COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING 45, 196–214 (1989)

Fast Surface Tracking in Three-Dimensional Binary Images

DAN GORDON* AND JAYARAM K. UDUPA

Medical Image Processing Group, Department of Radiology, University of Pennsylvania, 418 Service Drive, Philadelphia, Pennsylvania 19104-6021

Received July 8, 1987; revised July 29, 1988

In medical 3-dimensional display, an input scene is represented by an array of volume elements (abbreviated as "voxels"), and an object in the scene is specified as a "connected" set of voxels. In such applications, surface tracking is an important, and often time-consuming, precursory step. One of the most efficient surface detection algorithms reported in the literature tracks a specified surface of a 3-dimensional object by visiting each boundary face in the surface twice. In this paper, we present a new definition of discrete objects and boundaries that leads to a modified algorithm, which, on the average, visits only one-third of the boundary faces twice (and the rest once). The algorithm has been implemented in our display software package and is found to achieve a run-time reduction of approximately 35%. This timing includes the computation of surface-normal information which is needed for the realistic rendering of surfaces. Without this computation, the saving would be about 55%. © 1989 Academic Press, Inc.

1. INTRODUCTION

The advent of new imaging modalities such as computerized tomography, magnetic resonance imaging, ultrasound imaging, and positron emission tomography has made possible imaging cross sections of the human body [20]. By imaging cross sections contiguously placed over a region of the body, it is possible to determine the distribution of a specific physical property (e.g., X-ray attenuation in computerized tomography) at discrete rectangular grid points over a 3-dimensional (3D) region of the body. Recent years have seen an upsurge of activities relating to methodologies of making use of information in such 3D digital images [51]. One may identify three components in such activities—*visualization* which is concerned with methods of displaying the 3D structures of organs and organ systems, *quantitation* which deals with techniques of making measurements on the structures, and *manipulation* which has to do with altering the structures either for better visualization or for quantitation. The current activities in this area consist of the development of algorithms for visualization, manipulation, and quantitation, their hardware/software implementation, evaluation of their diagnostic and therapeutic uses, and clinical validation. The present paper addresses a problem related to visualization.

The methodologies of visualization can be broadly classified into two groups—those that are (or can be) implemented on the mini-computer of the imaging equipment (such as the computerized tomography scanner), and those that need special additional hardware. Methods of varifocal mirror display [3, 35], optical and holographic display [25, 31], and specialized implementation of display algorithms using VLSI technology [13, 33, 19, 28, 26, 5] belong to the latter category. The

*Department of Computer Science, Texas A & M University, College Station, TX 77843-3112.

0734-189X/89 $3.00

former group, to which the subject of the present paper belongs, may be further subdivided into two subcategories—those approaches that first produce a surface description of the 3D object and then generate a depiction of the surface on a 2-dimensional screen [34, 12, 42, 21, 44, 7, 46], and those that generate a depiction directly from the object description [18, 22, 41, 15, 54, 9, 10, 27, 37, 8, 11, 39, 23, 47]. The main advantages of the surface-based approaches are (i) a connected object can be isolated automatically from the rest of the background which is useful in some applications, (ii) the volume of meaningful objects, which is often a very useful physiological parameter, can be computed, and (iii) high-quality shaded-surface images can be generated. (An important disadvantage of these approaches, at the present time, seems to be that surface representation is not appropriate for interactive manipulation of objects.) An important precursory step in these approaches to visualization is the formation of object surfaces.

There are several approaches to surface formation. One approach is to first obtain the borders of the object on each cross section and then patch them using triangular [12, 44, 7, 46], trapezoidal [34], spline [42], or square [50] surface patches, or using solid primitives [43, 4]. An alternative is to use the object itself to define its surface and then track the surface either using the 3D binary image [1, 53, 2] resulting from segmenting (say, by thresholding) the cross-sectional images, or using the cross-sectional (gray-level) images directly [30, 55, 32]. The main advantage of surface tracking in binary images is that it is considerably faster than that in gray-level images. This combined with the previously stated advantages were the major considerations in the design of the turnkey display system reported in [48].

The present paper focuses on the surface tracking algorithm [1] (hereafter referred to as BD, an acronym for boundary detection) implemented in the system in [48]. BD treats a 3D binary image to be equivalent to a 3D array of cubes each of which has a value of "0" or "1" assigned to it. Given such an array and a face shared by a 1- and a 0-cube, BD produces a list of all those faces common to a 1- and a 0-cube that are "connected" to the given face. In the process, each output face is visited twice, and a hashing mechanism is used to keep track of the already-visited faces (when a face is visited the second time it is removed from the hash table).

In this paper, we present a new definition of discrete surfaces and a modification to BD. The modified algorithm (which was first briefly outlined in [16]) visits, on the average, only $\frac{1}{3}$ of the boundary faces twice. Since the rest of the boundary faces are visited only once, hashing will be performed on only $\frac{1}{3}$ of the faces. This saving combined with the special tracking options available in the new algorithm result in an overall time saving of at least 35%. Such saving in time is vital to reducing the cost of applying 3D display methodologies to medical problems. We do not elaborate this point further, but refer the reader to [14] for a detailed discussion of the need for high-speed surface detection in medical computer graphics.

An important unsolved problem relating to the new algorithm is the proof of its correctness. While the correctness of BD has been proved [24], the correctness of the new algorithm depends on certain stronger connectivity properties of discrete surfaces. This aspect needs further investigation. We may recall here that BD was in clinical use for quite some time before its correctness was established. The new algorithm performed correctly on all (more than 100) complex medical and simpler test objects to which it was applied. We have been using the new algorithm for over two years in both technical as well as clinical research as related to medical 3D

imaging. We have applied the algorithm to well over 100 objects including simulated test objects and complex medical objects such as the skull, brain, spine, heart, and pelvis. The algorithm performed correctly as conjectured in this paper in every instance. In the case of simulated objects correctness was checked by comparing the surface output by the algorithm with the known surface of the object. In the case of medical objects, correctness was established either by comparing the surface with the surface output by BD, or by displaying the surface in a dynamic movie mode and visually inspecting the surface. In every instance, the volume enclosed by the surface was computed to make sure that the computed volume agrees with the known normal anatomic volume.

The organization of the paper is as follows: Section 2—our terminology and definitions, Section 3—the original surface-tracking algorithm, Section 4—the new algorithm, Section 5—implementational details, Section 6—results and discussion, Section 7—conclusions.

2. TERMINOLOGY AND DEFINITIONS

We assume that the 3-dimensional space is partitioned into identical cubes by equally spaced planes perpendicular to the three coordinate axes x, y, z. We call the closed set of points in the 3-dimensional space that represents each such cube a **voxel**. Note that a voxel consists of all points interior to the cube as well as points forming faces, edges, and vertices of the cube.

We define a binary scene $\mathcal{S}$ as a tuple (V, g), where V is a set of voxels

$$V = \{v / v = (v_1, v_2, v_3), \text{ where, for } 1 \leq l \leq 3,$$
$$v_l \text{ is an integer, and } 0 \leq v_l \leq b_l\},$$

b_l being the dimension of the scene in direction l, and g is a mapping that assigns a value 0 or 1 to each voxel in V. We refer to those voxels to which the value 1 is assigned as 1-voxels, and to those to which the value 0 is assigned as 0-voxels. We assume that those voxels v for which either $v_l = 0$ or $v_l = b_l$ (for some $1 \leq l \leq 3$) are 0-voxels. If a given binary scene does not satisfy this condition we simply pad it all around with an extra layer of 0-voxels.

Now consider a single voxel. We assign three *directed* circuits to the voxel as shown in Fig. 1. Such an orientation of a voxel via lateral circuits was used in [24] to represent a surface as a directed graph. We use the circuits to define adjacency and connectivity of both voxels and their faces. Unlike in [24], we distinguish between the three circuits and call them R-, G-, and B-circuits. We call the faces and edges of the voxel through which an i-circuit passes the **i-faces** and **i-edges** of the voxel, where $i \in \{R, G, B\}$. Clearly, each circuit i passes through four i-faces and four i-edges of the voxel such that two of the four edges of each i-face are i-edges. Note that every face is both an i-face and a j-face for some i, $j \in \{R, G, B\}$, with $i \neq j$. An i-edge is not a j-edge if $j \neq i$.

Of course, complex objects are made up of many voxels. Imagine that every voxel in the binary scene (1- as well as 0-voxels) has three circuits assigned to it. The following notions of adjacency and connectivity of voxels define how voxels may be put together to form objects.

DEFINITION 1. Two voxels u and v are said to be **1$_i$-adjacent**, for $i \in \{R, G, B\}$, if their intersection is an i-edge of both u and v; they are said to be

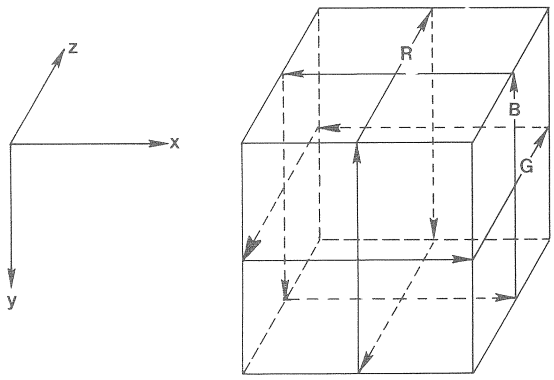

FIG. 1. Association of three directed circuits labelled R, G, B with a voxel. These circuits are used in the definition of adjacency and connectivity of voxels and their faces.

2_i-adjacent if their intersection is an i-face of both u and v. We also define the union of such adjacency relations. For example, two voxels are said to be **n_{ij}-adjacent**, for $1 \leq n \leq 2$ and some $i, j \in \{R, G, B\}$, if they are n_i-adjacent or n_j-adjacent.

Clearly, n_i-, n_{ii}- and n_{iii}-adjacency are all identical. If two voxels are n_{ijk}-adjacent, for $1 \leq n \leq 2$ and distinct $i, j, k \in \{R, G, B\}$, we also say that they are **n-adjacent**.

DEFINITION 2. Let **A** be a subset of V (we use $\mathcal{V} = (V, g)$ to denote a binary scene throughout this paper). Two voxels u and w are said to be **n_i-connected in A**, for $i \in \{R, G, B\}$, if $u = w$ or there exists a sequence of voxels $u_1 (= u)$, $u_2, \ldots, u_m (= w)$ all in A such that u_t is n_i-adjacent to u_{t+1}, for $1 \leq t < m$. We similarly define n_{ij}- and n_{ijk}-connectivity using n_{ij}- and n_{ijk}-adjacency, respectively. We call n_{ijk}-connectivity also **n-connectivity** (assuming i, j, k are distinct).

Clearly, n_i, n_{ij}-, and n_{ijk}-connectivity are equivalence relations. The partitions induced by them on **A** are called respectively **n_i-**, **n_{ij}-**, and **n_{ijk}-components** of **A**. We call the n_{ijk}-components of **A** also the **n-components** of **A**, when i, j, k are distinct.

Other approaches to the treatment of topological and geometric notions in 3- and higher dimensional discrete spaces exist; see, for example, [53, 36, 38]. The main departure here is that the adjacency and connectivity relations are made directionally sensitive. Clearly, there are numerous ways in which connected components can be defined as we have just demonstrated. Roughly speaking, objects whose surfaces we are interested in correspond to connected components. Though a given physical object can be described using a variety of different connectivity relations, some of these relations are better than others in the sense they lead to more efficient and less redundant tracking strategies.

Now we proceed to define adjacency and connectivity of boundary elements.

DEFINITION 3. A **boundary face** f in a binary scene $\mathcal{V}$ is the intersection of two voxels u and z, both in V, such that u is a 1-voxel, z is a 0-voxel, and u and z are 2-adjacent.

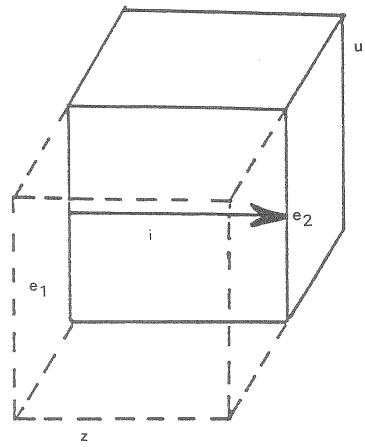

FIG. 2. The definition of in-edge and out-edge. The circuit labelled i, for some $i \in \{R, G, B\}$, emanates from edge e_1 and is incident upon edge e_2.

It is easily seen that f is an i-face of u as well as an i-face of z for some $i \in \{R, G, B\}$. For defining adjacency and connectivity of boundary faces we will make use of the fact that f is an i-face of u. Hence, to identify the 1-voxel among the two voxels that form boundary face f, we represent f by the ordered pair (u, z).

DEFINITION 4. Let $f = (u, z)$ be a boundary face of $\mathcal{V}$ and an i-face of u for some $i \in \{R, G, B\}$. Let e_1, e_2 be two i-edges of f such that the i-circuit passing through f goes from e_1 to e_2 (see Fig. 2). Then e_1 is called an **in-edge** of f and e_2 an **out-edge** of f with respect to circuit i.

The following notions of adjacency of boundary faces is fundamental to the topological foundations of BD and of the algorithm presented in this paper.

DEFINITION 5. Let $f_1 = (u, z)$ $f_2 = (u', z')$ be two boundary faces of $\mathcal{V}$, and $i \in \{R, G, B\}$. Then,

(a) f_1 is said to be **T_i-adjacent** to f_2, and denoted $f_1 T_i f_2$, iff:
  (1) f_1 and f_2 share a common i-edge e;
  (2) e is an out-edge of f_1 and an in-edge of f_2 with respect to i;
  (3) exactly one of the following cases holds (see Fig. 3):
    (3.1) $u = u'$ and the remaining voxels sharing e are 0-voxels (Fig. 3(a));
    (3.2) f_1, f_2 are in the same plane and u, u' are on the same side of this plane (Fig. 3(b));
    (3.3) $z = z'$ (Fig. 3(c)).

(b) f_1 is said to be **T_i*-adjacent** to f_2, and denoted $f_1 T_i^* f_2$, iff conditions (1) and (2) above hold, and
  (3) exactly one of the following cases holds (see Fig. 4):
    (3.1) $u = u'$ (Fig. 4(a));
    (3.2) same as (3.2) above (Fig. 4(b));
    (3.3) $z = z'$, and the fourth voxel sharing e is a 1-voxel (Fig. 4(c)).

We also define the union of such adjacency relations. For example, if f_1 and f_2 are two boundary faces, then we say that $f_1 T_{ij} f_2$, for some i, $j \in \{R, G, B\}$ if

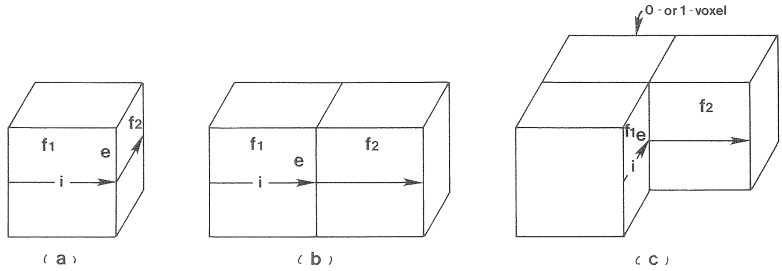

FIG. 3. The definition of T_i-adjacency. f_1 is said to be T_i-adjacent to f_2 if exactly one of the cases in (a), (b), (c) holds.

$f_1T_if_2$ or $f_1T_jf_2$. In particular, if $f_1T_{ijk}f_2$ for distinct $i, j, k \in \{R, G, B\}$, then we say that f_1Tf_2. We similarly define T_{ij}^*, T_{ijk}^*, and T^*.

Our notion of T-adjacency is similar to that of **P**-adjacency of [24]. An important difference is that we think of T-adjacency as having three components each of which defines adjacency along the R-, G-, and B-circuits, respectively.

DEFINITION 6. Let $\mathcal{V} = (V, g)$ be a binary scene, U be the set of all 1-voxels in V and $Z = V - U$. The **boundary** $\delta(U, Z)$ between U and Z is defined to be the set of all boundary faces of $\mathcal{V}$.

We are now ready to precisely state the surface tracking problem: **given** a binary scene $\mathcal{V}$ and a boundary face f_0 in $\delta(U, Z)$, to **track** an appropriate connected component of $\delta(U, Z)$ that contains f_0. The main difference between the surface tracking approach of BD and that presented here is in the definition of the connectivity relation on $\delta(U, Z)$. The two connectivity relations are defined using T_i- and T_i^*-adjacency, respectively. In the following sections, we formulate this difference.

3. DESCRIPTION OF ALGORITHM BD

DEFINITION 7. Let $\mathcal{T}$ be the reflexive, symmetric, and transitive closure of the T-adjacency relation on $\delta(U, Z)$. We say that a set F of boundary faces is **T-connected** iff for any two elements f, f' of F, $(f, f') \in \mathcal{T}$. A set E of boundary faces is said to be a **T-component** of a set F of boundary faces iff the following conditions are satisfied: (i) E is a subset of F; (ii) E is T-connected; (iii) E is not a

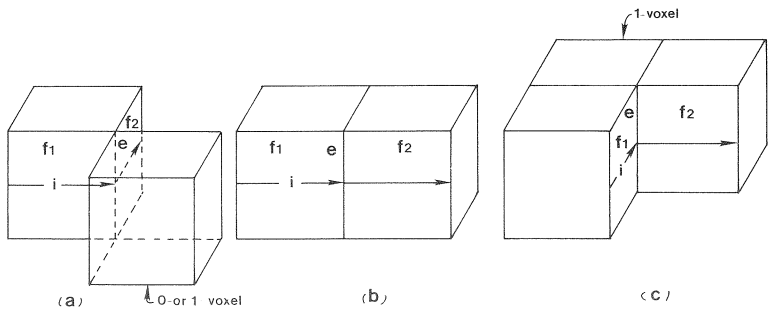

FIG. 4. The definition of T_i^*-adjacency. f_1 is said to be T_i^*-adjacent to f_2 if exactly one of the cases in (a), (b), (c) holds.

proper subset of any set satisfying (i) and (ii) (i.e., E is an equivalence class of the relation $\mathcal{T}$).

DEFINITION 8. Let $\mathcal{V} = (V, g)$ be a binary scene, U the set of all 1-voxels in V, $Z = V - U$, A_1 a 1-component of U, and Z_0 a 2-component of Z. Then $S = \delta(A_1, Z_0)$ is called **a surface of A_1 visible from Z_0**. BD tracks surfaces such as S. In the remainder of this section we state the two main results derived in [24] and outline Algorithm BD.

RESULT 1. Let $\mathcal{V}$, U, Z, A_1, Z_0, and S be as in Definition 8. Then S is a T-component of $\delta(U, Z)$.

This important result is stated and proved as Theorem 1 in [24]. It is shown that for every boundary face $f \in \delta(U, Z)$ (i) there are exactly two boundary faces $f_1, f_2 \in \delta(U, Z)$ such that fTf_l, for $1 \leq l \leq 2$, and (ii) there are exactly two boundary faces $f_1', f_2' \in \delta(U, z)$ such that $f_l'Tf$, for $1 \leq l \leq 2$. Further $\delta(U, Z)$ is represented as a digraph in which a boundary face is a node and T-adjacency represents an arc of the digraph. It is also shown that S corresponds to a component of such a digraph, and that this component has a binary spanning tree rooted at any given node. Thus, the problem of tracking S that contains a given boundary face f_0 is translated into a problem of traversing a binary spanning tree rooted at the node corresponding to f_0.

Given a binary scene and an initial boundary face f_0, BD proceeds to the two faces that are T-adjacent to f_0. From every face discovered in this manner, the algorithm proceeds to two new T-adjacent faces. Since for each face f there are two faces that are T-adjacent to f, it is reached twice during tracking. A "marking" mechanism is used to keep track of the faces that are discovered for the first time. When a face is reached the second time it is "unmarked."

INPUT: A binary scene $\mathcal{V}$ and a boundary face f_0.
(Usually f_0 is specified interactively by pointing to a location close to the boundary surface on the display of a slice of the binary scene.)
OUTPUT: A list L of faces in that T-component of $\delta(U, Z)$ that contains f_0.
DATA STRUCTURES: A queue Q that contains faces to be processed and a list M of marked faces.

ALGORITHM BD.

1. queue f_0 and put two copies of f_0 in M;
2. *while* Q is not empty
   a. remove a face f from Q;
   b. find f_l, for $1 \leq l \leq 2$, such that fTf_l;
   c. output f;
   d. *for* $l \leftarrow 1$ to 2 *do*
      d1. *if* $f_l \in M$ *then* delete f_l from M;
      d2. *else* queue f_l and put f_l in M;
      end for;
   end while ;
end BD

The starting face f_0 is different from other faces in the sense that, every face except f_0 is already visited once at the time the face is removed from Q in step 2a. Whereas, f_0 at this time still remains to be visited. This is why we need two copies of f_0 in M initially.

RESULT 2. Algorithm BD terminates, and at that time L contains precisely one copy of each face in that T-component of $\delta(U, Z)$ which contains f_0 (Theorem 2 in [24]).

There are numerous details that relate to the implementation of BD in the minicomputer environment. Those that are relevant to the new algorithm will be addressed in Section 5. For further details see [14].

4. THE NEW SURFACE TRACKING ALGORITHM

The new algorithm is based on the finding that one may completely ignore one of the three circuits and still be able to track all the faces of a connected surface, provided the notions of surface and connectivity of boundary faces are appropriately redefined.

Imagine then that each voxel in V has two circuits assigned to it. Note that, for the single voxel of Fig. 1, each circuit associated with the voxel passes through four of the six faces. By ignoring one circuit, we are left with only two faces that have two circuits going through them; each of the rest of the four faces has only one circuit passing through it. In any binary scene $\mathscr{V}$, hence, there are two types of boundary faces. In the first category, each boundary face $f = (u, z)$ is such that (i) f is an i-face of u for some $i \in \{R, G, B\}$ and (ii) f is not a j-face of u for any $j \in \{R, G, B\}$ unless $j = i$. We call such faces **type-1 boundary faces**. In the second category, the faces are such that (i) f is both an i-face and a j-face of u for some i, $j \in \{R, G, B\}$ and $i \neq j$ and (ii) f is not a k-face of u for any $k \in \{R, G, B\}$ unless $k = i$ or $k = j$. We call such faces **type-2 boundary faces**. It is easily seen that every boundary face in $\mathscr{V}$ is either of type 1 or of type 2. We may further note that, on an average, just one-third of the boundary faces in $\mathscr{V}$ are of type 2.

The following definitions of T_{ij}^*-connectivity and T_{ij}^*-component take into account the above modification using the T_i^*-adjacency defined in Section 2.

DEFINITION 9. Let $k \in \{R, G, B\}$ be the ignored circuit, and $\mathscr{T}_{ij}^*$ be the symmetric, reflexive, and transitive closure of the T_{ij}^*-adjacency relation on $\delta(U, Z)$ with $i, j \in \{R, G, B\} - \{k\}$ and $i \neq j$. We say that a set F of boundary faces is **$\mathbf{T_{ij}^*}$-connected** iff, for any two elements f, f' of $F, (f, f') \in \mathscr{T}_{ij}^*$. A set E of boundary faces is said to be a **$\mathbf{T_{ij}^*}$-component** of a set F of boundary faces iff the following conditions are satisfied: (i) E is a subset of F; (ii) E is T_{ij}^*-connected; (iii) E is not a proper subset of any set satisfying (i) and (ii) (i.e., E is an equivalence class of the relation $\mathscr{T}_{ij}^*$).

DEFINITION 10. Let $\mathscr{V} = (V, g)$ be a binary scene, U the set of all 1-voxels in V, $Z = V - U$, A_0 a 2-component of U, and k the ignored circuit. Let Z' be a 1_{ij}-component of Z, where $i, j \in \{R, G, B\} - \{k\}$ and $i \neq j$. Then, $S' = \delta(A_0, Z')$ is called a **surface of $\mathbf{A_0}$ visible from $\mathbf{Z'}$**. The new algorithm, which we call NBD (abbreviation for new BD), tracks surfaces such as S'.

An example is shown in Fig. 5 to illustrate the difference between the surfaces S (defined in Section 3) and S'. Here, V consists of a $5 \times 5 \times 5$ array of voxels, and U consists of all the voxels in a $3 \times 3 \times 3$ array (situated centrally in the $5 \times 5 \times 5$ array) except the central voxel and a voxel on an edge of the $3 \times 3 \times 3$ array as shown.

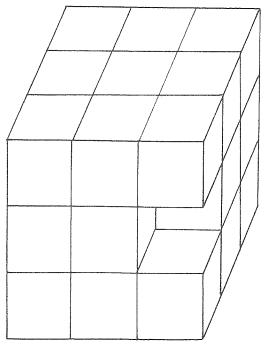

FIG. 5. An example to illustrate the differences between surfaces S and S'.

Clearly, U has only one 1-component, namely U itself; Z has two 2-components —the first (Z_0^1) consists of the single 0-voxel at the center of the $3 \times 3 \times 3$ array, and the other (Z_0^2) consists of all the remaining 0-voxels of Z. Hence, according to the definition of S, there are two surfaces, S_1 and S_2 of U—the first (consisting of 6 boundary faces) visible from Z_0^1, and the other (consisting of 56 boundary faces) visible from Z_0^2. Clearly, S_1 and S_2 are T-components of $\delta(U, Z)$.

Further, U has only one 2-component, namely U itself. Assuming that we ignore circuit G, it is easily seen that Z has two 1_{RB}-components—the first (Z_1') consists of the central 0-voxel and the other (Z_2') consists of the remaining 0-voxels of Z. By the definition of S', there are two surfaces S_1' and S_2' of U—the first visible from Z_1' and the other visible from Z_2'. Clearly, $Z_1' = Z_0^1$, $Z_2' = Z_0^2$, $S_1 = S_1'$, and $S_2 = S_2'$. If, however, we ignore either circuit R or circuit B, Z will have only one 1_{ij}-component (since the central 0-voxel now gets connected to the remaining 0-voxels of Z). Consequently, there will be only one surface (consisting of 62 boundary faces) of U, and $S' = \delta(U, Z)$.

ASSERTION 1. For every type-t ($t = 1$ or 2) boundary face in a binary scene $\mathscr{S}$ there are exactly $2t$ boundary faces f_l, f_l' in $\mathscr{S}$, for $1 \leq l \leq t$, such that $fT_{ij}{}^*f_l$ and $f_l'T_{ij}{}^*f$, where i, j are the two non-ignored circuits.

Proof. It suffices to prove the assertion for type-1 faces. The proof for type-2 faces is similar.

Let $f = (u, v)$ be a type-1 boundary face in $\mathscr{S}$. Clearly f has only one out edge. There are exactly four voxels u, v, v', and v'' that share the out-edge of f, of which u is a 1-voxel and v is a 0-voxel. From Definition 5 and considering the four possible combinations of the values that can be assigned to v' and v'', it follows that there is only one boundary face f_1 in $\mathscr{S}$ that shares the out-edge of f such that $fT_i{}^*f_1$. Similarly, there is only one boundary face f_1' in $\mathscr{S}$ that shares the in-edge of f such that $f_1'T_i{}^*f$. □

The following conjecture is the backbone of NBD; its validity is essential for NBD to produce the expected output. Though we do not have a formal proof of its validity, it has been found to hold good on all the objects—simple artificial objects as well as complex medical objects—to which NBD has been applied to date.

CONJECTURE 1. Let $\mathcal{V}$, U, Z, A_0, Z', and S' be as in Definition 10. Then S' is a T_{ij}^*-component of $\delta(U, Z)$.

The input and data structures for NBD are the same as for BD. The algorithm assumes that the initial face f_0 is of type 2 (if f_0 is of type 1, the implemented program follows the circuit passing through f_0 until a type-2 face is reached which is considered to be the real f_0).

INPUT: A binary scene $\mathcal{V}$ and a type-2 boundary face f_0.

OUTPUT: A list L of faces in that T_{ij}^*-component of $\delta(U, Z)$ that contains f_0.

DATA STRUCTURES: A queue Q that contains faces to be processed and a list M of marked faces.

ALGORITHM NBD.

1. queue f_0 and put two copies of f_0 in M;
2. *while* Q is not empty *do*
   a. remove a face f from Q and find its type t (1 or 2);
   b. find f_l, for $1 \le l \le t$, such that $fT_{ij}^*f_l$;
   c. output f;
   d. *for* $l \leftarrow 1$ to t *do*
      d1. *if* f_l is of type 1 *then* queue f_l
          else { f_l is of type 2—same as BD}
      d2. *if* $f_l \in M$ *then* delete f_l from M
      d3. *else* queue f_l and put f_l in M;
      end for;
   end while;
end NBD

(As described under Algorithm BD, in step 2a, since f_0 is in a situation different from the other type-2 faces, initially we need two copies of f_0 in M. The type t of a face is easily determined by table lookup since we know which of the three circuits was ignored.)

THEOREM 1. Algorithm NBD terminates.

Proof. Assume otherwise, and let $f_1, f_2, \ldots, f_n, \ldots$ be the infinite sequence of boundary faces output by NBD in step 2c. Since there are a finite number of 1-voxels in $\mathcal{V}$, at least one face appears more than once in the above sequence. Let m be the minimal index such that $f_m = f_l$ for some $1 \le l < m$. (f_m may be considered as the first face to be output twice.) Since a boundary face is output only after it is removed from Q, it follows that f_m was queued twice. There are two cases to consider.

Case 1. f_m is a type-1 boundary face. Let $i \in \{R, G, B\} - \{k\}$ be the circuit through f_m, k being the ignored circuit, and f be the boundary face such that $fT_i^*f_m$. By Assertion 1, there is exactly one such face in $\mathcal{V}$. Hence in both times when f_m was queued in step 2d, it was reached via f. This implies that before f_m was queued the second time, f was already output the second time in step 2c; i.e., f appears twice in the sequence $f_1, f_2, \ldots, f_{m-1}$. This contradicts the minimality of m.

Case 2. f_m is a type-2 boundary face. Let i, j be the two circuits through f_m (both different from k). By Assertion 1, there are exactly two boundary faces, f, f' such that $fT_i^*f_m$ and $f'T_j^*f_m$.

Claim. f_m was reached three times before it was queued for the second time. To prove the claim, assume that $f_m \neq f_0$ (f_0 is the input type-2 boundary face).

When f_m was reached the first time, it was placed in Q and in M. The second time it was reached, it was not queued because it was found in M, and it was removed from M. Therefore, for f_m to have been queued two times, it must have been reached three times.

If $f_m = f_0$, then f_m was initially queued and two copies of it were placed in M. Therefore, when f_m was reached the first two times it was not queued, and each time one copy of f_m was removed from M. So, for f_m to have been queued two times, it must have been reached three times.

It is obvious that f_m must have been reached twice either from f or from f'. Therefore, by arguments similar to those of Case 1, either f or f' was output twice before f_m was output the second time—contradiction. □

COROLLARY 1. When NBD terminates L contains precisely one copy of each boundary face in that T_{ij}^*-component of $\delta(U, Z)$ that contains f_0.

5. IMPLEMENTATIONAL CONSIDERATIONS

In this section we identify the major computational steps of Algorithm NBD, and describe how we implemented these steps on a 16-bit minicomputer of a medical imaging scanner (GE 8800 and 9800 CT scanners). We will concentrate only on those steps which had to be treated differently from those of Algorithm BD either because of expected improvements in performance or because of the difference in the nature of the two algorithms. We refer the reader to [14] for implementational details that are common to both algorithms.

Input

The input operation appears implicitly in Step 2b of NBD. For a given boundary face $f = (u, z)$, to find f_l such that $fT_{ij}^*f_l$, NBD has to determine if certain voxels in the neighborhood of u are 0- or 1-voxels. Since this step is very similar to the corresponding step in BD, we refer the reader to [14] for details.

Queueing and Dequeueing

Though the queueing (Steps $2d_1$, $2d_3$) and the dequeueing (Step 2a) operations are less expensive than other operations, the actual queueing discipline used has a major influence on the performance of NBD. Here we describe three queueing strategies that we implemented and discuss their merits and demerits. Our format of storing a face in Q is illustrated in Fig. 6. The queue–dequeue overhead consists of packing and unpacking the face descriptors in four bytes.

Last In First Out (LIFO). In this discipline, the most recently detected face becomes the next face to be processed. Clearly, using this strategy, if f is a type-1 boundary face, there is no need to queue (hence dequeue) f_l. If f is of type 2, one f_l has to be queued since there are two possible next faces. It is easily seen that, for LIFO, the queue–dequeue overhead is, on the average, about $\frac{1}{6}$ that when all faces are queued. However, there are two serious disadvantages of the LIFO discipline. First, since NBD using LIFO queueing tends to continually explore new territories in the surface, the queue (and the list) grows very quickly. If we do not use virtual memory for the queue (and the list), this causes queue (and list) overflow even for

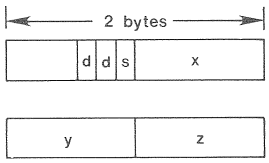

FIG. 6. A face is stored in Q using four bytes. Three bytes are used for the x-, y- and z-coordinates of the center of the face. The other byte (actually only three bits) is used for storing information about the direction of the face—the axis (d) to which the face is perpendicular in two bits and the sign of the axis (s) in one bit.

objects of moderate size. Second, because of the continual exploration of new regions, NBD constantly requires new segments of the input to be paged in (this is because the computer of the imaging scanner can store only a small fraction of the input binary scene), thus causing high input overhead. Hence, we have given up the LIFO discipline.

First In First Out (FIFO). In this strategy, the least recently detected face becomes the next face to be processed. Clearly, each detected face should be queued (hence dequeued) when FIFO queueing is used. Thus, the queue–dequeue overhead is about six times that using the LIFO discipline. However, since surface tracking using FIFO queueing tends to proceed in a "locally exhaustive" fashion, the problems of queue (and list) overflow and input overhead are far less severe. Similar observations have been made based on the implementation of BD reported in [14]. Because of its special nature, NBD permits an additional, more effective queueing discipline which is a hybrid of the LIFO and FIFO strategies.

Hybrid Queueing. In this discipline, we use the LIFO order if f is a type-1 face, else we use the FIFO strategy. What this means is that if f is of type 1, f_l is taken to be the next face. If f is of type 2, both f_ls are queued in the FIFO order. Though we have not investigated the theoretical aspects of the effect of queue discipline on maximum queue (and list) size and on input overhead, of the many disciplines we have implemented (including a variety of other hybrid techniques), we find this strategy to be the most effective.

Marking

For each type-2 boundary face f_l detected in Step 2b, NBD has to check the membership of f_l in the list M (Step $2d_2$) and either delete (Step $2d_2$) or insert (Step $2d_3$) f_l in M. These three operations done on M are common to BD also. The difference is that in NBD, the number of faces on which these operations are done is, on the average, about $\frac{1}{3}$ of those on which similar operations are done in BD. We have chosen to implement M as a hash table with a linked list mechanism to resolve collisions [29]. The same method was used in BD [14], but hashing is done differently and more effectively in NBD.

The hash function is built around the face descriptors (see Fig. 6)—the x, y, and z coordinates of the face and the parameter s (0 or 1) which indicates the direction of the face. Since all type-2 faces have the same value of d, it is not included in the

hash function. The actual function we use is given by

$$h(x, y, z, s) = x + y + z + (X + Y + Z)s,$$

where X, Y, Z are the maximum possible values of x, y, z, respectively. When a face f is reached, the value of h is computed and used as an address to the hash table. The content of the hash table at this address acts as a pointer to a chain of linked sublists which store faces with the same value for h. These sublists are explicitly searched to check if f belongs to M.

The motivation for choosing the above hash function is the following. For a fixed s, h assumes constant values on planes which are at 45° to the principal axes. Since it is uncommon for medical objects to have such planar surfaces, the hash function generates addresses that have a fairly uniform distribution. This ensures that no chain of sublists is unusually long, and hence, collision resolution is efficient.

Neighborhood Face Encoding

Our main objective of detecting a surface is to display it. In the computer graphics literature, many high quality surface shading algorithms have been reported, but these algorithms would be very time consuming if applied to the well over 200,000 faces produced by the surface tracking algorithms. We have developed an alternative approach [6] which works effectively on such surfaces. In this approach, the shade assigned to a face f is based on the orientation of four faces, in addition to the orientation of f. In BD, two of these four faces are the faces that are T-adjacent to f; the other two are faces to which f is T-adjacent. In NBD, these four faces are defined similarly by replacing T by T_{ij}^*. Since each of these four faces has three possible orientations with respect to f, the neighborhood face orientation information (with $3^4 = 81$ possibilities) can be stored in one byte. This information is appended to the face descriptor when the face is output.

TABLE 1
Input, Output, and Timing Description for BD and NBD

	Object 1	Object 2	Object 3
Size of the binary scene	160 × 255 × 198	251 × 255 × 144	223 × 237 × 239
BD No. of faces	297,212	352,280	741,538
Volume (in voxels)	1,213,265	825,564	5,482,979
Time	15.0 min	18.5 min	43.1 min
NBD No. of faces	298,106	352,226	740,926
Volume (in voxels)	1,212,733	825,554	5,482,612
Time	9.5 min	12.7 min	28.1 min
NBD (no neighborhood face encoding)	6.6 min	8.6 min	19.7 min
NBD (no neighborhood face encoding, no output)	6.0 min	8.0 min	18.5 min

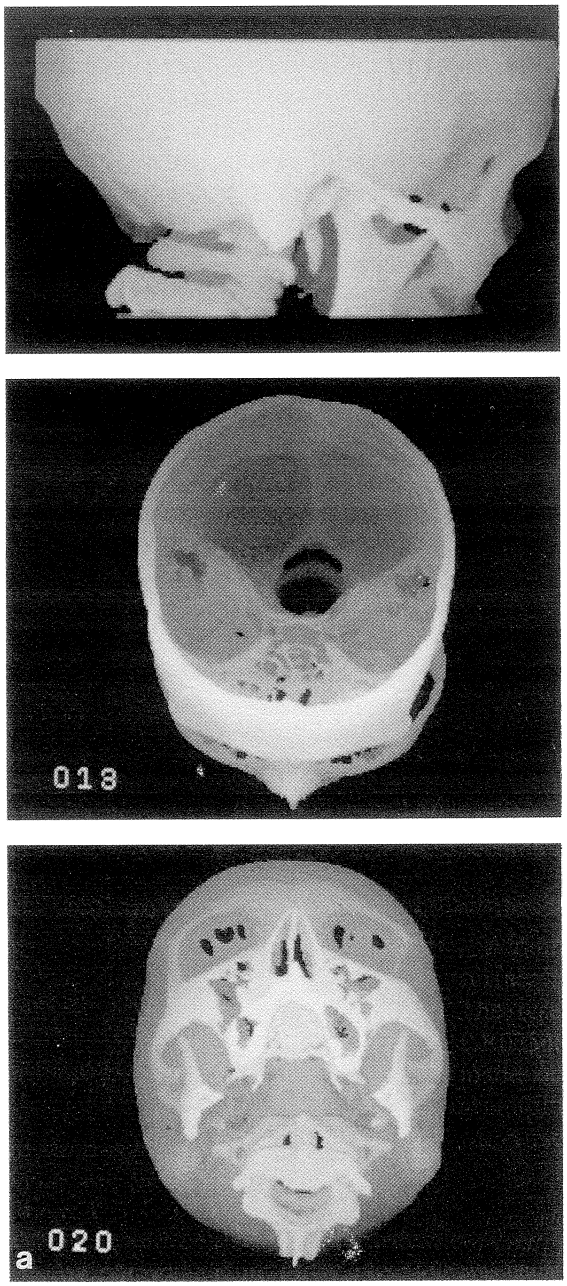

FIG. 7. Illustration of the displays generated from the surfaces detected by BD (images in (a)) and NBD (images in (b)). The input 3D binary scene was obtained by preprocessing (see [52]) a set of slice images produced by a GE CT/9800 scanner. The surface displayed in (a) consisted of 332,876 faces and enclosed 543,619 voxels. These figures for the surface in (b) are 334,780 faces and 541,908 voxels.

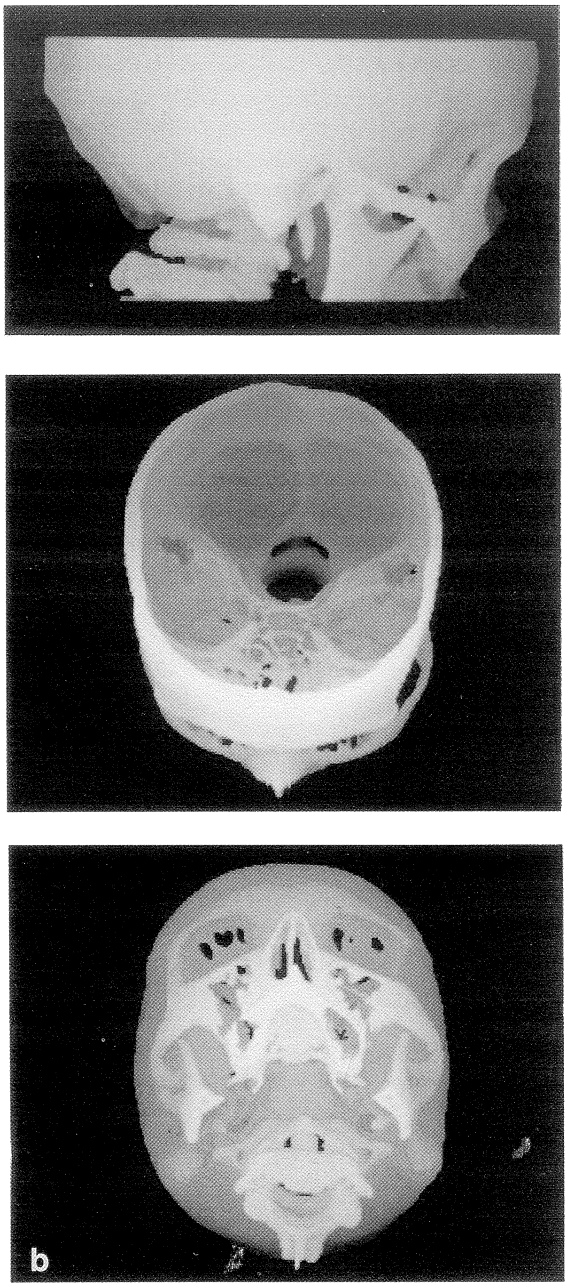

FIG. 7.—*Continued*

There are two important differences between BD and NBD from the point of view of neighborhood face encoding. First, because of the difference between T- and T_{ij}^{*}-adjacency, for a given face, BD and NBD may produce different neighborhood face orientation information. Consequently, the shading procedure will assign different shades to f based on the different neighborhood information. Thus, it is possible that for the same input to BD and NBD, the images generated from the output of BD and NBD are somewhat different. In our experience, this difference, in terms of the quality of image, has been insignificant (see next section for display examples). The second difference relates to the computational requirement. While determining f_1 and f_2 in step 2b of BD, we already know their orientation with respect to f. Of the remaining two face orientations we know one, because all faces (except f_0) removed from Q were put there in step $2d_2$ of BD. Thus, in BD, three neighbor codes result as a by-product of surface tracking, and only the fourth is calculated explicitly. In NBD, however, the neighbor code calculation turns out to be more expensive than that in BD. Clearly, for type-2 faces, this calculation is identical to that in BD. For a type-1 face f, since there is only one next face and since f is reached from only one face, two neighbor codes are available without requiring computation. The remaining two should be calculated explicitly. Thus, in NBD, on the average, $\frac{2}{3}$ of the faces need two neighbor code computations and $\frac{1}{3}$ need one computation. This seems to reduce the overall time saved by NBD as we report in the next section.

6. EXPERIMENTAL RESULTS AND DISCUSSION

We now report on comparative timings for BD and NBD. Table 1 summarizes the statistics for three medical objects for all of which the source of the data is X-ray computerized tomography. Object 1 consists of a part of the spinal column of a patient, object 2 is a dry skull specimen, and the human bony structure in the sacral region constitutes object 3.

The first row of the table specifies the size of the input binary scene. The second and the third row give the number of faces and volume (in terms of the number of voxels enclosed by a surface) for the surfaces output by BD and NBD. The timings given in these rows show that, for identical input and comparable output, NBD takes about 35% less execution time than does BD. Since surfaces S and S' are, by definition, somewhat different, for the same input, the surfaces output by BD and NBD are somewhat different as seen from the number of faces and volume computed by the algorithms. In practice, this difference does not really matter. One may also argue for and against the connectivity criterion used by the two algorithms, which happens to be the main harbinger of the difference in the nature of the outputs. We close this discussion by referring the reader to Fig. 7 which compares several views of the surfaces output by BD and NBD keeping input the same.

The significances of the further saving in time reported in rows 4 and 5 of the table are the following: First, there exist surface shading algorithms [17] which do not require neighborhood face orientation information. Second, there are many clinical applications [40, 45] wherein the objective is not the visualization of object surfaces but the computation of the volume enclosed by them [49]. For example, in our work on the application of 3D display to craniofacial surgery to correct orbital deformities, we routinely compute the volume of orbits preoperatively and at

successive stages postoperatively to assess the effectiveness of surgical procedures. In such applications, NBD can be used as a fast volume estimator.

A number of questions relating to the new algorithm still remain unresolved. First, though we are completely convinced about the correctness of Conjecture 1 it remains to be formally proved. Second, it is not clear how the choice of the circuit to be ignored influences the execution time. It may be argued that, because of the queueing, dequeueing, and marking overhead associated with the type-2 faces, the circuit to be ignored should be so chosen such that the resulting number of type-2 faces will be a minimum. In the examples considered in this sectin, this choice did not make any significant difference in the execution time. Third, there may be other queueing strategies that make a better trade-off between the queue, dequeue overhead, and the input overhead—for example, maintaining a miniqueue for each of a number of cubic subregions of the scene.

Finally, NBD may be considered to have resulted from reducing redundancy in BD. A logical continuation of this procedure should yield an algorithm that allows tracking the faces on a surface in a single path—i.e., by proceeding from each face only in one direction. Does such a path exist, and if so, is it possible to decide locally (as NBD does) as to where the next boundary face is?

7. CONCLUSIONS

We have arrived at a new definition of a discrete surface, and based on this definition, modified an existing state-of-the-art surface tracking algorithm. The new algorithm has been fully implemented in a clinical environment and has been found to yield an average saving of at least 35% of the execution time.

We believe that understanding the geometry and topology of 3D discrete surfaces becomes increasingly important as the applications of 3D display in medicine become more and more widespread.

ACKNOWLEDGMENTS

A number of people have contributed to our understanding of the problems in 3D surface tracking, particularly, Professors E. Artzy, G. Frieder, and G. T. Herman. The research reported in this paper is supported by NIH Grants HL28438 and RR02546. The visit of the first author to the Medical Image Processing Group was made possible by a grant from the South Australian Craniofacial Unit. We are grateful to Ms. M. A. Blue for typing, Mr. S. Strommer for photography, and to Ms. M. Kirsch for the drawings.

Note added in proof. Proof of Corollary 1: It suffices to show that if a face f is output by NBD, then so are f', f'', where $f'T_i{*}fT_i{*}f''$. f'' is obviously produced. To see why f' is produced, note that either f' was output before f, or else, by following circuit i from f, NBD eventually reaches f' from the other side.

REFERENCES

1. E. Artzy, G. Frieder, and G. T. Herman, The theory, design, implementation, and evaluation of a three-dimensional surface detection algorithm, *Comput. Graphics Image Process.* **15**, 1981, 1–24.
2. H. H. Atkinson, I, Gargantini, and M. V. S. Ramanath, Determination of the 3D border by repeated elimination of internal surfaces, *Computing* **32**, 1984, 279–295.
3. B. Baxter, Three-dimensional viewing device for examining internal structures, *SPIE Proc.* **283**, 1981, 111–115.

4. J. F. Brinkley, Knowledge-driven ultrasonic three-dimensional organ modeling, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-7**, 1985, 431–441.
5. R. L. Cook, L. Carpenter, and E. Catmull, The Reyes image rendering architecture, *Comput. Graphics* **21**, 1987, 92–102.
6. L. S. Chen, G. T. Herman, R. A. Reynolds, and J. K. Udupa, Surface rendering in the cuberille environment, *IEEE Comput. Graphics Appl.* **5**, 1985, 33–43.
7. L. T. Cook, S. J. Dwyer, III, S. Batnitzky, and K. R. Lee, A three-dimensional display system for diagnostic imaging applications, *IEEE Comput. Graphics Appl.* **3**, 1983, 13–19.
8. L. S. Chen, H. M. Hung, and J. K. Udupa, Towards real-time display of 3D medical objects, in *Proceedings, 18th Hawaii International Conference on System Sciences*, 1985, Vol. III, pp. 160–172.
9. P. Dev, S. Wood, J. P. Duncan, and D. N. White, An interactive graphics system for planning reconstructive surgery, in *Proceedings, National Computer Graphics Association Conference, Chicago, Il., June 1983*, pp. 130–135.
10. E. J. Farrel, R. Zappulla, and W. C. Yang, Color 3D imaging of normal and pathologic intracranial structures, *IEEE Comput. Graphics Appl.* **4**, 1984, 5–17.
11. G. Frieder, D. Gordon, and R. A. Reynolds, Back-to-front display of voxel-based objects, *IEEE Comput. Graphics Appl.* **5**, 1985, 52–60; Addendum **5**, 1985, 83.
12. H. Fuchs, Z. M. Kedem, and S. P. Uselton, Optimal surface reconstruction for planar contours, *Commun. ACM* **20**, 1977, 693–702.
13. H. Fuchs, J. Poulton, A. Paeth, and A. Bell, Developing pixel-planes: A smart memory-based raster graphics system, in *Proceedings*, 1982 *Conference on Advanced Research on VLSI, MIT, Cambridge*, 1982, pp. 137–146.
14. G. Frieder, G. T. Herman, C. R. Meyer, and J. K. Udupa, Large software problems for small computers: An example from medical imaging, *IEEE Software* **2**, 1985, 37–47.
15. C. J. Gibson, A new method for the three-dimensional display of tomography images, *Phys. Med. Biol.* **28**, 1983, 1153–1157.
16. D. Gordon, Boundary detection and display: Some informal research notes, typed notes, Department of Computer Studies, University of Haifa, July 1982.
17. D. Gordon and R. A. Reynolds, Image-space shading of three-dimensional objects, *Computer Vision Graphics Image Process.* **29**, 1985, 361–376.
18. J. F. Greenleaf, J. S. Tu, and E. H. Wood, Computer-generated three-dimensional oscilloscopic images and associated techniques for display and study of the spatial distribution of pulmonary blood flow, *IEEE Trans. Nucl. Sci.* **NS-17**, 1970, 353–359.
19. S. M. Goldwasser, R. A. Reynolds, T. Bapty, D. Baraff, J. Summers, D. Talton, and E. Walsh, A 3D physician's workstation with real-time performance, *IEEE Comput. Graphics Appl.* **5**, 1985, 44–57.
20. G. T. Herman, (Guest Ed.) *Proc. IEEE* **71** (special issue on computerized tomography), 1983.
21. G. T. Herman, and H. K. Liu, three-dimensional display of human organs from computed tomograms, *Comput. Graphics Image Process.* **9**, 1979, 1–29.
22 L. D. Harris, R. A. Robb, T. S. Yuen, and E. L. Ritman, Non-invasive numerical dissection and display of anatomic structure using computerized X-ray tomography, *SPIE Proc.* **152**, 1978, 10–18.
23. P. B. Heffernan and R. A. Robb, A new method for shaded surface display of biological and medical images, *IEEE Trans. Med. Imaging* **MI-4**, 1985, 26–38.
24. G. T. Herman and D. Webster, A topological proof of a surface tracking algorithm, *Comput. Vision Graphics Image Process.* **23**, 1983, 162–177.
25. D. G. Jasson and R. P. Kosowsky, Display of moving volumetric images, *SPIE Proc.* **507**, 1984, 82–92.
26. D. Jackel, The graphics PARCUM system: A 3D memory based computer architecture for processing and display of solid objects, *Comput. Graphics Forum* **4**, 1985, 21–32.
27. S. M. Jaffey and K. Dutta, Digital reconstruction methods for three-dimensional image visualization, *SPIE Proc.* **507**, 1984, 155–163.
28. A. Kaufman, Towards a 3-D graphics workstation, in *Advances in Graphics Hardware I* (W. Strasser, Ed.), pp. 17–26, Springer-Verlag, New York/Berlin, 1987.
29. D. E. Knuth, *Sorting and Searching, The Art of Computer Programming*, Vol. 3, Addison-Wesley, Reading, Ma, 1973.

30. H. K. Liu, Two- and three-dimensional boundary detection, *Comput. Graphics Image Process.* **6**, 1977, 123–134.
31. D. C. L. Lacey, Geometric modeling with image plane integral holography, *SPIE Proc.* **507**, 1984, 121–126.
32. D. Morgenthaler and A. Rosenfeld, Multi-dimensional edge detection by hypersurface fitting, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-3**, 1981, 482–486.
33. D. J. Meagher, Applying solids processing methods to medical planning, in *Proceedings, National Computer Graphics Association Conference, Dallas, TX*, 1985, vol. III, pp. 101–109.
34. J. C. Mazziotta and K. H. Huang, THREAD (three-dimensional reconstruction and display) with biomedical applications in neuron ultrastructure and computerized tomography. *Amer. Fed. Inform. Process. Soc.* **45**, 1976, 241–250.
35. P. H. Mills, H. Fuchs, and S. M. Pizer, High-speed interaction on a vibrating mirror 3D display, *SPIE Proc.* **507**, 1984, 93–101.
36. D. Morgenthaler and A. Rosenfeld, Surfaces in 3D digital images, *Inform. and Control* **51**, 1981, 227–247.
37. S. Nakamura, Three-dimensional digital display of ultrasonograms, *IEEE Comput. Graphics Appl.* **4**, 1984, 36–45.
38. A. Rosenfeld, Three-dimensional digital topology, *Inform. and Control* **50**, 1981, 119–127.
39. R. A. Reynolds, D. Gordon, and L. S. Chen, A dynamic screen technique for shaded graphics display of slice-represented objects, *Comput. Vision Graphics Image Process.* **38**, 1987, 275–298.
40. D. Roberts, J. Pettigrew, J. K. Udupa, and C. Ram, Three-dimensional imaging and display of the temporomandibular joint, *Oral Surg. Oral Med. Oral Pathol.* **58**, 1984, 461–474.
41. M. L. Rhodes, An algorithmic approach to controlling search in three-dimensional image data, in *SIGGRAPH '79 Proceedings, Chicago, IL*, 1979, *pp.* 134–142.
42. A. Sunguroff and D. Greenberg, Computer-generated images for medical applications, *Comput. Graphics* **12**, 1978, 196–202.
43. B. I. Soroka, *Understanding Objects from Slices: Extracting Generalized Cylinder Descriptions from Serial Sections*, Ph.D. dissertation, Computer and Information Science, University of Pennsylvania, Philadelphia, 1979.
44. M. Shantz, Surface definition for branching contour-defined objects, *Comput. Graphics* **15**, 1981, 242–259.
45. M. C. Stalnecker, L. A. Whitaker, J. K. Udupa, and J. A. Katowitz, Applications of three-dimensional imaging, in *ASPRS/PSEF/ASMS Annual Scientific Meeting, Kansas City, October 27–November 1, 1985.*
46. W. K. Smith, D. S. Schlusselberg, B. G. Cutler, and D. J. Woodward, Hierarchical database design for biological modeling, in *Proceedings, National Computer Graphics Association Conference, Chicago, IL, June 1983*, pp. 106–116.
47. S. S. Trivedi, Representation of three-dimensional binary scenes, in *Proceedings, National Computer Graphics Association Conference, Dallas, TX*, 1985, Vol. III, pp. 132–144.
48. J. K. Udupa, G. T. Herman, P. S. Margasahayam, L. S. Chen, and C. R. Meyer, 3D98: A turnkey system for the display and analysis of 3D medical objects, *SPIE Proc.* **671**, 1986, 154–168.
49. J. K. Udupa, Determination of 3D shape parameters from boundary information, *Comput. Graphics Image Process.* **17**, 1981, 52–59.
50. J. K. Udupa, Interactive segmentation and boundary surface formation for 3D digital images, *Comput. Graphics Image Process.* **18**, 1982, 213–235.
51. J. K. Udupa, 3D imaging in medicine, in *Conference Proceedings, Eighth Annual Conference and Exposition of the National Computer Graphics Association, NCGA /87, 1987*, Vol. II, pp. 73–104.
52. J. K. Udupa, *DISPLAY82—A System of Programs for the Display of Three-Dimensional Information in CT Data*, Technical Report MIPG67, Medical Image Processing Group, Department of Radiology, University of Pennsylvania, Philadelphia, 1983.
53. J. K. Udupa, S. N. Srihari, and G. T. Herman, Boundary detection in multidimensions, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-4**, 1982, 41–50.
54. M. W. Vannier, J. L. Marsh, and J. O. Warren, Three-dimensional computer graphics for craniofacial surgery planning and evaluation, *Comput. Graphics* **17**, 1983, 263–274.
55. S. W. Zucker and R. A. Hummel, An optimal three-dimensional edge operator, in *Proceedings, IEEE Computer Science Conference on Pattern Recognition and Image Processing*, 1978, pp. 162–168.