DIGITAL CONVEX HULLS FROM HIERARCHICAL DATA STRUCTURES

M.-M. Yau and S.N. Srihari

Department of Computer Science
State University of New York at Buffalo, Amherst, N.Y.

ABSTRACT

The convex hull is a useful tool in describing the shape of objects in two- and three-dimensional images. Since digital images can often be efficiently represented in high resolution by hierarchical data structures, it is useful to compute convex hulls directly from these data structures. In general, there is no unique minimal convex digital pattern that contains a given digital object, and for certain applications in shape analysis, an approximate image convex hull within the accuracy of discretization is sufficient. The algorithm presented in this paper computes such convex hulls by taking advantage of the implied sorting in a quadtree or octree representation. It reduces the problem to that of finding the convex and concave envelopes of the depth values of unobscured points along two opposite directions. The digital convex hull is computed by the intersection of the convex epigraph of the convex envelope and the convex hypograph of the concave envelope. Experimental results in the computation of convex hulls will be described.

RÉSUMÉ

L'enveloppe convexe est un outil utile pour décrire la forme des objets bi- et tri-dimensionnels. Étant donné que les images numériques peuvent souvent être représentées de façon efficace en résolution élevée par des structures hiérarchiques de données, il est utile de calculer les enveloppes convexes directement à partir de ces structures de données. En général, il y a plus d'un diagramme numérique convexe minimal qui contient un objet numérique donné et, pour certaines applications sur le plan de l'analyse des formes, une enveloppe convexe approximative dans les limites de la précision de discrétisation est suffisante. L'algorithme présenté dans le présent document calcule ce type d'enveloppe convexe en profitant du triage implicite dans une tétraarborescence ou une octoarborescence. Le problème se résume à trouver les enveloppes convexes et concaves des valeurs de profondeur des points non obscurcis le long de deux directions opposées. L'enveloppe convexe numérique est calculée par l'intersection de l'épigraphe convexe de l'enveloppe convexe et de l'hypographe convexe de l'enveloppe concave. Une description sera faite des résultats expérimentaux du calcul des enveloppes convexes.

1. INTRODUCTION

In the analysis and recognition of the shape of a solid, the convex hull is a useful computational tool. The convex hull of a solid provides not only an approximation of its three-dimensional (3D) shape, but also allows exact shape description in terms of the convex deficiency (which is the set difference between the convex hull and the solid).

Solids in 3D images, such as those produced by computed tomography [SRIHARI 80] are usually represented as points in discrete space. Present 3D convex hull algorithms either pertain to points in continuous space or have been adapted from continuous space algorithms without taking advantage of the discreteness of data. In this paper we propose an approach to 3D convex hull computation that is based on a practical characterization of digital convex hulls and that makes use of sorting usually present in image data structures.

Although convexity is a well-understood concept in classical analysis and geometry, there are many definitions of digital convexity. In Sections 2 and 3 we review convexity and existing convex hull algorithms for continuous and discrete 3D spaces respectively. Section 4 presents the theoretical background of the algorithm. Implementation with tree data structures is discussed in Section 5.

2. CONTINUOUS SPACE

A set S of points in Euclidean space is convex if and only if the line segment joining every pair of its points consists entirely of points of S. The convex hull of S, denoted H(S), is the smallest convex set containing S, i.e., H(S) is the intersection of all convex sets containing S. Several algorithms for obtaining H(S) from a set S of n points are known. In the 2D case, the time complexity of determining H(S) is O(nlogn) for an arbitrary set of points [SHAMOS 75] and O(n) for the vertices of a polygon [SKLANSKY 72]. Algorithms for the 3D case are considered in [APPEL 76], [DIJKSTRA 76] and [PREPARATA 77].

The algorithm of [APPEL 76] computes the convex hull of a polyhedron given a set S of its n vertices. For each pair of points of S, the line segment e joining them is determined. By projecting S onto a plane perpendicular to e and testing if the projection of e is enclosed within the 2D convex hull of the remaining (n-2) projected points of S, it is determined whether or not e should be included in a list of edges of H(S). Elements of the list are examined for loops which are then assembled into polygonal faces. Since there are n(n-1)/2 possible line segments e and the test for containment in a 2D convex hull is O(nlogn), the algorithm complexity is $O(n^3 \log n)$.

The 3D convex hull algorithm of [DIJKSTRA 76] pertains to a set of points such that no four points lie on a plane. H(S) is initialized to the two oriented triangular faces defined by three arbitrary points in S. For each additional point p, a face in the current H(S) is either a light face or a dark face depending on whether imaginary light rays emanating from p illuminates the face from outside or inside of H(S). If every face in the current H(S) is dark with respect to p then p lies in the current H(S). If not, certain faces (i.e., their edges) are deleted from H(S) and certain others are included in H(S), as determined by the boundary between the set union of light faces (the light cap) and the set union of dark faces (the dark cap)-which forms a closed circuit. Faces in the light cap are removed from H(S), and new faces connecting p to edges on the boundary are added to H(S). Since for each point in S, at most O(n) number of edges need to be examined, the complexity of the algorithm is $O(n^2)$.

An algorithm based on "divide-and-conquer" can reduce the complexity of 3D convex hull computation to O(nlogn) [PREPARATA 77]. First we sort S into a list according to a coordinate and partition the list into two nearly equal subsets. The convex hulls A and B of the subsets are recursively computed. The sorting guarantees nonoverlapping A and B, which can then be merged in linear time--by constructing a cylindrical triangulation τ which is tangential to A and B along two circuits and by removing portions of A and B that are obscured by τ.

3. DISCRETE SPACE

3.1 Digitization

A method of sampling a 3D volume is to use an array of points, called digital points, whose coordinates are integers. With each digital point $V \equiv (V1,V2,V3)$ of this lattice, we associate the set of points (x1,x2,x3) of continuous space satisfying $xi-1/2<Vi\leq xi+1/2$ and refer to the resulting volume element as voxel V (analogous to pixel for a 2D picture element).

If Q is a set of points representing an object in continuous space then we say that a set of voxels S is the digital image of Q, denoted $S = I(Q)$, if $S = \{V | V$ is a voxel and $V \cap Q \neq \phi\}$.

A set of voxels S is referred to as a digital region. If the voxels of S are connected (see [SRIHARI 80] for 3D digital connectivity) then S is said to be a digital solid.

3.2 Digital Convexity

A number of different definitions of digital convexity are given in the literature.

3.2.1 Convex Preimage

A digital region S is convex if there exists at least one convex object Q such that $S=I(Q)$. This intuitively appealing definition does not lead to a finite procedure for testing convexity.

3.2.2 Midpoint Convexity

A digital region S is convex if whenever voxels $X,Y \in S$, at least one of the midpoint cover voxels is in S. A midpoint cover of X and Y is the smallest set of midpoint voxels such that for every segment between a point of X and a point of Y there is a midpoint voxel included. Z is a midpoint voxel of X and Y if and only if Z contains the midpoint of a segment between a point in X and a point in Y [HODES 70].

3.2.3 Convex Solid

A digital region S can be regularized by half-cell expansion and the convexity of S determined by the convexity of its half-cell expansion E(S) [SKLANSKY 72]. E(S) is the discrete image of S in a new lattice obtained by displacing the current lattice by half cell in the three directions.

The half-cell expansion can be used to define a digital convex solid [KIM 80]. A digital solid S is convex if it is simple and there is a convex solid Q such that $I(Q)=E(S)$. A simple solid S is a finite 6-connected set of voxels having no pair of voxels $X,Y \in S$ such that the line segment joining the centers of X and Y is parallel to an axis and lies outside the voxels of S. The simple solid requirement is introduced to eliminate edge effects due to single voxel gaps and ensure simple connectedness.

It can be shown that this definition of convexity leads to a finite procedure for testing convexity--one that utilizes concepts of semidigital points and nearness of points. A point $X = (x1,x2,x3)$ is said to be a semidigital point if at least one of the xi is an integer, and two points X and Y are said to near each other if $\max_i \{|xi-yi|\} < 1$. A simple digital solid S is digital convex if and only if the Euclidean convex hull of the digital points of S, H(S), is such that every semidigital point on the faces of H(S) is near a digital point of S [KIM 80]. Thus a procedure to test the convexity of a digital region S is to first test for simplicity of S, then compute H(S) from the digital points of S and finally test for nearness of the semidigital points on H(S) to the digital points of S.

3.3 Digital Convex Hull

In terms of the last characterization of digital convexity, the digital convex hull $H(S)$ of a digital region S is the smallest (in terms of set inclusion) simple digital solid that contains S and each point of H(S) is near a point of $H(S)$. Let K(S) be the set of all digital points in H(S). The following result is shown in [YAU 81a]: if S is 6-connected then $H(S)=K(S)$. An example of S, H(S) and $H(S)$ is shown in Fig. 1.

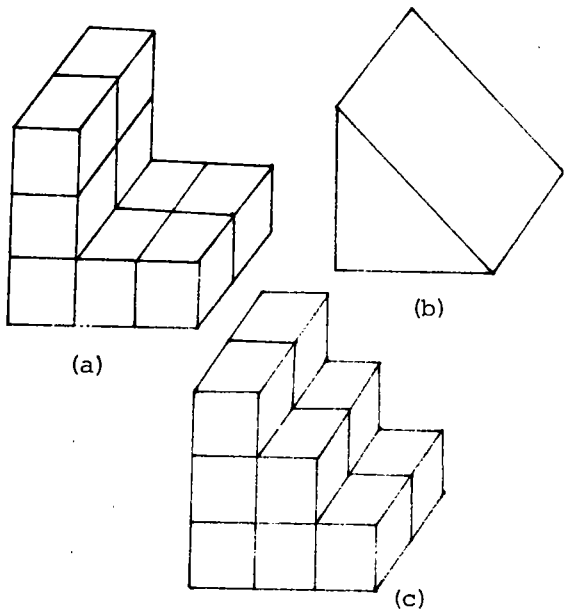


Fig. 1. A digital solid (a), the Euclidean convex hull of its digital points (b) and its digital convex hull (c).

4. THEORETICAL BACKGROUND

In this section we will demonstrate that the problem of computing $H(S)$ of a digital solid S is equivalent to

(i) determining two functions of two variables called the ceiling and floor functions (which give the maximum and minimum extent of S along opposite directions),

(ii) determining the concave and convex envelopes of the ceiling and floor functions respectively, and

(iii) determining the digital points in the set intersection of the convex epigraph and concave hypograph of the respective envelopes.

4.1 Ceiling and Floor

Let S be a finite, 6-connected 3D digital solid. Let $S' = \{(x1,x2) \mid (x1,x2,x3) \in S\}$. We define the floor and ceiling of S to be the functions defined on domain S' as

$\alpha_s(x1,x2) = \min\{z \mid (x1,x2,z) \in S\}$ and

$\beta_s(x1,x2) = \max\{z \mid (x1,x2,z) \in S\}$ respectively.

4.2 Convex Set

If $S = \{Si\}$ is a set of points in d-dimensional Euclidean space $\mathbb{R}^d$ then it is known that [MANGASARIAN 69] $H(S) = \{X \mid X = \sum_i m_i Si$ where $\sum_i m_i = 1$ and $m_i \geq 0\}$. Each point of $H(S)$ is said to be a convex combination of the points of S. If $S = H(S)$ then S is a convex set.

4.3 Convex and Concave Functions

Let $f(X)$, $X = (x1,x2)$ be a real-valued function of two variables defined on a domain $D \in \mathbb{R}^2$. The graph of f is defined as the set $G(f) = \{[X,f(X)] \mid X \in D\}$.

If D is a convex set then f is said to be convex if and only if for $X,Y \in D$, $0 \leq m \leq 1$, $(1-m)f(X) + mf(Y) \geq f((1-m)X+mY)$. If f is convex then the straight line segment joining points $[X,f(S)]$ and $[Y,f(Y)]$ lies above $G(f)$ between X and Y.

If D is a convex set then f is said to be concave if and only if for $X,Y \in D$, $0 \leq m \leq 1$, $(1-m)f(X) + mf(Y) \leq f((1-m)X+mY)$. If f is concave then the straight line segment joining points $[X,f(X)]$ and $[Y,f(Y)]$ is below $G(f)$ between X and Y.

Thus if f is convex then -f is concave and vice versa.

4.4 Convex and Concave Envelopes

Let $H(D)$ denote the Euclidean convex hull of D, the domain of f. The convex envelope f^- of f is defined by the graph

$$G(f^-) = \{[X,e] \mid X \in H(D),\ e = \min\{\sum_i m_i f(Xi) \mid \sum_i m_i = 1,\ m_i \geq 0 \text{ and } X = \sum_i m_i Xi,\ Xi \in D\}\}.$$

The concave envelope f^+ of f is given by the graph $G(f^+)$ which is defined similar to $G(f^-)$ with max in place of min.

4.5 Epigraph and Hypograph

The epigraph and hypograph of f are defined to be $G^+(f) = \{[X,e] \mid X \in D,\ f(X) \leq e$ and $e \in \mathbb{R}\}$ and $G^-(f) = \{[X,e] \mid X \in D,\ f(X) \geq e$ and $e \in \mathbb{R}\}$ respectively. Thus the epigraph and hypograph of f are the two infinite regions above and below the graph $G(f)$.

It is shown in [MANGASARIAN 69] that if f is defined on a convex set D, then f is convex if and only if $G^+(f)$ is a convex set and f is concave if and only if $G^-(f)$ is a convex set.

Theorem 1: $G^+(\alpha_s^-) \cap G^-(\beta_s^+) = H(S)$.

It immediately follows that the digital convex hull of S is given by the digital points common to $G^+(\alpha_s^-)$ and $G^-(\beta_s^+)$.

The following algorithm for computing $H(S)$ of a 6-connected digital solid can now be stated.

Algorithm Digital Convex Hull

Step 1 Determine the ceiling and floor functions α_s and β_s from S.

Step 2 Apply Algorithm Convex Envelope to α_s to find α_s^-.

Step 3 Apply Algorithm Convex Envelope to $-\beta_s$ to find β_s^+.

Step 4 Determine the digital points in the intersection of $G^+(\alpha_s^-)$ and $G^-(\beta_s^+)$.

4.6 Computing the Convex Envelope

Since the convex combination is a linear combination, $G(\alpha_s^-)$ is a piecewise linear convex surface. It has the property that for each of its points v, there exists at least one plane P which contains v such that all of $G(\alpha_s^-)$ is contained in one closed half-space produced by P. P is called a supporting plane of $G(\alpha_s^-)$ at v.

The intersection of any supporting plane with $G(\alpha_s^-)$ contains at least one extreme point of $G(\alpha_s^-)$. A point v in $G(\alpha_s^-)$ is said to be an extreme point of $G(\alpha_s^-)$ if there do not exist two different points v^1, v^2 in $G(\alpha_s^-)$ such that $v = mv^1 + (1-m)v^2$, $0<m<1$. It follows from the definition of α_s^- that all the extreme points of $G(\alpha_s^-)$ are points in $G(\alpha_s)$.

The line joining two distinct extreme points of $G(\alpha_s^-)$ is called an edge of $G(\alpha_s^-)$ if it is the intersection of $G(\alpha_s^-)$ with a supporting plane.

The triangular face defined by three distinct extreme points of $G(\alpha_s^-)$ is called a face of $G(\alpha_s^-)$ if it is the intersection of of $G(\alpha_s^-)$ with a supporting plane. We shall assume all faces are oriented and use the convention that if a polygonal face is defined by $v^1 \ldots v^m$, where $v^1,\ldots,v^m$ are its vertices, then an observer is on the outside of the face if $v^1,\ldots,v^m$ appear in counterclockwise order. Otherwise the observer is on the inside of the face. The inside of a face of $G(\alpha_s^-)$ is the side on which $G(\alpha_s^-)$ lies. We shall define the normal $\eta(p) = (\eta_1(p), \eta_2(p), \eta_3(p))$ of a face p to be the outward normal and the angle $\psi(p,q)$ between two faces p and q to be the angle between $\eta(p)$ and $\eta(q)$.

Since the construction of the convex envelope α_s^- is equivalent to determining all the faces of $G(\alpha_s^-)$, we would like to derive a criterion for the next face, assuming that we can always determine at least one face of $G(\alpha_s^-)$.

Theorem 2: Let $p = v^1v^2v^3$ be a face of $G(\alpha_s^-)$. Then $q = v^3v^2v'$ is a face of $G(\alpha_s^-)$ if and only if $\psi(p,q) = \max_{v \in G(\alpha_s)} \{\psi(p, v^3v^2v) \mid \eta_3(v^3v^2v) < 0\}$.

The proof follows from the observation that q is a face of $G(\alpha_s^-)$ <=> the plane containing q is a supporting plane <=> there does not exist $v'' \in G(\alpha_s)$ such that v^1 and v'' lie on opposite sides of q.

Theorem 2 suggests a procedure for determining the faces of $G(\alpha_s^-)$, given a set of starting faces. Usually it is expensive to compute the angle between two faces; we would like to reduce, as much as possible, the number of times this operation is performed. This is done by first preprocessing the prints in $G(\alpha_s)$ to remove from further consideration points that are not extreme points on two mutually orthogonal planes and secondly adopting an efficient search strategy for the face $q = v^3v^2v^1$ of $G(\alpha_s^-)$, given $p = v^1v^2v^3$ is a face of $G(\alpha_s^-)$.

4.6.1 Preprocessing

Let $x^1 < \ldots < x^{|S_1'|}$, $y^1 < \ldots < y^{|S_2'|}$ be the elements in $S_1' = \{x1 \mid (x1,x2) \in S' \text{ for some } x2\}$, $S_2' = \{x2 \mid (x1,x2) \in S' \text{ for some } x1\}$ respectively. Let $\alpha_{s_1}(x^i)$, $\alpha_{s_2}(y^i)$ be the induced floor functions of α_s on $S_1'(x^i) = \{x2 \mid (x^i,x2) \in S'\}$, $S_2'(y^i) = \{x1 \mid (x1,y^i) \in S'\}$ respectively, $i = 1,\ldots,|S_1'|$, $j = 1,\ldots,|S_2'|$. If we order the points in $G(\alpha_{s_1}(x^i))$ or $G(\alpha_{s_2}(y^i))$ by their first coordinates, we can regard them as the ordered vertices in a weakly externally visible polygon [TOUSSAINT 80] and apply the linear convex hull algorithm in [SKLANSKY 72] to find the extreme points in their convex envelopes. Since extreme points in $G(\alpha_s^-)$ are necessarily extreme points in both $G(\alpha_{s_1}^-(x^i))$ and $G(\alpha_{s_2}^-(y^i))$ for some $i \in \{1,\ldots,|S_1'|\}$, $j \in \{1,\ldots,|S_2'|\}$, we apply the above procedure to generate from S' the reduced set A over which the convex envelope is to be computed.

4.6.2 Search Strategy

Given a face $p = v^1v^2v^3$ on $G(\alpha_s^-)$, our strategy is to use divide-and-conquer to find $q = v^3v^2v^1$, a face of $G(\alpha_s^-)$ that stands on the edge v^3v^2. Initialize S_1 to be the square that contains A. Either its contents are simple enough to process with some decision procedure or we recursively subdivide S_1 into its four quadrants. Let $u^1u^2u^3$ be the projection of $v^1v^2v^3$ on S_1. Initialize w to 0. Let

$$\delta_1(z) = \min_{(x1,x2)\ S_1} \{\psi(p,r) \mid r = v^3v^2(x1,x2,z)\},$$

$$\delta_2(z) = \max_{(x1,x2)\in S_1} \{\psi(p,r) \mid r = v^3v^2(x1,x2,z)\}.$$

where $z = \max\{x3 \mid (x1,x2,x3) \in G(\alpha_s),\ (x1,x2)\in S_1\}$. S_1 is simple enough if one of the following conditions hold:

1. S_1 lies on the same side of u^3u^2 as u^1, in which case $\eta_3(r) \geq 0$, q is not in S_1.
2. $S_1 \cap A = \emptyset$ or $\delta_2(z) < w$, q is not in S_1
3. There is only one point v" projected onto S_1.

In case of condition (3), v" is returned as a candidate for an extreme point. Among the values returned from the subquadrants of S_1, the one that gives rise to the largest value of $\psi(p,v^3v^2v")$ is used to form q.

In case of S_1 not simple, w is updated to be the largest $\delta_1(z)$ computed over the four quadrants of S_1. The computation of $\delta_1(z)$ and $\delta_2(z)$ is made simple by noticing that the extrema always occur at two different corners of the square S_1.

We shall discuss in the next section the implementation details of obtaining the set of starting faces and the data structure for the various parts of the algorithm.

5. IMPLEMENTATION WITH HIERARCHICAL DATA STRUCTURES

The enormous amount of data in 3D digital images calls for efficient data and control structures for image analysis. Frequently serial section (or slice-by-slice) processing is employed to alleviate problems of storage and thrashing between main and secondary memory. The hierarchical quadtree data structure is useful for compactly representing slice data, and its generalization known as the octree representation ([JACKINS 80], [SRIHARI 80]) is suitable for graphics display and search operations. The octree is based on recursively subdividing a cubical volume into octants until each octant has uniform properties, e.g., every voxel of the octant is full (or has value 1). In this section we discuss an implementation of the digital convex hull algorithm for a digital solid that is represented in such a data structure.

Generation of Floor and Ceiling

Let T(S) be the octree representation of a 6-connected digital solid S. The quadtrees $T(G(\alpha_s))$ and $T(G(\beta_s))$ representing the floor and ceiling of S can be computed by recursively visiting the 0,...,7 subtrees in order, associating the (2i)-th and (2i+1)-th sons at a given level of T(S) to the i-th son at the same level of $T(G(\alpha_s))$ and $T(G(\beta_s))$, for i = 0,1,2,3. The values of octree leaves are written onto the corresponding quadtree leaves in the following manner: The value of a leaf in $T(G(\alpha_s))$ stays the same, once it has been assigned a value, while the value of a leaf in $T(G(\beta_s))$ is always overwritten by the value of the current octree leave associated with it. Values of nonterminal nodes in $T(G(\alpha_s))$ and $T(G(\beta_s))$ are the minimum and maximum respectively of their son values.

As an example, consider the 6-connected digital solid S and its octree representation in Figure 2(a) and (c). $G(\alpha_s)$ and its quadtree representation $T_1 = T(G(\alpha_s))$ are shown in Figure 3. Non-existing branches are indicated by dotted lines. We shall describe only the processing of T_1 to find the convex envelope, since the computation of the concave envelope from $T_2 = T(G(\beta_s))$ is similar.

Preprocessing

To construct the induced convex envelope $\alpha_{s_1}(x^i)$ and $\alpha_{s_2}(y^i)$, the binary subtrees of T_1 are traversed in a manner similar to the reversed process of the construction of quadtrees from binary trees [YAU 81b]. The

order in which the binary tree leaf nodes are visited corresponds to the sorted order of the vertices on a polygon. First the extreme points of $G(\alpha^-_{S_1}(x^i))$ are determined for $i = 1,\ldots,|S'_1|$. Nodes that do not correspond to an extreme point are removed and values of their ancestors are updated if necessary. (Fig. 4(a), (b)). Secondly, the extreme points of $G(\alpha^-_{S_2}(v^i))$ are determined for $j = 1,\ldots,|S'_2|$, using only the nodes remaining on T_1 after the first reduction step, removing further nodes that do not correspond to an extreme point in the current step (Fig. 4(c), (d)). The projection A on S' of extreme points common to the two steps are represented by the shaded squares in Fig. 5a.

Determination of the Starting Faces

The extreme points $w^1,\ldots,w^k$ of H(A), sorted in counterclockwise order, can be determined by applying the procedure for computing $\alpha^-_{S_1}(x^i)$, described in Section 4.6,

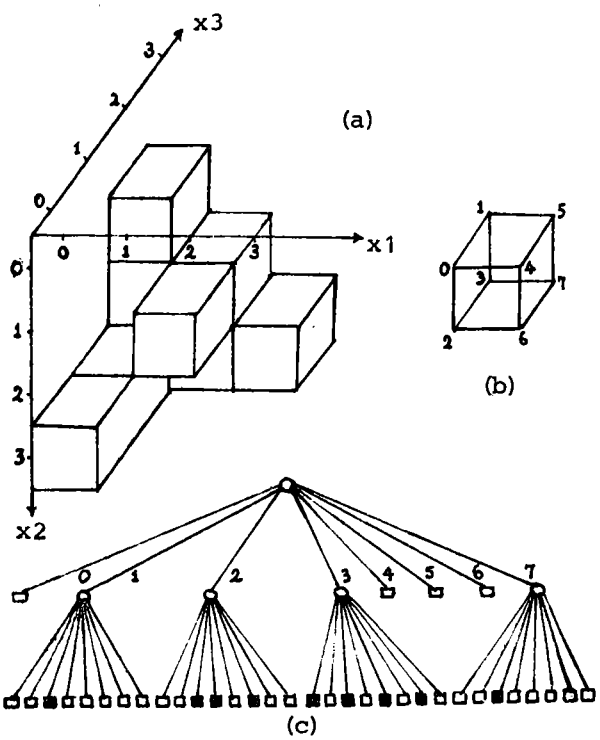


Fig. 2.(a) 6-connected digital solid S, (b) Octant indices, (c) Octree representation T(S) of (a).

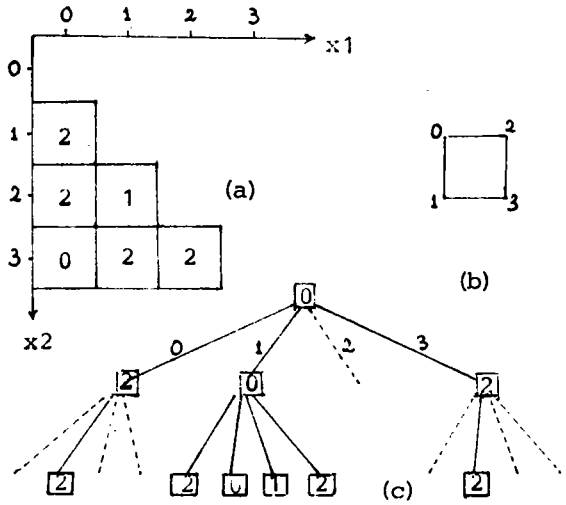


Fig. 3. (a) $G(\alpha_S)$, (b) quadrant indices, (c) initial $T_1 = T(G(\alpha_S))$, (undefined branches are indicated by dotted lines).

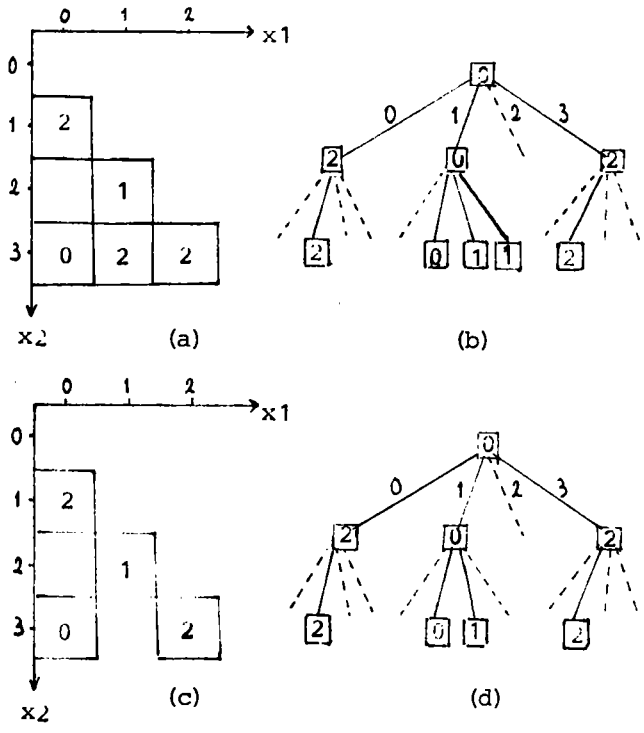
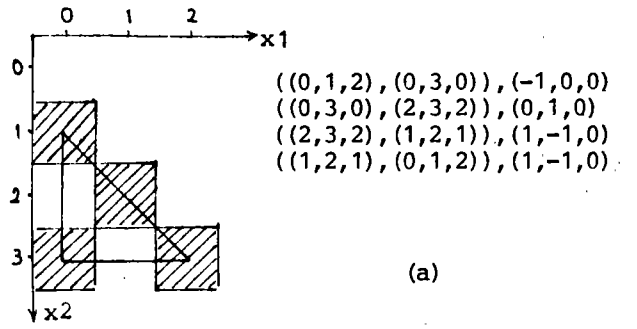
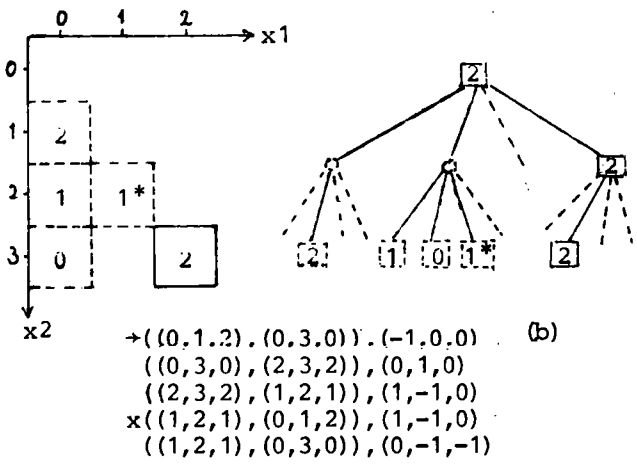


Fig. 4.(a) & (b) T1 after construction of first face, (c) & (d) T1 after construction of second face.





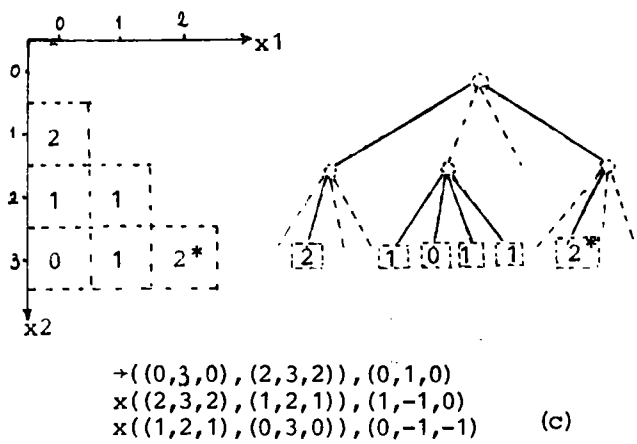


Fig. 5.(a) Boundary of $H(A)$ and initial entries on Q, (b) & (c) T_1 in 2 steps of construction of convex envelope (→: current top entry on Q, x: entry to be removed from Q, *: extreme point for the construction of new face, +: entry to be added to Q, [ ]: marked entry).

to the floor and ceiling of A, and suitably adjusting the sign of the ceiling. For each consecutive w^i and w^{i+1} (addition is modulo k), we construct $v = (w_1^i, w_2^i, w_3^i + c)$ for some positive integer c. Then $p = v^{i+1}v^iv$ is a starting face for the construction of the convex envelope. The normal to p can be expressed as an integer-valued order-triple, since its vertices are digital points $(v^iv^{i+1}, \eta(p))$ is added to a queue Q. The initial value of Q for the particular example we are considering is given in Fig. 5a.

Construction of Convex Envelope Faces

The top entry $(u^1u^2, \eta(p))$ is removed from Q and the unmarked nodes on T_1 are searched according to the decision rules described in Section 4.6 to return u_3, the extreme point on $G(\alpha_s^-)$ that maximizes the angle $\psi(p, u^1u^2u^3)$. $(u^1u^3, \eta(u^1u^2u^3))$ and $(u^3u^2, \eta(u^1u^2u^3))$ are entered onto Q if u^1u^3 and u^3u^2 are not present in the first field of an existing element of Q. Otherwise the matching entries are removed from Q. The truncated value of $q = u^1u^2u^3$ at each digital point over the projection of q on S' are entered as marked values on T_1. Value of nonterminal nodes on T_1 are updated so that the minimum is taken only over unmarked nodes. The process is repeated until Q is empty.

T_1 and Q at the two iterations of the process is illustrated in Figure 5(b) and (c).

Construction of Convex Epigraph

The octree representation $T(G^+(\alpha_s^-))$ is constructed by recursively visiting T_1 subtrees, creating at each ith son the corresponding (2i) and (2i+1)-th sons of the octree, for i=0,1,2,3. Values of the octree node is set to full if its third coordinate in 3D cartesian space is not smaller than the value of its corresponding T_1 node, and empty otherwise. The octree representation of the digital convex hull is obtained by computing the intersection of the octrees of the corresponding epigraph and hypograph. This is done by traversing the two trees in parallel, setting a node to full only if it is full in both trees (Fig. 6).

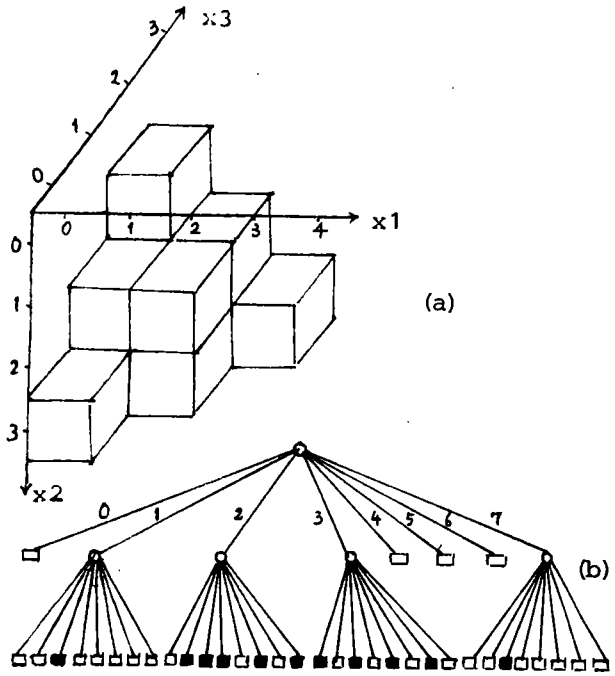


Figure 6. a) Digital convex hull $H(S)$, (b) Octree representation $T(H(S))$.

6. SUMMARY AND CONCLUSIONS

Computation of the convex hull of a solid is useful in 3D analysis. Several algorithms have been proposed for points in 3D continuous space. Methods for computing the digital convex hull from data structures that are suited to high resolution 3D digital images are needed. We have proposed a new approach to 3D digital convex hull computation that appears to be suitable for data structures in the form of serial sections or hierarchies.

ACKNOWLEDGEMENT

This work was supported in part by the National Science Foundation Grant IST-80-10830 and in part by SUNY/Buffalo BRSG funds. The authors wish to thank Ms. Gloria Calato who prepared the manuscript.

REFERENCES

[APPEL 76] A. Appel and P.M. Will, Determining the three-dimensional convex hull of a polyhedron, IBM J. Res. Develop., 20, 590-601, Nov. 1976.

[DIJKSTRA 76] E.W. Dijkstra, The problem of the convex hull in three dimensions, Ch. 24 in A discipline of programming, Prentice-Hall, N.Y., 1976.

[HODES 70] L. Hodes, Discrete approximation of continuous convex blobs, Siam J. Appl. Math., 19, 477-485.

[JACKINS 80] C.L. Jackins and S.L. Tanimoto, Oct-trees and their use in representing 3D objects, Comp. Graph. and Image Proc., 14, 249-270, 1980.

[KIM 80] C.E. Kim and A. Rosenfeld, Convex digital solids, TR-929, U. of Maryland, Computer Science Center, 1980.

[MANGASARIAN 69] O.L. Mangasarian, Nonlinear Programming, McGraw-Hill, N.Y., 1969.

[PREPARATA 77] F.P. Preparata and S.J. Hong, Convex hulls of finite sets of points in two and three dimensions, Comm. ACM, 20, 87-93, 1977.

[SHAMOS 75] M.I. Shamos and D. Hoey, Closest-point problems, Proc. 16th Ann. IEEE Symp. on Foundations of Computing, 151-162, 1975.

[SKLANSKY 72] J. Sklansky, Measuring concavity on a rectangular mosaic, IEEE Trans. on Comput., C-21, 1355-1364, 1972.

[SRIHARI 80] S.N. Srihari, Representation of three-dimensional digital images, TR-162, Dept. of Computer Science, SUNY/Buffalo, July 1980.

[TOUSSAINT 80] G.T. Toussaint, Pattern recognition and geometrical complexity, Proc. 5th Int. Conf. Pattern Recog., 1324-1347, 1980.

[YAU 81a] M-M. Yau, Hierarchical representation of three-dimensional digital solids, forthcoming Ph.D. dissert., Dept. of Computer Science, SUNY/Buffalo.

[YAU 81b] M-M. Yau and S.N. Srihari, Recursive generation of hierarchical data structures for multidimensional digital images, TR-170, Dept. of Computer Science, SUNY/Buffalo, Feb. 81. Also to appear in Proc. PRIP-81, Dallas, TX.