A. Appel
P. M. Will

Determining the Three-dimensional Convex Hull of a Polyhedron

Abstract: A method is presented for determining the three-dimensional convex hull of a real object that is approximated in computer storage by a polyhedron. Essentially, this technique tests all point pairs of the polyhedron for convex edges of the convex hull and then assembles the edges into the polygonal boundaries of each of the faces of the convex hull. Various techniques for optimizing this process are discussed. A computer program has been written, and typical output shapes are illustrated. Finding the three-dimensional convex hull is approximately the same computer burden as eliminating hidden lines.

Introduction

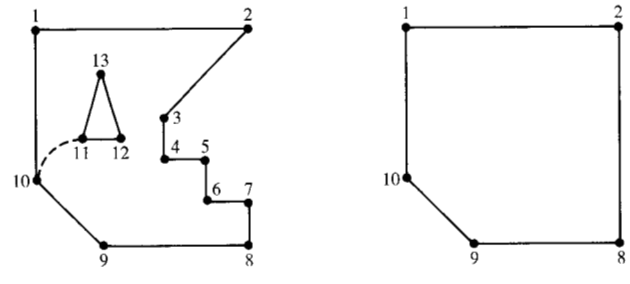
The determination of convex hulls and their properties has been of continuous interest in linear programming and pattern recognition [1–6]. Freeman has discussed the use of the two-dimensional convex hull for the stock cutting or allocation problem [7]; Balinski has described a method for finding all vertices of an n-dimensional convex polyhedral set defined by a system of linear inequalities [8]; and Nagy and Tuong have normalized two-dimensional data by transforming the convex hull of a pattern into a square [9, 10]. This paper describes a method for determining the three-dimensional convex hull of an object.

Two-dimensional solutions

A body is defined as being convex if a line joining any two points on the n-dimensional figure is entirely contained within the body. One can consider the polygon to be a set of edge points, and the convex polygon can be obtained from operations on this set; but it is more convenient, as in computer graphics, to consider the sets to be ordered lists. When the order of points on a polygon is given as a list, the lines on the polygon are drawn between successive points in this list. A convex point in a two-dimensional list can be defined as the intersection of two lines that form an internal angle less than 180°. A nonconvex point has an internal angle equal to or greater than 180°.

A polygon with an interesting boundary and its two-dimensional convex hull are shown in Fig. 1. A two-dimensional convex hull is a polygon formed only of convex points. Points on the original polygon list can be inspected to delete all nonconvex points. The new list must then be tested again for convex points since the deletion of some points may create new nonconvex points. The iterative nature of this procedure is illustrated in Fig. 2.

Figure 1 Typical polygon (a) and its convex hull (b).



The identification of the convex hull of unordered two-dimensional points has been described by Graham [11] and Jarvis [12]. Graham's approach is to order the original data set on the basis of polar angles about an interior point and then use an angle measurement scheme similar to that shown in Fig. 2. Jarvis contends that this ordering procedure is itself expensive in terms of comparison tests and that it is not necessary. With reference to Fig. 3, Jarvis's method is as follows:

1. Find a point definitely outside the point set. This is the initial origin.

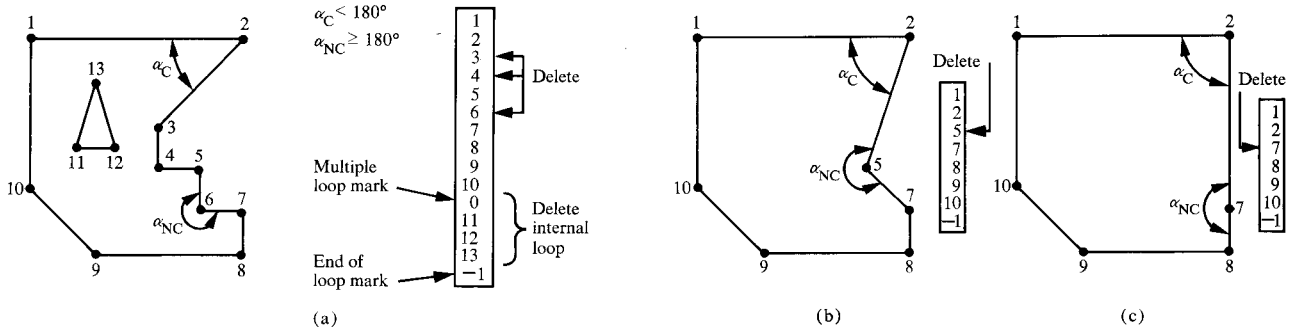


Figure 2 Derivation of convex hull shown in Fig. 1. Points 11, 12, and 13 can be deleted immediately from original polygon shown at (a) with the original point list. Polygon before second pass (b) has had points 3, 4, and 6 removed. Polygon before third pass (c) has also had point 5 removed.

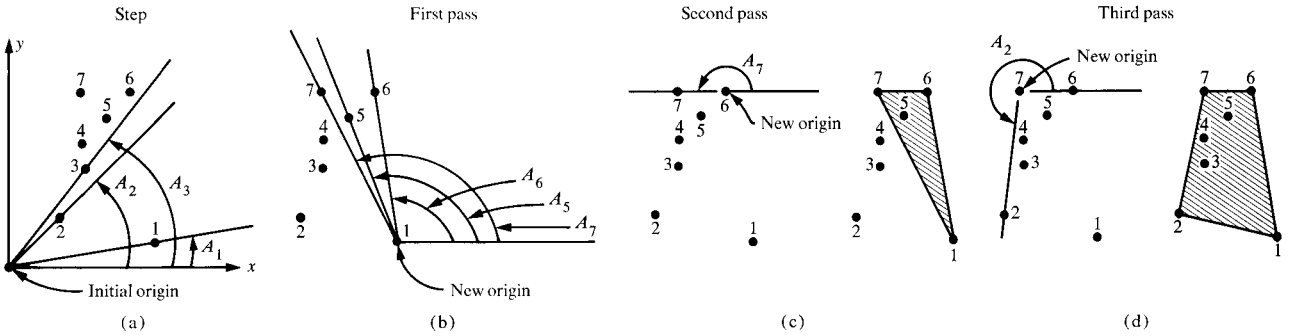


Figure 3 Jarvis algorithm for two-dimensional convex hulls. At (a) the minimum A_i is A_1, so point 1 is the first point on the convex hull. Using the new origin at (b), A_6 is the minimum A_i, so that point 6 is the second point on the convex hull. With point 6 as the new origin at (c), A_7 is the minimum A_i, and point 7 is the third point on the convex hull. Because point 5 falls within the area described by points 1, 6, and 7, it need not be considered any longer. At (d), with point 7 as the new origin, the minimum A_i is A_2, and point 2 is the fourth point on the convex hull. Now points 3, 4, and 5, which fall in the area of points 1, 6, 7, and 2, can be ignored, and the convex hull is complete.

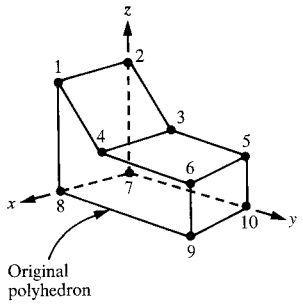
2. For each point i in the set, calculate the polar angle A_i between a horizontal line through the origin and the line connecting the origin and the point i.
3. Find a point j for which A_j is the minimum of all angles A_i. Point j is to be listed as a point on the convex hull.
4. Make point j the new origin, and repeat steps 2 and 3. Each time a convex hull point is found, relocate the origin for calculating A_i. Delete points from consideration if: a) they have already been identified as being on the convex hull, or b) they lie in the region so far determined to be included in the convex hull.

By deleting points from consideration as the convex hull grows, the Jarvis algorithm finds new convex hull points with increasing speed. The important observation in Jarvis's work is that ordering the point set is not necessary and may even be misleading.

Requirements of a three-dimensional solution

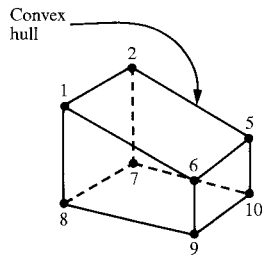
Several ways exist by which a convex hull may be described, and the application of the convex hull prescribes the description. Generally for two-dimensional work, such as pattern fitting, an ordered list of points is sufficient, but for picture scanning a region boundary such as a line list or a parametric curve may be necessary. The actual convex hull may be difficult to work with, so a rectangle [7] or a circle may be used as an approximation.

We consider objects that are generally encoded as polyhedral approximations [13, 14, 15]. Such polyhedra are usually specified by a vertex list and a topological map such as shown in Fig. 4. A vertex list and topological map suitably describe the convex hull. Therefore we need to find which of the original set of vertex points on the polyhedron description are vertex points of the convex hull, and we need to know how these points can be



Vertex list

Point	x	y	z
1	1	0	2
2	0	0	2
3	0	1	1
4	1	1	1
5	0	2	1
6	1	2	1
7	0	0	0
8	1	0	0
9	1	2	0
10	0	2	0

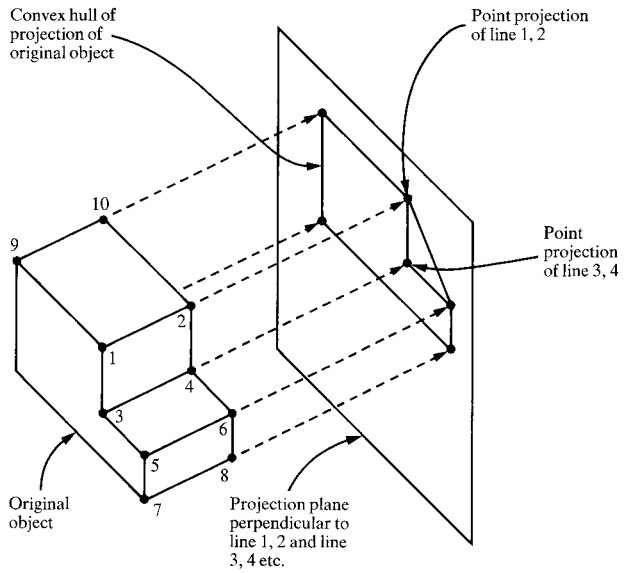


Topological map

Surface	Points
I	1,2,3,4
II	3,4,6,5
III	5,6,10,9
IV	1,2,7,8
V	7,8,9,10
VI	1,4,6,9,8
VII	2,3,5,10,7

Figure 4 Typical original data and required specification of a convex hull. Points 3 and 4 have been deleted from the vertex list, and surfaces I and II have been deleted from the topological map. Surface VIII with connectivity 1, 2, 5, 6 must be added, and surfaces VI and VII must be respecified.

Figure 5 Determination of convex edges from a projection plane perpendicular to the edge. The point projection of lines (1, 2), (5, 6), (7, 8), and (9, 10) are convex points, hence these lines are convex edges. The point projection of line (3,4) is not a convex point, and this line is not a convex edge.



connected together to form the boundaries of the plane surfaces of the convex hull. In the two-dimensional convex hull problem, points of the original data set had to be eliminated and lines that were not part of the original polygon had to be found. In the three-dimensional problem, points have to be eliminated, and new lines and planes must be found.

Properties of convex polyhedra

Several properties of a convex polyhedron can serve as a basis for determining the convex hull of points in three-dimensional space. For example:

1. Any point inside the convex polyhedron is on the material side of all planes of the polyhedron.
2. Each convex point of the convex hull is a vertex of a convex cone of the convex polyhedron.
3. Each convex edge of the convex hull is the intersection of exactly two convex planes and has an internal dihedral angle less than 180°.
4. The boundary of any plane surface is a convex polygon.
5. Any projection of a convex polyhedron is a convex polygon.
6. In any projection of a convex polyhedron, the edges that produce the outline are a closed curve in three-dimensional space.
7. A convex polyhedron encloses the given points in space with minimum surface area.

Doubtless, this list of properties can be extended. Properties 1 and 2 are most frequently used in linear programming. Properties 1 and 6 can be used for hidden line elimination [16]. Properties 3 and 4 are the only characteristics that make it possible to delete some given points on the basis of local measurements. All of the other properties are characteristics of the entire object and are expensive to determine.

Algorithm for determining the three-dimensional convex hull

The algorithm to be described attempts to find the edges of the convex hull. Identifying the edges of the three-dimensional convex hull seems to be most appealing for two reasons: 1) If the edges are known, vertex points on the convex hull can be identified, and with small effort the topological map can be constructed; 2) the probability of wasteful testing for convex edges is less than the probability of wasteful testing for convex planes. With regard to this last reason, the number of two-combinations of N things is usually smaller than the number of three-combinations of N things:

$$C(N, r) = (N!)/(r!)(N - r)!$$

$$C(6, 2) = 15, C(6, 3) = 20.$$

The basic algorithm is as follows:

1. Test all possible pair combinations of the vertex list to identify convex edges. The test for convex edges is based on the property that if all points are projected parallel to this edge, the edge projects as a convex point on the polygon image. See Fig. 5. Convex edges are not collinear.
2. Form loops in three-dimensional space of convex edges. Each edge must be used twice, and all the edges used in any particular loop must lie on the same convex plane. A convex plane has the property that all points of the object in three-dimensional space lie in the plane or are only on one side of this plane (property 1).

For objects that are known a priori to be "well behaved," a preprocessing step may be added. (Well behaved is defined later). This preprocessing deletes from the original vertex list any point that lies on an edge that is an inside corner. Such edges and any points on them cannot contribute to the description of the convex hull (properties 3 and 4). Identifying inside corners is a common capability of three-dimensional modeling programs [17]. See Fig. 6.

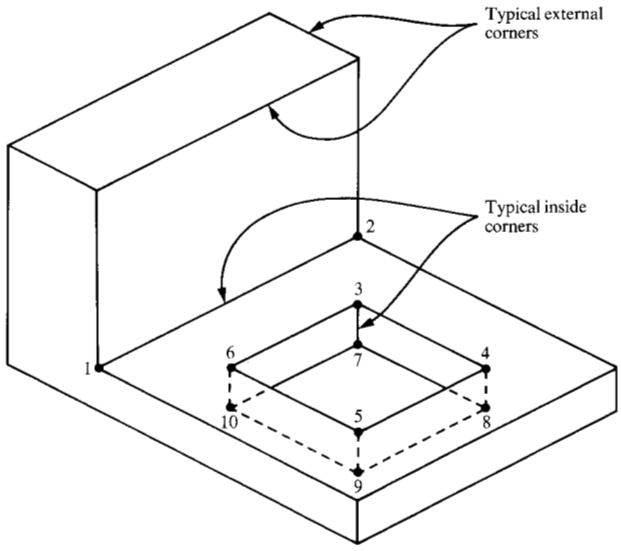


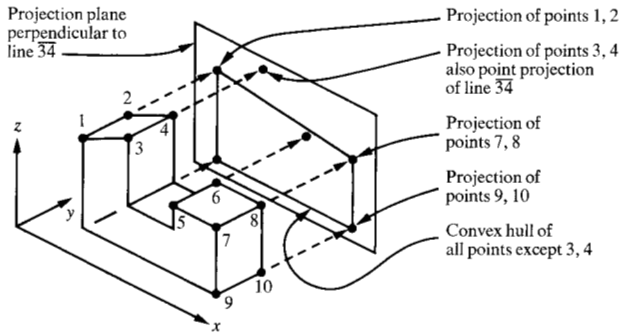
Figure 6 Points 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10 are obviously not convex hull vertex points because they lie on inside corners.

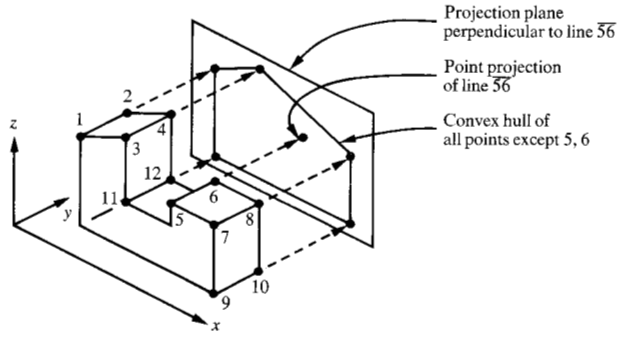
Determination of convex hull edges

The problem now is to find pairs of points on the vertex list that are end points of lines in three-dimensional space that are, in turn, edges of the three-dimensional convex hull. The key characteristic of a convex hull edge is shown in Fig. 7: If the entire set of points on the object is projected onto a plane perpendicular to a convex edge, the convex edge projects as a point that is a convex point on the projection.

An alternative, and what has proved to be a more useful, definition of a convex edge is also shown in Fig. 7. The convex edge is seen as projecting as a point outside of the two-dimensional convex hull of all the other points projected onto the projection plane. Step 1 is therefore: Choose pairs of vertex points, project orthogonally to them, and test for containment. A test is then needed for the containment of a point in a polygon.

Figure 7 A characteristic of convex edges. At top line (3, 4) is determined to be a convex edge, since its point projection lies outside of the convex hull of the projections of all the other points on the object. At bottom line (5, 6) is not a convex edge, since its point projection lies inside the convex hull of the projection of all the other points on the object. Line (11, 12) did not have to be projected because it is an inside corner, and points 11 and 12 need never be projected.





Polygon star test

A useful property of three points on a plane is the sense of *implied vorticity* of these points [18]. In the matrix equation for area of a triangle:

$$AREA = \tfrac{1}{2}\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}.$$

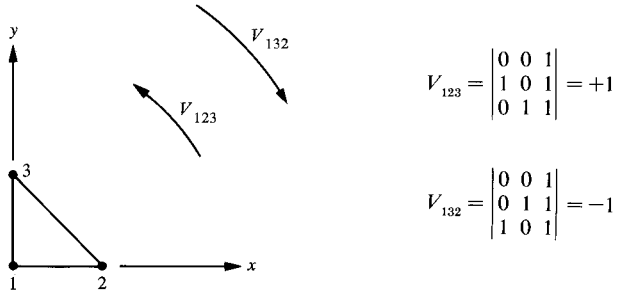


Figure 8 Sense of implied vorticity and the recognition of direction of rotation.

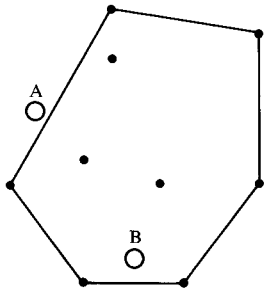


Figure 9 Placement of two points relative to the convex hull of a given set of points.

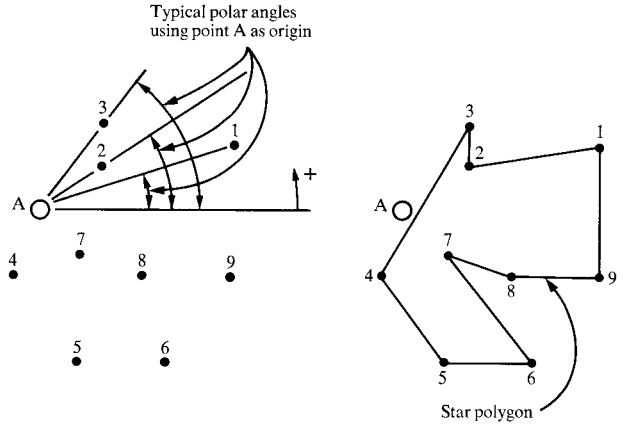


Figure 10 Generation of star polygon for point A.

The value of area can be positive or negative, depending on the order in which points are entered in the matrix; the area is zero if the three points are collinear or coincident. This property can be used to sense an implied rotation, as shown in Fig. 8. A counterclockwise rotation has a positive sense of vorticity, a clockwise rotation has a negative sense of vorticity, and no rotation has a sense of zero. A useful characteristic of implied vorticity is that any point inside a triangle has the same sense of vorticity relative to each of the lines of the triangle as the three vertex points of the triangle. The implied vorticity can be used to indicate when a point lies inside or outside of the convex hull of a set of points.

This test can be called a *star polygon test* because of the general character of the polygons encountered in this application, and its value is that it avoids having to find the two-dimensional convex hull of the set of projected points. With reference to Fig. 9, it is necessary to determine whether points A or B lie within the convex hull of the given points. A star polygon can be constructed for the given set of points if they are sorted on the basis of increasing polar angle, with the origin of polar angle measurement being the test point. Star polygon construction is shown for points A and B in Figs. 10 and 11, respectively. When the point under examination is outside the star polygon, it is also outside the convex hull of the star polygon. In Fig. 10, the sense of rotation of line (2, 3) relative to point A is counterclockwise, but the sense of rotation of line (3, 4) is clockwise. In Fig. 11, the sense of rotation of all lines is counterclockwise about point B. Obviously, any *change* in implied vorticity indicates that a point is outside of a star polygon. Figure 12 illustrates collinear possibilities that are detected when implied vorticity is zero. Possibility I or II can occur since the sorting of points is based only on polar angle. In instances I and II and similar collinearities, the point being tested is considered as being outside of the star polygon, whereas for possibility III the point is considered as being inside the star polygon.

In Fig. 13 an object is shown having several collinear points with the same projection on the orthographic projection plane. During the construction of star polygons, such occurrences are easily detected and points that form lines contained within other lines can be deleted from the vertex list immediately. For example, in Fig. 13, the interior points 2, 3, 6, and 7 should be deleted from the vertex list. The calculation of orthographic projections is an opportune time for detecting interior points since the distance from the orthographic projection plane can serve as a basis for sorting.

This step in the algorithm enumerates pairwise combinations of vertex points and tests for convexity. It is worth while considering techniques that quickly delete from the list of vertices all points that are not ends of a convex edge. Apart from a few specific tests discussed in detail later, this is an open question for research.

Finding surface loops

At this stage of processing, the three-dimensional convex hull is essentially known. All convex edges have

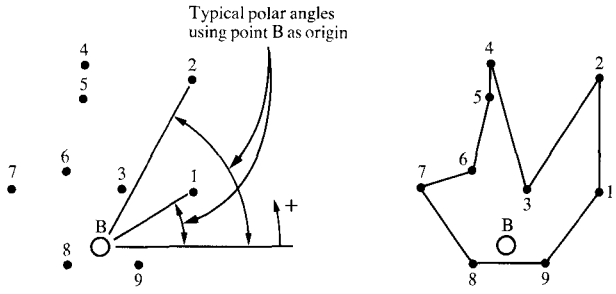


Figure 11 Generation of star polygon for point B.

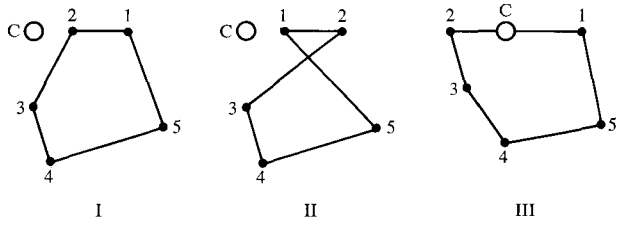


Figure 12 Collinear possibilities. Point C is the basis of construction of the star polygons. In the first two cases point C is considered to be outside the star polygon, and the edge that projects as point C is taken to be a convex edge. In the third case point C is taken to be inside the star polygon and is not treated as a convex edge projection.

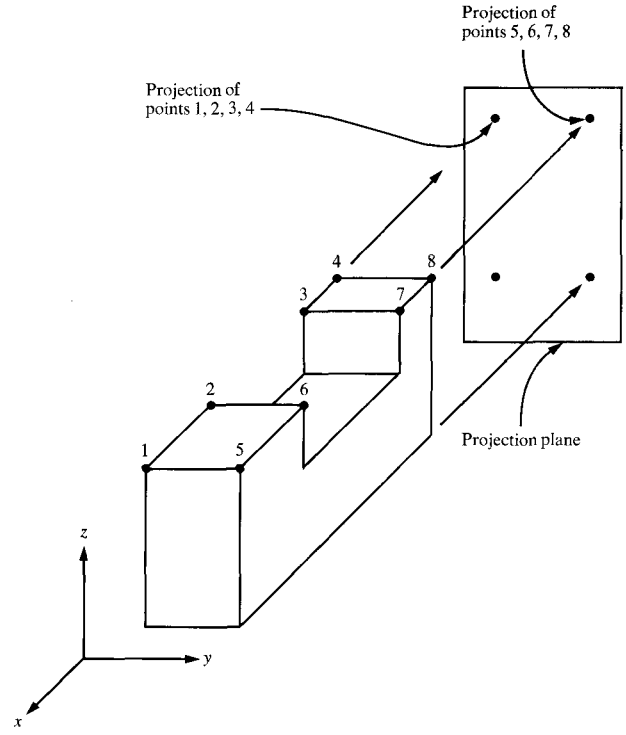


Figure 13 An object with coincident edge projections.

been found and listed, and the original list of vertex points has been reduced to only those points that have been found to be end points of a convex edge. A typical convex edge list is shown in Fig. 14. For some purposes, it may be necessary to find the polygonal boundary of each of the plane surfaces of the convex hull. Such boundaries are formed by closed loops of edges in three-dimensional space. The key property used in forming such loops is that each convex edge is used in exactly two loops, because an edge is the intersection of only two convex planes. A convex plane loop can be thought of as a list of edges where the first and last entries have a common end point and adjacent entries have a common end point. The plane surface bounded by a loop must also be a convex plane; that is, there are no points on one side of the plane.

The procedure for finding surface loops is basically to start a list of edges on a single convex plane by finding two edges that intersect and lie on a unique plane of the convex hull and then add coplanar connected edges to this list until the loop closes. A new surface loop list can now be started. A count of the number of times an edge is used is made and continually updated, and as soon as an edge is used twice it is no longer considered. When all edges have been used twice, all loops should have been found and this phase of processing stops.

The topological map can be derived from a loop list by listing the vertex points that are shared by adjacent edge entries. A more detailed description of loop finding is given in the appendix.

Preprocessing to speed execution

It is worthwhile applying preprocessing to reduce the number of projections and star polygon tests. The preprocessing algorithm alluded to previously involves the determination of inside corners, since they could not contribute to the convex hull of a well behaved object. The attractive feature of this idea is that the test is well known in computer graphics and is a comparatively direct and inexpensive method of reducing the original vertex list. There are instances where this procedure using an inside corner fails, as shown in Fig. 15. However, such instances are extremely rare, since points are almost always the intersection of three planes. If an object is known to have points with no more than three intersecting planes, then it may be considered to be well behaved and this preprocessor may be used. For well behaved objects, any point on an inside corner can be deleted from the original vertex list. For optimum reliability where the objects are very complex, points need not be eliminated unless they are the intersection of two inside corners. Another test for points that lie on an in-

Typical convex edge list

Line label	First point	Second point
L	LC1(L)	LC2(L)
1	1	2
2	1	3
3	1	4
4	2	3
5	2	6
6	3	5
7	4	5
8	4	6
9	5	6

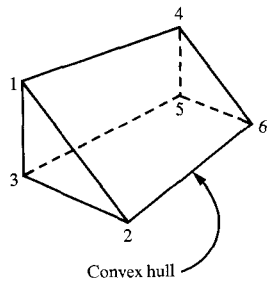


Figure 14 A convex edge list for a single convex polyhedron. The line label is L, and the ends of the line are LC1 and LC2 indexed by L.

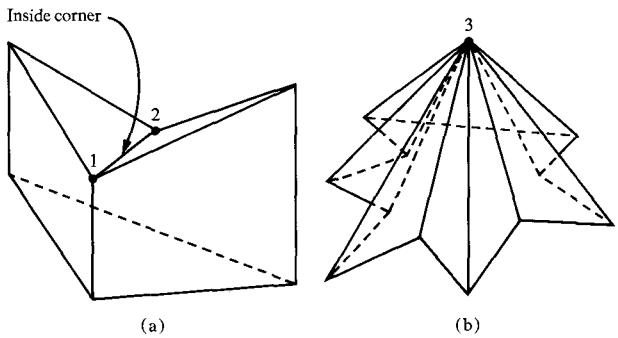
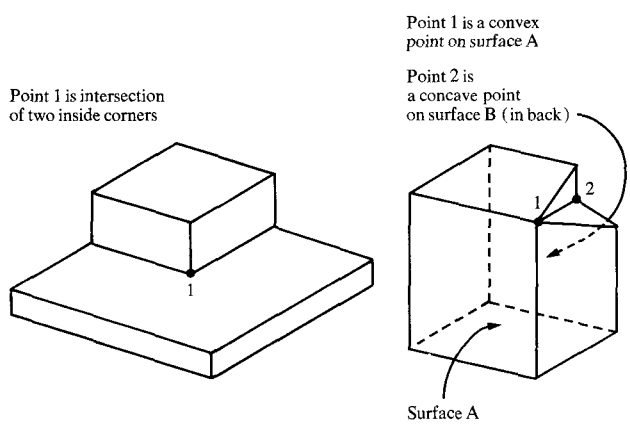


Figure 15 Instances where a point of an inside corner should not be deleted. At (a) point 2 can be deleted but point 1 should not be deleted. There are many inside corners at (b), and point 3 should not be deleted.

Figure 16 Two checking techniques for points that are detected as being on inside corners: 1) Any point that lies on the intersection of two inside corners and only one outside corner cannot be a convex hull vertex. 2) A point that is on an inside corner should also be a concave point on a polygon face in order to be deleted.



side edge is to determine whether the point is a concave point on any polygonal face of the object. Checking techniques are illustrated in Fig. 16 but are assumed to be unnecessary for most objects.

It may be argued that no point should be kept in the vertex list if it does not lie on a convex plane of the original polyhedron. To the contrary, Fig. 17 illustrates three instances in which convex points do not lie on any convex planes of the original object.

We assume it is not worthwhile to form the convex hull of each polygonal face of the original polyhedron because such a procedure is indirect and for some points would be at least triply redundant. There does not seem to be any simple relation between the two-dimensional convex hull of points on each face of the original polyhedron and the three-dimensional convex hull of the total polyhedron. Whereas a given polyhedron has a unique convex hull, an infinitude of polyhedra can have the same convex hull.

Automatic identification of inside corners

As illustrated in Fig. 3, the minimum data required to specify a polyhedron is a vertex list and a topological map of vertex connectivity. The topological map is a list of which points are on each surface and also is a series of points connected by lines. Say that a typical line i connects point (x_1, y_1, z_1) to point (x_2, y_2, z_2). The parametric equations of each line i are calculated and stored:

$$x = x_1 + a_i t,$$

$$y = y_1 + b_i t,$$

$$z = z_1 + c_i t,$$

where

$$a_i = (x_2 - x_1)/l_i,$$

$$b_i = (y_2 - y_1)/l_i,$$

$$c_i = (z_2 - z_1)/l_i,$$

$$l_i = [(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2]^{\frac{1}{2}}.$$

Parameters a_i, b_i, c_i, l_i are stored and indexed by the line label. The equation of each plane surface j of the polyhedron is calculated and stored:

$$a_j x + b_j y + c_j z + d_j = 0,$$

where a_j, b_j, c_j are found from the cross product of any two lines on plane j that intersect with an angle less than 180° as determined by the condition:

$$a_k\, a_l + b_k\, b_l + c_k\, c_l > 0,$$

where k and l are the intersecting line labels. The normal to surface j has the parametric equation

$$x = a_j t + x_0,$$

$$y = b_j t + y_0,$$

$$z = c_j t + z_0,$$

where x_0, y_0, and z_0 are any point on the plane j, and

$$a_j^2 + b_j^2 + c_j^2 = 1.$$

Also,

$$d_j = -(a_j x_0 + b_j y_0 + c_j z_0).$$

It is important that each surface normal should point into the volume enclosed by the polyhedron. If not, the signs of a_j, b_j, and c_j must be reversed. A method for ensuring the correct sign has been described previously [19]. Each line of the polyhedron can be taken as a vector (a_i, b_i, c_i), and a normal vector (a_j, b_j, c_j) can be found for each surface. If line i is the intersection of two surfaces, say k and l, line i is an inside corner when the scalar triple product of the line i, the normal vector k, and the normal vector l is negative:

$$I = a_i, b_i, c_i,$$

$$K = a_k, b_k, c_k,$$

$$L = a_l, b_l, c_l,$$

$$[IKL] = \begin{vmatrix} a_i & a_k & a_l \\ b_i & b_k & b_l \\ c_i & c_k & c_l \end{vmatrix}.$$

Some simple instances are illustrated in Fig. 18.

Details of implementation

The techniques described have been reduced to practice with a FORTRAN program operating under VM/370 on an IBM System/370, model 168. A series of polyhedra and their convex hulls is shown in Fig. 19. Inside corners are found during hidden line elimination by the previously described program LEGER [18]. The convex hulls are drawn as soon as the edge list is known. It is not possible to measure the convex hull calculation times exactly because of the operating system overhead, but in general the convex hull determination takes about as much time as a hidden line elimination. Convex hull determination for the test figures shown varied from 0.20 to 0.84 second total CPU time, including drawing a wire frame picture and listing intermediate results. Time required increases approximately with the length of the convex point list and the number of surface loops in the convex hull, but we cannot yet show or estimate time requirements. A complex original object does not necessarily imply a complex convex hull. It has been observed that a simple object with a complex convex hull took more time for analysis than a complex object with a simple convex hull.

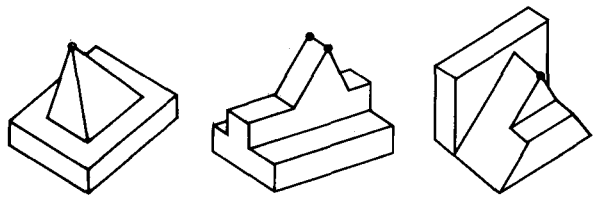
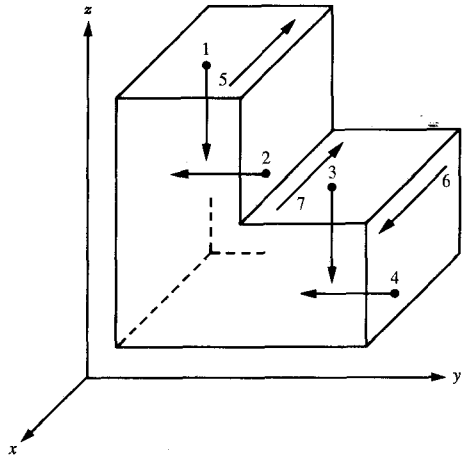


Figure 17 Three polyhedra in which a convex hull vertex is initially not on any polygon face that is a convex plane. There are no points of the object on the spatial side of a convex plane.



$$[\mathrm{IKL}] = \begin{bmatrix} a_i & A_k & A_l \\ b_i & B_k & B_l \\ c_i & C_k & C_l \end{bmatrix}$$ where line i is oriented to run counterclockwise on surface k.

$$[512] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix} = 1 \quad \text{outside corner}$$

$$[723] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} = -1 \quad \text{inside corner}$$

$$[643] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} = 1 \quad \text{outside corner}$$

Figure 18 Use of scalar triple product to detect inside (concave) corners.

Summary

We have described some techniques for determining the three-dimensional convex hull of real objects. Finding the convex edges of the convex hull is the essential step in this procedure. Toward this end, we have introduced the concept of identifying convex edges by projecting the entire object parallel to an edge and observing whether the point projection of the edge fell outside its associated star polygon. The star polygon test avoids the problem of having to find the two-dimensional convex

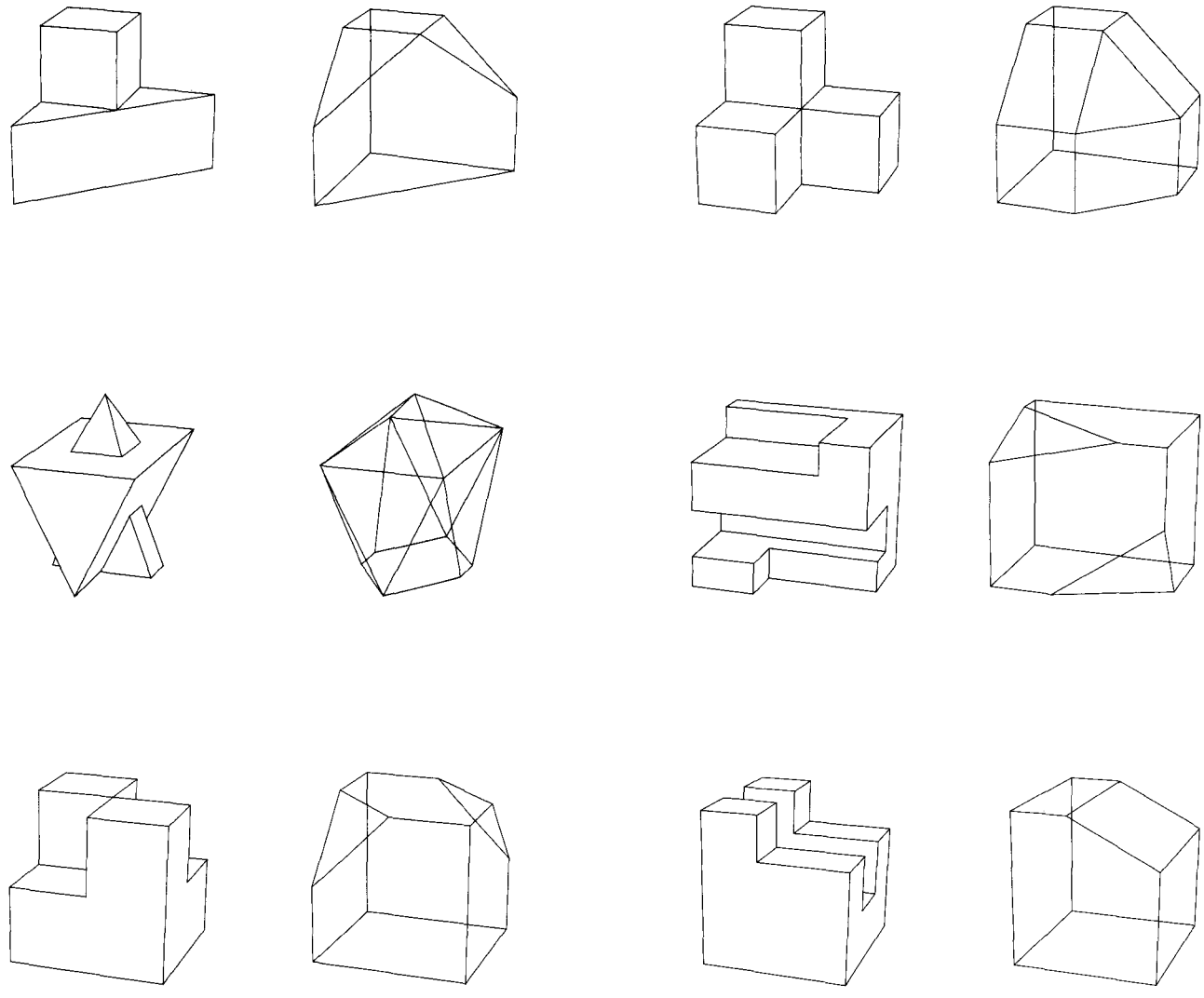


Figure 19 Computer generated examples of convex hulls.

hull of the projection of the original object. It is interesting that no two-dimensional convex hulls were ever required for this three-dimensional problem. Tactics for minimizing computation by culling the original vertex list have been presented but have been implemented to only a limited extent. Use of the present program for the determination of the three-dimensional convex hull of real objects presents no logistical burden in computer storage or time. Solving for a convex hull of a simple object is apparently as costly in total computational overhead as eliminating hidden lines.

Appendix: Finding plane surface loops of a convex polyhedron given the edges of the polyhedron

The task of finding the bounding loops of each plane surface of the three-dimensional convex hull given a complete list of convex edges is similar to certain problems in graph theory [20–22]. The edge list of a convex polyhedron can be used to construct a graph that is obviously isomorphic to a planar graph. The set of circuits that outline each face of this planar graph is the surface loop description of the convex hull. It is possible to find the faces of a planar graph without specific dimensional knowledge of the vertex points in space, but a description of this technique is beyond the scope of this paper. Our approach is more mechanistic. That is, while assembling edges to form surface loops, the convex edges being joined together are continually tested for coplanarity in three-dimensions. It has not been necessary to check that the loops are convex polygons.

1. Initialize an edge use array IUC(L) to all zeros. Every time an edge is used in a loop its corresponding IUC entry is increased by 1. As soon as a particular IUC

attains a value of 2, the associated edge cannot be used for any other loops. As soon as all elements in the array IUC(L) attain a value of 2, processing should stop.

2. Take an edge L1 from the list of edges. If the edge has an associated IUC(L1) = 2, take another edge. Processing stops when all IUC elements attain a value of 2 or all edges were addressed in this step once.
3. Choose some other edge L2 from the list of edges where IUC(L2) is less than 2.
4. Check each end point of edge L1 and edge L2 for a common vertex point. If edge L2 does not have an end point that is also on L1, go back to step 3. When edge L1 and edge L2 have a common vertex point, obviously these two edges intersect and form a plane in space.
5. The vertex point common to edges L1 and L2 and the other two points on these edges should not coexist simultaneously on any other loop, since this would imply coplanar loops, which is not possible on a convex polyhedron. Check that the three vertex points on L1 and L2 do not coexist on any other loop if any are yet identified. If coexistence occurs, go back to step 2.
6. Calculate the equation of the plane P in space formed by the intersecting edges L1 and L2.
7. Make certain that points on the vertex list lie either in or are to only one side of plane P. A convex plane can have no points on its spatial side. If points are found to lie on both sides of plane P, this indicates that P is not a convex plane so go back to step 2. A list of points in plane P can be stored at this time to speed up later steps in this procedure.
8. Classify the intersection of edges L1 and L2 according to type as shown in Fig. 20. At this time, two entries can be made in the loop table as shown in Fig. 21, and two vertex points have been identified: 1) the starting points SP of the loop, 2) the next match point MP on the loop. The starting point of the loop is the first point on the loop and is used to determine when a loop is finished. The match point is used to find the next line on the loop. Entries in the loop table are positive or negative and indicate the direction a line takes as it is used in a loop. When the line runs from LC1 to LC2, the direction is taken to be positive.
9. Take an edge L3 from the list of convex edges with IUC(L3) less than 2 and where L3 is not L1 or L2.
10. Determine whether any end point of L3 is MP. If no end of L3 is MP, repeat step 9. If an end of L3 is MP, determine whether the other end point (LE3) of L3 is SP, and if it is, add L3 to the loop table, end the current loop, increase the IUC value by one for each line used in the loop, and start another loop at step 1. If LE3 is not SP, ascertain that point LE3 is in the plane of L1, L2; if it is not, go back to step 9. If point LE3 is in the plane of L1, L2, add line L3 to the list of lines in the current loop. If LC1(L3) = MP,

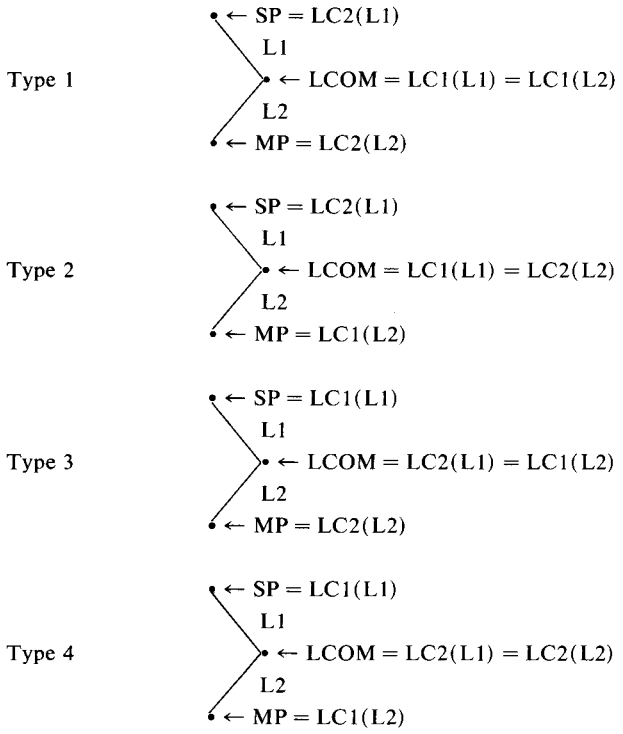


Figure 20 Classification of the intersection of edges L1 and L2. The common point is LCOM, SP is the starting point, and MP is the current match point.

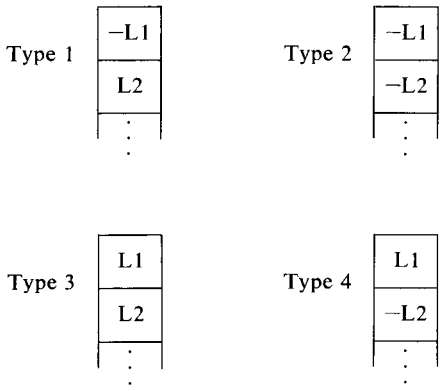


Figure 21 Initial entries in the loop table based on the type of initial intersection. First loop table entry for the object shown in Fig. 14 would be type 1.

First loop

N	LUP(1, N)
1	−1
2	2
3	−4

L	IUC (L)
1	1
2	1
3	0
4	1
5	0
6	0
7	0
8	0
9	0

Second loop

N	LUP(2, N)
1	−1
2	3
3	8
4	−5

L	IUC (L)
1	2
2	1
3	1
4	1
5	1
6	0
7	0
8	1
9	0

Third loop

N	LUP(3, N)
1	2
2	6
3	−7
4	−3

L	IUC (L)
1	2
2	2
3	2
4	1
5	1
6	1
7	1
8	1
9	0

Fourth loop

N	LUP(4, N)
1	4
2	6
3	9
4	−5

L	IUC(L)
1	2
2	2
3	2
4	2
5	2
6	2
7	1
8	1
9	1

Fifth loop

N	LUP(5, N)
1	−7
2	8
3	−9

L	IUC (L)
1	2
2	2
3	2
4	2
5	2
6	2
7	2
8	2
9	2

All 2's indicate end of processing

Figure 22 Growth of loop table for the object shown in Fig. 14.

the entry is positive, and if LC2(L3) = MP, the entry is negative. Reset the label MP to the vertex point LE3, and go back to step 9 to find a new line on the current loop.

The growth of a loop table for a simple object is shown in Fig. 22. It is advisable during debugging and in anticipation of calculation errors to prevent indefinite processing by limiting the number of times step 9 can be executed for a particular set of L1 and L2. In general, the entire list of convex edges should not be searched more often than the number of lines in the list.

Given convex edge list

L	LC1(L)	LC2(L)
1	1	2
2	1	5
3	1	6
4	1	7
5	2	3
6	2	5
7	2	8
8	3	4
9	3	8
10	4	5
11	4	6
12	6	7
13	7	8

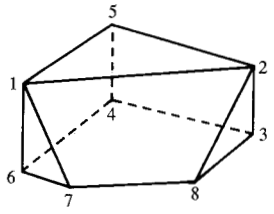


Figure 23 Evolution of a convex edge list (above) to a loop table (facing page) to a topological map. If LUP (K, N)=+, then SURFACE(K, N) = LC1(LUP(K, N)). If LUP(K, N)=−, then SURFACE(K, N) = LC2(LUP(K, N)).

After all edge loops of the convex planes have been found, a topological map can be easily derived from the loop table. The topological map is a list of all points on a surface in the order in which these points are connected together. Such a map is a listing of end points of the loop table where one point is taken from each line. When the entry is positive, the first point on the line is taken, LC1(L), and where the line entry is negative, the second point on the line is taken, LC2(L). A typical topological map is shown in Fig. 23.

References

1. G. Hadley, *Linear Algebra*, Addison-Wesley Publishing Co., Inc., Reading, Massachusetts, 1961, ch. 6.
2. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York, 1973, ch. 9.
3. A. Rosenfeld, *Picture Processing by Computer*, Academic Press, New York, 1969.
4. P. F. Lambert, "Designing Pattern Categorizers with Extremel Paradigm Information," *Methodologies of Pattern Recognition*, edited by S. Watanabe, Academic Press, New York, 1969.
5. G. -A. Burdet, "Generating All the Faces of a Polyhedron," *SIAM J. Appl. Math.* **26,** 479 (1974).
6. B. F. Mitchell, V. F. Dem' Yanov, and V. N. Malozemov, "Finding the Point of a Polyhedron Closest to the Origin," *SIAM J. Control* **12,** xxx (1974).
7. H. Freeman, "Computer Processing of Line-Drawing Images," *Comput. Surveys* **6,** 57 (1974).
8. M. L. Balinsky, "An Algorithm for Finding All Vertices of Convex Polyhedral Sets," *J. Soc. Indust. Appl. Math.* **9, 72** (1961).
9. N. Tuong, *L'Emploi des Projections Coniques Dans La Reconnaisance des Formes* (The Use of Conic Projections in Pattern Recognition), Department d'Informatique, Universite de Montreal, March 1969.
10. G. Nagy and N. Tuong, "Normalization Techniques for Handprinted Numerals," *Commun. ACM* **13,** 475 (1970).

Loop table

N	LUP(1, N)	LUP(2, N)	LUP(3, N)	LUP(4, N)	LUP(5, N)	LUP(6, N)	LUP(7, N)
1	−1	−1	2	−3	−5	−5	−8
2	2	4	−10	4	6	7	9
3	−6	13	11	−12	−10	−9	−13
4		−7	−3		−8		−12
5							−11

Topological map

N	SURFACE 1	SURFACE 2	SURFACE 3	SURFACE 4	SURFACE 5	SURFACE 6	SURFACE 7
1	2	2	1	6	3	3	4
2	1	1	5	1	2	2	3
3	5	7	4	7	5	8	8
4		8	6		4		7
5							6

Figure 23 (Continued from facing page.)

11. R. L. Graham, "An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set," *Inf. Process. Lett.* **1,** 123 (1972).
12. A. A. Jarvis, "On the Indentification of a Convex Hull of a Finite Set of Points in the Plane," *Inf. Process. Lett.* **2,** 18 (1973).
13. I. C. Braid, *Designing with Volumes*, Cantab Press, Cambridge, England, 1973.
14. J. Encarnacao and W. Giloi, "PRADIS-An Advanced Programming System for 3-D Display," *AFIPS Conf. Proc.* **40,** 1969, p. 985.
15. A. Appel and A. Stein, "The Interactive Design of Polyhedra," *Proc. ONLINE 72 Conference, Volume 2*, Online Computer Systems Ltd., Brunel University, England, 1972, p. 383.
16. P. Loutrel, "Determination of Hidden Edges in Polyhedral Figures: Convex Case," *Technical Report 400-145*, Department of Electrical Engineering, New York University, September 1966.
17. A. Appel, "Modeling in Three Dimension," *IBM Syst. J.* **7,** 310 (1968).
18. A. Appel, "The Notion of Quantitative Invisibility and the Machine Rendering of Solids," *Proc. ACM Nat. Meeting*, 1967, P. 387.
19. A. Appel and A. J. Stein, "Effecting Fast Correction of Surface Normals in Computer Generated Three-Dimensional Depictions," *IBM Tech Disclosure Bull.* **17,** 599 (1974).
20. F. Rubin, "A Search Procedure for Hamilton Paths and Circuits," *J. Assoc. Comput. Mach.* **21,** 576 (1974).
21. J. Hopcroft and R. Tarjan, "Efficient Planarity Testing," *J. Assoc. Comput. Mach.* **21,** 549 (1974).
22. T. Kamal, "A Systematic Method of Finding All Directed Circuits and Enumerating All Directed Paths," *IEEE Trans. Circuit Theory* **CT-14,** 166 (1967).

Received June 19, 1975; revised June 15, 1976

The authors are located at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598.