eds., Artificial Intelligence, Expert System and Symbolic Computing (Elsevier Science Publishers, B.V., North-Holland, 1992) IMACS.
[7] S.H. Lo, Delaunay triangulation of non-convex planar domains Int. J. Numer. Methods Engrg. 28 (1989) 2695–2707.
[8] V.T. Rajan, Optimality of the Delaunay triangulation in R^d, Discrete Comput. Geometry 12 (1994) 189–202.
[9] A. Bowyer, Computing Dirichlet tesselations, The Comput. J. 24 (1981) 162–166.
[10] D.F. Watson, Computing the n-dimensional Delaunay tesselation with application to Voronoi polytopes, The Comput. J. 24 (1981) 167–172.
[11] D. Avis and B.K. Bhattacharya, Algorithms for computing d-dimensional Voronoi diagrams and their duals, in: F.P. Preparata, ed., Advances in Computing Research, Volume 1 (JAI Press, 1983) 159–188.
[12] N.P. Weatherill and O Hassan, Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints, Int. J. Numer. Methods Engrg. 37 (1994) 2005–2039.
[13] P.L. George and F. Hermeline, Delaunay's mesh of a convex polyhedron in dimension d. Application to arbitrary polyhedra, Int. J. Numer. Methods Engrg. 33 (1992) 975–995
[14] P.L. George, F. Hecht and E. Saltel, Automatic mesh generator with specified boundary, Comput. Methods Appl. Mech. Engrg. 92 (1991) 269–288.
[15] H. Edelsbrunner, Algorithms in Combinatorial Geometry (Springer-Verlag, New York, 1987).
[16] B. Delaunay, Sur la sphere vide, Bull. Acad. Sci. URSS, Class. Sc. Nat., 793-800, 1934.
[17] Arne Maus, Delaunay triangulation and the convex hull of n points in expected linear time, BIT 24 (1984) 151–163.
[18] Homer H. Chen and Thomas S. Huang, A survey of construction and manipulation of Octree, Comput. Vision, Graphics and Images Proc. 43 (1988) 409–431.
[19] B. Joe, Construction of three-dimensional Delaunay triangulations using local transformations, Comput. Aided Geometric Des. 8 (1991) 123–142.
[20] H. Edelsbrunner, F.P. Preparata and D.B. West, Tetrahedrizing point sets in three dimensions, J. Symbolic Comput (1990).