# Computing the $n$-dimensional Delaunay tessellation with application to Voronoi polytopes†

D. F. Watson

*Department of Geology and Geophysics, Edgeworth-David Building, University of Sydney, NSW 2006, Australia*

**The Delaunay tessellation in $n$-dimensional space is a space-filling aggregate of $n$-simplices. These $n$-simplices are the dual forms of the vertices in the commonly used Voronoi tessellation. Several efforts have been made to simulate the 2-dimensional Voronoi tessellation on the computer. Additional problems occur for the 3 and higher dimensional implementations but some of these can be avoided by alternatively computing the dual Delaunay tessellation. An algorithm that finds the topological relationships in these tessellations is given.**

(Received October 1979, revised July 1980)

## 1. Introduction

The Delaunay tessellation, in 3-dimensional space, is an aggregate of space-filling, disjoint, irregular tetrahedra and may be thought of as a particular 3-dimensional triangulation. In 2-dimensional space it is an array of triangles. Such a triangulation provides an essential step for spatial analysis of punctulate data. Contouring programs usually require information concerning the degree of consanguinity amongst the data points, such as a grid or triangulation, before comparisons of the functional values at these data points are meaningful.

The Delaunay triangulation is also the geometric dual of the Voronoi tessellation. This tessellation has been used as a model in many areas of applied science. In particular, it is assumed to approximate the interface network in polycrystalline materials. To use it in this manner, the Voronoi tessellation must be considered to be a finite sample taken from an infinite population of which it is typical. However, in this respect, its form is not well known in $\geqslant$ 3-dimensional space.

This paper presents an algorithm for computing the structure of the Delaunay $n$-dimensional triangulation for any stochastic array of data points. Such an algorithm can be used to find all of the adjacency relationships amongst $N$ data points by triangulation in $n$-dimensions and in time of $O(N^{\lceil(2n-1)/n\rceil})$, provided that the $N$ data points are partially ordered.

Further, the structure of the dual Voronoi tessellation is immediately obtained by geometric conversion. However, only those Voronoi domains strictly interior to the Delaunay tessellation are available since the boundary of the Delaunay tessellation yields only partial information on the Voronoi structure used as a polycrystalline model.

### 1.1 *The Delaunay Structure*

The defining points, or nuclei, of the 3-dimensional Delaunay tessellation lie at the vertices of the tetrahedra. Several tetrahedra will share an edge and even more will share a vertex. The four vertices of each tetrahedron lie on the surface of a sphere and no other vertex of the array lies within that sphere (see Miles, 1970, p. 107). This is because the centre of the sphere is in the position of a vertex in the dual Voronoi tessellation, and this vertex is by definition equidistant from the four nuclei.

This description for 3-dimensional space is readily extended to $n$-dimensional space. Thus the tetrahedron, or 3-simplex, becomes an $n$-simplex. Likewise, the $n + 1$ vertices of each $n$-simplex lie on the $(n - 1)$-dimensional surface of an $n$-dimensional hypersphere. The centre of that hypersphere, the simplicial circumcentre, is a vertex in the dual $n$-dimensional Voronoi tessellation. In all cases the $n$-simplices of a Delaunay tessellation are space-filling and disjoint, while for stochastic arrays of defining nuclei they are also uniquely defined.

### 1.2 *The Voronoi Structure*

A Voronoi tessellation partitions $n$-dimensional space into convex polytopes and may be thought of as a network of interfaces formed by impingement of expanding hyperspheres, centred at the nuclei, and growing at a constant rate from time zero. Thus some $(n + 1)$-tuples of such hyperspheres meet a mutual conclusion at a point which is equidistant from the $n + 1$ nuclei. That point is a vertex in the Voronoi tessellation. Therefore these $(n + 1)$-tuples of nuclei may be used to define a vertex in a Voronoi tessellation, or equivalently, an $n$-simplex in a Delaunay tessellation.

According to Sommerville (1927, p. 106), the numbers of $i$-dimensional domains on an $n$-simplex is ${}_{n+1}C_{i+1}$, for $0 \leqslant i \leqslant n$. Then, since the geometrical dual in $n$-dimensional space of an $i$-dimensional domain is an $(n - i)$-dimensional domain, we know the numbers of $(n - i)$-dimensional domains that include a 0-dimensional domain, or vertex, in a Voronoi tessellation is ${}_{n+1}C_{n-i}$. This duality means that the total of $n$-simplices incident to a given nucleus in a Delaunay tessellation is equal to the number of vertices on the dual Voronoi domain associated with that nucleus. The total of $n$-simplices in common with a given neighbouring nucleus is the number of vertices on the $(n - 1)$-dimensional face formed with that neighbour in the dual Voronoi domains. Similar remarks apply to other such values.

### 1.3 *Previous Work*

The theoretical properties of the Voronoi planar tessellation have been established by Miles (1970, p. 103). A computer simulation, of the space-filling members of an aggregate in the plane, was accomplished by Crain (1972). He generated individual polygons and tabulated their measurements to form histograms and estimate ergodic moments.

Various algorithms have been used to find the boundaries of the Voronoi polygons. Knowing that the nearest neighbouring nucleus to some given nucleus must form an interface, one may proceed by tracing a pathway, always in the same rotational direction, around each polygon, as was done by Rhynsburger (1973, p. 141). A more sophisticated approach by Green and Sibson (1978, p. 170), traces the boundary adjustments required as a new nucleus, and thus a new polygon, is fitted into an established aggregate.

Perimeter tracing, however, though a principal technique for the plane, does not adapt to $\geqslant$ 3-dimensional space. Calling on the mathematical definition of the Voronoi domain as an intersection of half-spaces (see Rogers, 1964, p. 74), the 3-dimen-

†Editorial note: This paper and that by Bowyer (this issue) cover some material in common. As these contributions were received at approximately the same time, the Editor feels it only right to include both papers.

CCC-0010-4620/81/0024-0167 $03.00

sional tessellation may be computed by finding the interfaces with each neighbour and assembling the results. This approach was used by Watson and Smith (1975, p. 110), where the points, or nuclei, were distributed in the unit cube. Then for each point taken in turn, the perpendicularly bisecting plane with each other point was used to eliminate a portion of the cube. The portion remaining is the Voronoi polyhedron, associated with that particular nucleus, as defined by the intersection of its facial planes.

That procedure is a straightforward technique for single domains but involves duplicated calculations when used as the initial steps to assemble an aggregate. For example, the 3-plane intersection at a vertex of a Voronoi domain in 3-space must be calculated for four different domains before it can be fitted into the aggregate. To keep a list of such vertices for each Voronoi domain means that a vertex will appear in four such lists, but it will not necessarily be in numerically identical form as a result of digital truncation. Even maintaining a list of neighbours for each point requires a given relationship to be tabulated twice. Additionally, search times for determining relationships grows non-linearly. Short cuts for storing and searching are sparse because of the large and variable number of elements and links for each domain. For higher dimensions, these considerations tend against the possibility of meaningful results.

Fortunately, the dual Delaunay tessellation is less difficult, as was pointed out by Boots (1974, p. 26). Each Delaunay $n$-simplex may be represented by an $(n + 1)$-tuple of indices to the nuclei and all of its topological elements may be designated or enumerated by using the combinatorial properties of the $n$-simplex. The search for relationships is just the search for duplicate combinations of indices.

## 2. Difficulties

### 2.1 *Non-valid Vertices*

Problems, of course, occur in several areas. The first obvious difficulty is in determining which $(n + 1)$-tuples of nuclei to keep or reject. In other words, we want to select particular minimal clusters of data points that are immediately adjacent. An $(n + 1)$-tuple is the smallest cluster of data points that provides an $n$-dimensional hull. The number of combinatorially possible $(n + 1)$-tuples is ${}_N C_{n+1}$. Most of these are not real possibilities and for large numbers of data points, this is more than can be examined in a reasonable time, even in 2-dimensional space. Criteria such as shortest diagonal or greatest minimum angle have been used to discriminate between 3-tuples in planar studies. These criteria become somewhat involved in dimensions greater than two because the sum of the internal angles at the vertices of the $n$-simplex is not a constant as it is in 2-space.

However, the solution to this problem is, in theory, relatively easy. Simply by choosing only those $(n + 1)$-tuples that lie on $n$-dimensional hyperspheres which contain no other nuclei assures us of the correct result in all cases. This must be so because if any other nucleus is within some circumsphere defined by some $n + 1$ nuclei, then it is closer to the circumcentre than those $n + 1$ nuclei are, and a vertex in the Voronoi tessellation could not occur for those particular $n + 1$ nuclei. In effect, from the set of all combinatorially possible $(n + 1)$-tuples we reject all $(n + 1)$-tuples with non-empty circumspheres.

Now if we are given an initial aggregate of one or more $n$-simplices whose circumspheres are empty, we may introduce additional data points by observing which circumspheres are intersected. Then for each such intersected circumsphere, replace the $n$-simplex with all possible $n$-simplices formed from these $n + 2$ data points on condition that each new $n$-simplex has an empty circumsphere. The logic for this approach is shown in **Fig. 1.** When a new nucleus is introduced in this manner, the aggregate must remain a Delaunay tessellation since all circumspheres remain empty.

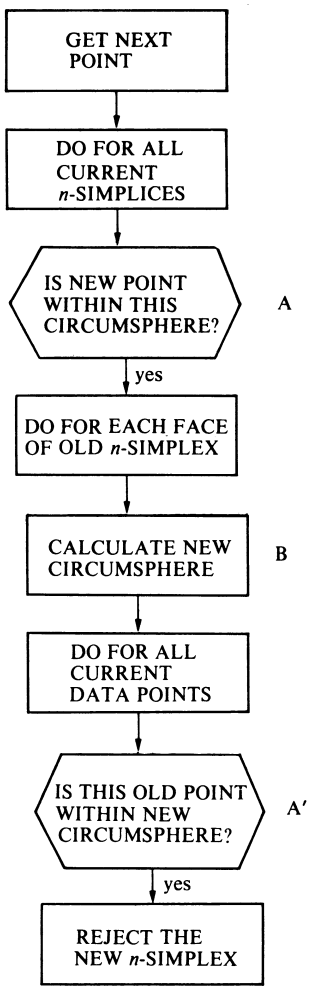


**Fig. 1 Logic for finding all $n$-simplices with empty circumspheres**

Fig. 1 shows that there are three main computational steps. The steps marked A and A′ both search for nuclei that intersect circumspheres but in opposite senses. At A we check all old circumspheres against a new point while at A′ we check all old points against a new circumsphere. The calculation marked B solves $n$ linear equations in $n$ unknowns.

### 2.2 *Degeneracies*

The main difficulty with this approach, however, concerns the so-called degenerate vertices in the Voronoi tessellation. Following Sibson (1978, p. 243), who expresses a degeneracy as the case when more than three domains meet at a vertex in 2-dimensional space, we consider the $n$th order degeneracy as the case when more than $n + 1$ Voronoi $n$-dimensional domains appear to share a vertex. Lower order degeneracies may also occur, for instance, when more than three 3-dimensional domains share an edge in 3-space, or in general, more than $n + 1 - i$ $n$-dimensional domains share an $i$-dimensional domain in $n$-space.

It will be noted, however, that the existence of a lower order degeneracy requires the existence of $n$th order degeneracies. This is readily observed in the case where four 3-dimensional polyhedra share an edge and therefore at least five polyhedra must share each vertex at the ends of that edge. It is also true in the general case; suppose that, for $1 \leqslant j \leqslant n, j + 2$ nuclei lie on the $(j - 1)$-dimensional surface of a $j$-dimensional hypersphere.

That $j$-dimensional hypersphere is a section of an infinite number of $(j + 1)$-dimensional hyperspheres. Any other nucleus will therefore form a $(j + 3)$-tuple of nuclei which will lie on the surface of a $(j + 1)$-dimensional hypersphere of which the $j$-dimensional hypersphere is a section. Thus, by induction on $j$, an $n$th order degeneracy must occur whenever a lower order degeneracy occurs. Therefore any method that detects all $n$th order degeneracies will also find any lower order degeneracies.

In theory, the possibility of $j + 2$ points, for $1 \leqslant j \leqslant n$, from a stochastic distribution occurring on the surface of an $j$-dimensional hypersphere has probability almost surely zero. There is, however, a small but finite probability for this situation in digital computation because of numerical truncation and it may occur for any configuration, not merely squares or rectangles as Boots (1974, p. 26) suggests.

For geometric rigour, however, five points that may appear to lie on the surface of a 3-dimensional sphere must be partitioned into 4-tuples. This will produce either two or three 4-tuples. For an illustration, consider the five points A, B, C, D and E. Let A and E hold relative pole positions and B, C and D be equatorial in any approximate sense. Then either the two 4-tuples (A, B, C and D) and (B, C, D and E) or the three 4-tuples (A, B, C and E), (A, C, D and E) and (A, B, D and E) will lie on empty circumspheres.

In the general case, if a newly introduced nucleus lies ambiguously on the $(n - 1)$-dimensional surface of a circumsphere defined by some other $n + 1$ nuclei, then there are two possible outcomes. Either the additional nucleus is outside the circumsphere and forms from 1 to $n - 1$ additional $n$-simplices with some $(n - 1)$-dimensional faces of the first $n$-simplex, or the new nucleus is inside the circumsphere. Then the old $n$-simplex must be rejected and from 2 to $n$ new $n$-simplices are formed with some $(n - 1)$-dimensional faces of the old simplex. The number of new $n$-simplices to be formed in either case depends on the relative orientation of the new nucleus with respect to the old $n$-simplex. It is essentially a question of how many $(n - 1)$-dimensional faces of the old $n$-simplex that the new nucleus can see or not see, respectively.

The problem of such apparent degenerate forms occurs whenever the distance from a new nucleus to the surface of a circumsphere is within the expected accumulated truncation error bounds. Note that we may not simply make an arbitrary choice as Green and Sibson (1978, p. 171), may do in 2-space, because this would not resolve any lower order degeneracies that may exist. For regular arrays of nuclei, the Voronoi vertices may be truly degenerate and then the Delaunay tessellation is non-unique.

### 2.3 *Computational Aspects*

There are two related considerations to computation in this approach.

(i) The Voronoi vertex, or simplicial circumcentre of an $(n + 1)$-tuple of nuclei, is the intersection of the ${}_{n+1}C_2$ $(n - 1)$-dimensional hyperplanes that perpendicularly bisect the line segment joining any two nuclei in the $(n + 1)$-tuple. So we must solve a set of simultaneous linear equations. This is the step marked B in Fig. 1. If these hyperplanes define a small included angle with each other the set of equations may be ill-conditioned.

(ii) Errors due to truncation in the operations of the search for intersection of hyperspheres may lead to incorrect rejection or acceptance of a given $(n + 1)$-tuple while not making a compensating error in its immediate environment. Such is the case when the decisions reached at steps A and A′ in Fig. 1 are contradictory. This produces structural inconsistencies (see Green and Sibson, 1978, p. 171) and the aggregate may be observed to lose its disjoint property.

These problems may be avoided by rejecting data beyond the software resolution and by evading the need for the search at A′. That search is not necessary if we save all the $(n - 1)$-faces of the deleted $n$-simplices and, ignoring pairs, forming the new $n$-simplices with all singly occurring $(n - 1)$-faces only after all old circumspheres have been checked for intersection. This is justified by observing that all of the $n$-simplices whose circumspheres contain the new nucleus must form a simplicial $n$-polytope, with all the old nuclei being on its boundary. Only the $(n - 1)$-faces that are on this boundary will form new $n$-simplices with the new nucleus which is in the interior of the simplicial $n$-polytope. All of the other $(n - 1)$-faces on the old $n$-simplices are interior to the simplicial $n$-polytope so would form $n$-simplices, with the new nucleus, that overlap those formed with the boundary $(n - 1)$-faces. We may be sure that the new hyperspheres thus formed will not intersect any nuclei outside the simplicial $n$-polytope because the new nucleus would have intersected any old hypersphere involving those nuclei.

By these means the correct $(n + 1)$-tuples appear directly as a result of the search for intersected circumspheres without requiring the search at A′ in Fig. 1 or the computation of circumcentres for non-valid $(n + 1)$-tuples. However, a point that occurs outside the existing simplicial complex, even though it intersects some circumspheres, will not compute correctly because it is not interior to the simplicial $n$-polytope.

### 2.4 *Boundary Effect*

Another problem, commonly known as the boundary effect, occurs on the Voronoi tessellation when, for instance, it is used as a model of polycrystalline materials. It is due to the altered ratios amongst the vertices, edges, faces and higher dimensional elements that appear on the surface of an aggregate relative to its interior. Also, because of the decreased density of nuclei at the boundary of a finite set, there is a shape distortion in the boundary polytopes that would not appear in the infinite array. This results in a statistical bias when boundary polytopes are included in a histogram. The solution almost always seems to be based on the principle of ignoring all information from outside an arbitrary limit or window, but alternatively this may be done by flagging domains that are incident to the boundary of the aggregate.

## 3. Efficiency

Several means of improving the efficiency of this approach may now be applied.

(i) If the chronological input order of the nuclei is maintained in each $(n + 1)$-tuple of indices, then the search for duplicate sets reduces to the search for duplicate ordered sets. This search is necessary for finding doubly occurring $n$-tuples and to establish other particular values such as the number of edges.

(ii) Also, since the radii of the circumspheres is used only in comparison, the squares of these values may be used to save the time of root extraction.

(iii) Then the searches for intersected circumspheres is thus also made more efficient by subtracting squared components of the nucleus to circumcentre distance from the squared radius. When the squared radius goes negative we know that the new nucleus is not within the circumsphere and this will often occur before all squared components have been subtracted.

(iv) To prevent storage and search times growing non-polynomially, we can progressively eliminate some of the boundary data if all the remaining input will still be interior. Then we can overwrite those data points and their associated $(n + 1)$-tuples for which computing is complete while still maintaining the correct environment to compute

new relationships. One way of achieving this is by a pre-processing which orders the data with respect to the first component. Then if the squared radius goes negative after the first subtraction, that particular circumsphere cannot be intersected by any of the remaining data points so computing is complete for that $(n + 1)$-tuple.

## 4. The Algorithm

### 4.1 *Theoretical Framework*

The discussion above may be condensed to form a theoretical basis for the algorithm. Given any stochastic array of data points, there exists a unique Delaunay tessellation of those points such that the $j$-dimensional circumsphere coincident to the vertices of any $j$-simplex in the tessellation has no data point in its interior.

A new data point may be introduced to any Delaunay tessellation by observing which circumspheres are intersected. All of the old $n$-simplices whose circumspheres contain the new data point form a simplicial $n$-polytope, with all of the old data points, (of the intersected circumspheres), lying on its $(n - 1)$-dimensional boundary. These old $n$-simplices will be replaced. The new point is interior to the simplicial $n$-polytope and forms new $n$-simplices with each of the boundary $(n - 1)$-simplices. The circumspheres of these new $n$-simplices will not intersect any of the other data points because these other points all lie on circumspheres not intersected by the new point. The tessellation remains a Delaunay tessellation because all circumspheres remain empty. The order in which the points are introduced does not affect the final configuration because that depends only on the number and position of the data points in the complete set.

If an arbitrary point fails to intersect any of the circumspheres in a Delaunay tessellation then the existing tessellation is not altered. However, new Delaunay $n$-simplices now exist between the arbitrary point and some of the $(n - 1)$-simplices on the boundary of the old structure. If $n + 1$ arbitrary points are added to the data set in this manner, the Delaunay $n$-simplices that tessellate the data set can be completely bounded by Delaunay $n$-simplices that include at least one of the arbitrary points.

Starting with $n + 1$ arbitrary points, known to form an $n$-simplex whose convex hull contains the whole data set, the structure of the Delaunay tessellation may be computed by adding the data points in an 'advancing front' sequence. The tessellation remains a Delaunay tessellation at all times. The Delaunay $n$-simplices behind the advancing front are in their final configuration while those ahead of the front are subject to alteration.

Any data point found to be in non-general position is rejected because it leads to a non-unique tessellation. A point is in non-general position when the resolution of the software is not sufficient to determine whether or not an intersection with a given circumsphere has actually occurred.

The Delaunay $n$-simplices of the data set are distinguishable from the bounding Delaunay $n$-simplices because the latter have at least one of the arbitrary points as a vertex. The Voronoi polytopes associated with the interior points of the data set are distinguishable from those associated with data points on the boundary because the latter share an $n$-simplex with at least one arbitrary point.

To implement this theory, the algorithm manipulates and maintains a list of $(n + 1)$-tuples of indices to the nuclei or data points. These $(n + 1)$-tuples represent Delaunay $n$-simplices. Each $(n + 1)$-tuple of indices is associated with an $(n + 1)$-tuple of real values being the coordinates of the simplicial circumcentre and the square of the circumsphere radius.

### 4.2 *Initialisation*

The algorithm assumes an initial aggregate of $n$-simplices, where each $n$-simplex has an empty circumsphere. These implementations used the vertices of an orthogonal $n$-simplex. Any $n$-simplex would do as well, but it must be large enough to enclose the complete range of input points. The vertices of this simplex serve as boundary indicators since any $(n + 1)$-tuple that includes one or more of these initial vertices must lie on the boundary.

### 4.3 *Operational Steps*

For each data point to be introduced, sequentially search the list of $(n + 1)$-tuples to find all circumspheres that contain the new point. This point must fall within some of the circumspheres for they enclose the $n$-simplices which are space-filling. For each such intersected circumsphere, flag the $(n + 1)$-tuple to indicate deletion of the associated $n$-simplex. The search is made by subtracting squared components of the new data point to circumcentre distance from the squared radius. Since the data are ordered by the first component, a negative result to the first subtraction indicates a completed $(n + 1)$-tuple.

Then each $(n - 1)$-face of the $n$-simplices whose circumspheres have been intersected, that is, each possible $n$-tuple of nuclei from each $(n + 1)$-tuple, is saved on a temporary list. If any $(n - 1)$-face is found to occur twice, both occurrences are dropped from the list since this face is shared by two adjacent $n$-simplices. This means that it is interior to the simplicial $n$-polytope formed by all $n$-simplices whose circumspheres have been intersected by the new nucleus. If the new nucleus falls within the limits of the expected accumulated truncation error of any circumsphere, the temporary list and the new nucleus are abandoned but alternate treatment such as recalculating the circumcentre by a higher precision routine could be used here. If such an ambiguity does not appear, new $n$-simplices are then formed with each of these singly occurring $(n - 1)$-faces and their circumcentres are calculated. By maintaining a count of the $n$-simplices incident to each nucleus and decrementing for each completed $n$-simplex, completed nuclei will show a zero count. Completed nuclei and $(n + 1)$-tuples are replaced as their indices appear on first-in–first-out stacks.

That completes the operations for insertion of a new nucleus into the tessellation.

### 4.4 *Storage and Execution Time*

Since the $N$ data points are ordered by their first component, we effectively compute a local $(n - 1)$-dimensional section through the data. If the data space is equidimensional, storage will be proportional to $N^{(n-1)/n}$. Then execution time for the $N$ data points will be better than proportional to $N^{(2n-1)/n}$.

### 4.5 *Voronoi Interpretation*

To convert these completed $(n + 1)$-tuples to particular Voronoi polytopes, a list of indices of uncompleted nuclei with a count of incident $n$-simplices is maintained. If any nucleus is found to share an $n$-simplex with any of the initialisation vertices, it is dropped from the list. As each $(n + 1)$-tuple is completed, the $n$-simplex count for each of its $n + 1$ nuclei is decremented and information such as circumcentre coordinates is recorded. Each polytope is complete when the $n$-simplex count for its nucleus has been decremented to zero.

### 4.6 *Computing in Linear Time*

The remarks above apply to the determination of the structure of a given set of data points as interpreted by the Delaunay–Voronoi tessellation. For investigations of the Delaunay–Voronoi structure itself, the algorithm may be adapted to run in linear time.

In this case, random points are generated within the unit $n$-cube. The boundary of that $n$-cube is considered to be a 'window' about the data. The initial $n$-simplex completely encloses the unit $n$-cube. Then as in Section 4.2, with each

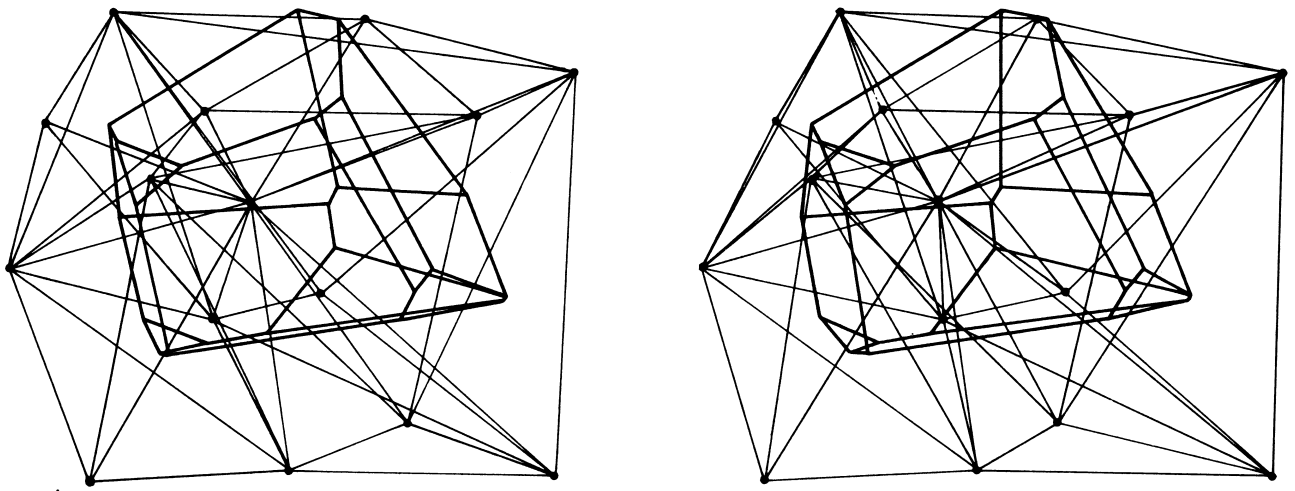

**Fig. 2 Naked eye inversion stereopsis diagram of Delaunay tetrahedra surrounding a nucleus (fine lines), and the associated Voronoi polyhedron (bold lines). Hold at easy arm's length and allow the eyes to cross slightly. When three images appear, observe the central one.**

newly generated point, we search for intersection with existing hyperspheres, find the simplicial $n$-polytope and form the new $(n + 1)$-tuples.

Once a preset number of nuclei, say $k$, have been triangulated within the bounds of the window frame, and before each subsequent new point is introduced, the size of the window is reduced by a factor of magnitude inversely proportional to $k$. This reduces the range in which the new points may appear and may bring some earlier generated points outside of the window frame. These are then flagged as external points. Then any $n$-simplex having at least one of its nuclei flagged is known to lie in the boundary zone. Any new nucleus that fell within the circumsphere of such an $n$-simplex must have a portion of its Voronoi domain on the boundary, so it is not recorded. As soon as an $(n + 1)$-tuple has its circumsphere completely outside the window, it cannot be intersected and is flagged as external also. External nuclei and $(n + 1)$-tuples are replaced as their indices appear on first-in–first-out stacks.

In this manner the type and number of relationships of a newly generated nucleus in the interior of the aggregate may be observed repeatedly without introducing surface bias. Then the execution time will be $O(kN)$.

## 5. Testing

Several methods were found to test the algorithm. In the 2-dimensional case, the increase in the number of triangles per added nucleus must be exactly two. This is because for any cluster or aggregate of Delaunay triangles, with all vertices on the perimeter, another triangle may be appended with one face only being interior and thus yields an increase in boundary edges of one. Therefore there are always two more edges than triangles. If a new nucleus intersects all the circumcircles of these triangles, it must form new triangles with each of the perimeter edges of the initial cluster. The net gain is two triangles.

In the 3-dimensional case, the number of tetrahedra incident to any nucleus is always an even number. This is because another tetrahedron may be appended to the aggregate of Delaunay tetrahedra, with all vertices being on the exterior, with either one or two faces being interior and thus alter the number of faces on the boundary by two or zero. So the number of boundary faces is at least four and always remains even.

In the 4-dimensional case, the increase in the number of 4-simplices when a new point is introduced is always an even number. That is, the difference between the number of 4-simplices in the aggregate and the number of 3-simplices on its boundary is even. By similar reasoning as was used above, this is because the appending of another 4-simplex to an aggregate of 4-simplices, with all vertices on the boundary, must change the number of 3-dimensional faces on that boundary by an odd number. Note that a 4-simplex has five 3-dimensional faces. One, two or three of these may be internal to the aggregate and thus change the number on the boundary by 3, 1 or $-1$ respectively. Four of the 3-dimensional faces could not be internal since no vertex is internal. Thus the difference between the number of 4-simplices in such an aggregate and the number of 3-dimensional faces on its boundary must always be an even number. These values were therefore scanned for violations and none were found.

All $(n - 1)$-faces of the Delaunay tessellations were listed to expose more or less than two occurrences. That would be the case if some $n$-simplices overlapped or were missing.

Naked eye inversion stereopsis, Speakman (1978, p. 827), has been used to display the 3-dimensional Delaunay configuration about a particular nucleus and its associated Voronoi polyhedron (see **Fig. 2**), as calculated by the algorithm. The use of naked eye inversion stereopsis is simply that of allowing the binocular lines of sight to intersect at the normal position for reading, while holding the page beyond that point so that the computed inversion stereo images will blend to form a 3-dimensional image between the eyes and the page. This display function was useful for debugging.

## 6. Implementations

Versions of 2-, 3- and 4-dimensions of the polynomial time form of this algorithm, with Voronoi interpretation, were implemented on the Control Data Cyber 170-730. Each required less than 200 FORTRAN statements, including I/O. The 2- and 3-dimensional versions were run for 3000 input points and the 4-dimensional version was run for 600 points. Execution time was found to increase more slowly than $N^{(2n-1)/n}$ and storage increased as $N^{(n-1)/n}$. They ran at a rate of approximately 0·01, 0·25 and 8·5 s per interior nucleus, respectively. Additionally, these three programs were stripped of I/O and Voronoi conversion code. The Delaunay tessellations were then computed on this machine at a rate of 107, 5, and 0·35 nuclei $s^{-1}$.

Additionally, 2-, 3- and 4-dimensional versions of the linear time algorithm were also run on this machine and generated interior Voronoi polytopes at an approximate rate of 0·05, 1·8 and 5 s per polytope.

**Acknowledgements**

Thanks are due to Dr D. Herbison-Evans of the Basser Department of Computer Science, University of Sydney, for reading the manuscript and for several helpful suggestions. The initial stages of this work were supported by a University of Sydney Postgraduate Research Studentship. Computing facilities were provided by the University of Sydney Computer Centre.

**References**

BOOTS, B. N. (1974). Delaunay triangles: an alternative approach to point pattern analysis, *Proceedings of the Association of American Geographers*, Vol. 6, pp. 26–29.

CRAIN, I. K. (1972). Monte-Carlo simulation of the random Voronoi polygons: preliminary results, *Search*, Vol. 3, pp. 220–221.

GREEN, P. J. and SIBSON, R. (1978). Computing Dirichlet tessellations in the plane, *The Computer Journal*, Vol. 21, pp. 168–173.

MILES, R. E. (1970). On the homogeneous planar Poisson point process, *Mathematical Biosciences*, Vol. 6, pp. 85–127.

PREPARATA, F. P. and MULLER, D. E. (1979). Finding the intersection of $n$ half-spaces in time $0(n \log n)$, *Theoretical Computer Science*, Vol. 8, pp. 45–55.

RHYNSBURGER, D. (1973). Analytic delineation of Thiesson polygons, *Geographical Analysis*, Vol. 5, pp. 133–144.

ROGERS, C. A. (1964). *Packing and Covering*, Cambridge Mathematical tracts No. 54, Cambridge University Press.

SIBSON, R. (1978). Locally equiangular triangulations, *The Computer Journal*, Vol. 21, pp. 243–245.

SOMMERVILLE, D. M. Y. (1927). The relations connecting the angle-sums and volume of a polytope in space of $n$ dimensions, *Proceedings of the Royal Society of London Series A*, Vol. 115, pp. 103–119.

SPEAKMAN, J. C. (1978). Visualising molecules without really trying your eyes, *New Scientist*, Vol. 78, p. 827.

WATSON, D. F. and SMITH, F. G. (1975). A computer simulation and study of grain shape, *Computers and Geosciences*, Vol. 1, pp. 109–111.

# Book reviews

*Cluster Analysis Algorithms* by H. Spath, 1980; 226 pages. (*Ellis Horwood*, £15·50)

This is a translation of a book which first appeared in German in 1975 with a slightly revised edition in 1977. It is attractively packaged and a quick flip through reveals plenty of diagrams and FORTRAN algorithms. This 'feet on the ground' impression unfortunately does not survive closer inspection. The author is a professor of mathematics and expects his readers to know about $L_p$ norms, mappings, matrices and the like. Distance measures are introduced as formulae with little verbal back up and much more space is devoted to proving that certain measures are indeed metrics than to discussing which measures are best to use in practice. At one point we find the amazing guidance "Those [methods of scaling data] for which the clusters are particularly easy to interpret are regarded as meaningful".

The algorithms are at least clearly printed, but they are inadequately commented, insufficiently protected, use the numerically inaccurate way of calculating corrected sums of squares and are (once, at least, on p. 54) wrong. The hierarchical methods do not allow output of dendrograms. The data sets used to illustrate the various techniques are all, unfortunately, German. This is mildly inconvenient when the positions of German towns are being clustered, but when the reader is left to interpret the results of a clustering of Bavarian postal zones, this reader for one felt somewhat irritated.

The bibliography covers seven pages and includes most key references. There is, however, no mention of Wishart's CLUSTAN package, GENSTAT or the BMD programs. The only references later than 1975 are to the author's own publications.

All in all, then, this book adds very little to the literature and has little advantage over existing ones. My own favourite is *Clustering Algorithms* by J. Hartigan.

P. R. FREEMAN (Leicester)

*Programming via Pascal* by J. S. Rohl and H. J. Barrett, 1980; 327 pages. (*Cambridge University Press*, £12·50, £5·95 paper)

This book is developed from a series of lectures and is organised so that each of the 24 chapters covers enough material for one lecture, hence students using the book should find topics presented in nice bite-size chunks. The whole of the language is mentioned, although I found some of the presentation somewhat cursory. In general, language features are introduced by example of their use with syntax diagrams steadily being built up as the subject unfolds. At the end of each chapter exercises are given which not only involve writing programs, but also ask the reader to do the very important task of amending programs too. Most of the exercises have a numerical flavour but a payroll program is also developed.

Despite the title of the book, the authors do not present much in the area of program design, this they claim (and I agree) is probably best done by discussion with an instructor. However, it does limit the utility of the book for those doing self-study of the subject. There are, however, two good but short chapters on testing and efficiency. The book also misses out on the wider view of programming, only a small mention is made of data structure design and documentation is virtually ignored.

In the end, whether you like the book or not will depend on how you see programming, in my view, two vital topics which should be introduced very early in a course are records and procedures, here they are not introduced until Chapters 15 and 16, respectively. Similarly, arrays which I do not consider too important at the early stages are introduced in Chapter 8 whilst files are left until Chapter 22.

In summary, I consider the book could make useful back-up reading to first year undergraduate Pascal courses but whether it is recommended will depend (as always) on the teacher's prejudices. In my case I will stick on Laurence Atkinson's book.

D. SIMPSON (Sheffield)

*Online Searching: An Introduction* by W. M. Henry, J. A. Leigh, L. A. Tedd and P. W. Williams, 1980; 209 pages. (*Butterworth*, £12·00)

Those whose primary concern is computer science or programming will find little of direct interest here. They may, however, be fascinated by a 76 page summary of commercially asscessible bibliographic data bases which includes details of size, search cost and updating frequency of most of the collections, together with summaries of the command languages available for searching these collections on-line.

For the information officer or librarian who has not yet come to grips with automated information retrieval, or who is dissatisfied with the limited facilities he has available, this is an outstanding general introduction. It is characterised by breadth of treatment, clarity and conciseness, and obviously reflects extensive practical experience. There is just enough detail to give the reader a genuine feel for the practicalities of on-line access to bibliographic data and the environment in which it is used. Though future developments are discussed, the reader's feet are always comfortably on the ground; despite the rate of development and the rate of inflation, the authors have wisely chosen to use examples from currently operational systems and to quote financial costs. Books of this kind often outlast intended 'evergreens' which, by generalisation and prognostication, obscure rather than expound the state of the art. Regrettably, the price may put this fine scene-setting book out of the reach of most information science and librarianship students.

J. INGLIS (London)