COMPUTER VISION AND IMAGE UNDERSTANDING
Vol. 63, No. 3, May, pp. 418–429, 1996

3D Digital Topology under Binary Transformation with Applications

P. K. SAHA

Electronics and Communication Sciences Unit, Indian Statistical Institute, 203 Barrackpur Trunk Road, Calcutta 700035, India

AND

B. B. CHAUDHURI

Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, 203 B.T. Road, Calcutta 700035, India

In this paper we study 3D digital topology under the transformation of an object point to a nonobject point and vice versa. As a result of such a transformation, an object component in the 3 × 3 × 3 neighborhood of the affected point may vanish or split into two or more components or more than one object components may merge into one. Also, cavities or tunnels in the 3 × 3 × 3 neighborhood may be destroyed or created. One of the goals of this paper is to develop an efficient algorithm (*topo_para*) to compute the change in the numbers of object components, tunnels and cavities in the 3 × 3 × 3 neighborhood of the transformed point. Another important contribution is the classification of different types of points (e.g., arc inner point, arc edge point, surface inner point, surface edge point) and detection of different types of junction points (e.g., junction between arcs, junction between surfaces and arcs, junction between surfaces) on the surface skeleton representation of a 3D digital image. Using these junction points it is possible to segment a 3D digital surface topologically into meaningful parts. Also, we describe an efficient algorithm for computing the Euler number of a 3D digital image using the topological parameters computed by *topo_para*.

I. INTRODUCTION

The study of digital topology [2, 5, 6, 10, 15, 16, 22] provides a sound mathematical basis for various image processing operations such as thinning [11, 17, 23, 24], surface recognition [7, 12, 14], and computation of the Euler number [1, 2, 25]. In this paper we consider 3D cubic grid to represent a three-dimensional (3D) binary digital image and concentrate on the effects of transforming an object point to a nonobject point or its inverse.

Object (black) points in an image may be grouped as a set of connected components. An object component may contain cavities and tunnels. A *cavity* is a 3D analog of hole in 2D signifying the existence of a component of nonobject (white) points surrounded by an object component. A *tunnel*, on the other hand, does not signify a new nonobject component. However, an object component contains a tunnel if it contains a solid handle or a hollow torus. An open-ended hollow cylinder also has a tunnel. In 2D there is no concept analogous to tunnel.

When an object point is transformed to a nonobject point, i.e., *deletion*, an object component may vanish or split into two or more object components. Similarly, cavities or tunnels of an object component may be destroyed or created by the transformation process. On the other hand when a nonobject point is converted to an object point, i.e., *addition*, two or more object components may join to form a single object component or a new object component may be created. Similarly, under such a transformation, cavities or tunnels may be created or destroyed. One of the goals of this paper is to develop an efficient algorithm (*topo_para*) to compute the change in the numbers of object components, tunnels, and cavities within the $3 \times 3 \times 3$ neighborhood of p when p is deleted. This study may be useful in many image processing operations, some of which are mentioned above. Another important contribution is the classification of different types of points (e.g., arc inner point, arc edge point, surface inner point, surface edge point) and detection of different types of junction points (e.g., junction between arcs, junction between surfaces and arcs, junction between surfaces) on the surface skeleton representation of a 3D digital image. Using these junction points it is possible to segment a 3D digital surface topologically into meaningful parts. Also, we describe an efficient algorithm for computing the Euler number of a 3D digital image using the topological parameters computed by *topo_para*.

In Section II we provide some useful definitions and

notations related to 3D digital topology. An efficient algorithm called *topo_para* for computing the topological changes in the numbers of object components, tunnels, and cavities within the $3 \times 3 \times 3$ neighborhood of the deleted point is developed in Section III. In Section IV we address the problems of classifying the points in a surface skeleton representation into edge points, inner points of a surface and arc, and different junction points. Segmentation of a 3D digital surface is described in Section IV. An algorithm to compute the Euler number of a 3D digital image using the topological parameters computed by *topo_para* is also described in Section IV. The results of application of the segmentation technique to several synthetically generated 3D objects are discussed in Section V.

II. GENERAL DEFINITIONS AND NOTATIONS

Here we present a few definitions on 3D digital topology frequently used in this paper. We consider 3D cubic grid [8] to represent a 3D digital image. In this paper, points refer to 3D digital grid points unless stated otherwise. We follow the conventional definition of *α-neighborhood* or *α-adjacency* of points, where $\alpha \in \{6, 18, 26\}$. Two nonempty sets of points S_1 and S_2 are said to be α-adjacent if at least one point of S_1 is α-adjacent to at least one point of S_2. Let S be a nonempty set of points. An *α-path* between two points p, q in S means a sequence of distinct points $p = p_0, p_1, \ldots, p_n = q$ of S such that p_i is α-adjacent to p_{i+1}, $0 \le i < n$. Two points $p, q \in S$ are *α-connected* in S if there exists an α-path from p to q in S. An *α-component* of S is a maximal subset of S where each pair of points is α-connected.

A 3D digital image $\mathcal{I}$ is defined as a quadruple $(\mathcal{Y}, \alpha, \beta, \mathcal{B})$. Here $\mathcal{Y}$ is the image space which is a set of all grid points (i, j, k), where i, j, k are integers and $i_{\min} \le i \le i_{\max}$; $j_{\min} \le j \le j_{\max}$; $k_{\min} \le k \le k_{\max}$. In other words, $\mathcal{Y}$ is the set of all 3D cubic grid points in a finite rectangular parallelepiped. $\mathcal{B}$ is defined as the set of black points in $\mathcal{I}$ and $\mathcal{Y} - \mathcal{B}$ is the set of white points. α-adjacency and β-adjacency are the adjacencies used for finding α-components and β-components in $\mathcal{B}$ and $\mathcal{Y} - \mathcal{B}$, respectively. Note that $\mathcal{Y} - \mathcal{B}$ denotes the set of white points in $\mathcal{I}$. In this paper we use 26-adjacency for black points and 6-adjacency for white points and call 26-components of $\mathcal{B}$ *black components* and 6-components of $\mathcal{Y} - \mathcal{B}$ *white components*. A point $p \in \mathcal{Y}$ is called an *interior point* of $\mathcal{Y}$ if $\mathcal{N}(p) \subset \mathcal{Y}$, otherwise p is called a *border point* of $\mathcal{Y}$. The set of all border points of $\mathcal{Y}$ is called the *border* of $\mathcal{Y}$ and is denoted as $\mathcal{Y}^*$. A *cavity* in $\mathcal{I}$ is a white component of $\mathcal{I}$ surrounded by a black component. According to our convention for $\mathcal{Y}$ a *cavity* is a white component of $\mathcal{I}$ containing no border point of $\mathcal{Y}$.

In the following discussions, $\mathcal{N}(p)$ is used to denote the set of 27 points in the $3 \times 3 \times 3$ neighborhood of the point p including p itself. The set of points of $\mathcal{N}(p)$ excluding p is denoted as $\mathcal{N}^*(p)$. Note that $\mathcal{N}^*(p)$ is the border of $\mathcal{N}(p)$. For a set of points S we define 26-*envelope* $\mathcal{E}(S)$ of S as follows:

$$\mathcal{E}(S) = \bigcup_{p \in S} \mathcal{N}(p) - S.$$

With respect to $\mathcal{N}(p)$, we classify the points according to their adjacency relations with p.

1. An *s-point* is 6-adjacent to p.
2. An *e-point* is 18-adjacent but not 6-adjacent to p.
3. A *v-point* is 26-adjacent but not 18-adjacent to p.

The nomenclature for the points of $\mathcal{N}(p)$ is explained in Fig. 1. In Fig. 1, *E*, *W*, *S*, *N*, *T*, and *B* denote *east*, *west*, *south*, *north*, *top*, and *bottom points*, respectively. Similarly, *TE* denotes *top-east point* and so on. An *E-surface* of $\mathcal{N}(p)$ is defined as a set {*E*, *SE*, *NE*, *TE*, *BE*, *TNE*, *TSE*, *BNE*, *BSE*} of exactly nine points. Five other surfaces of $\mathcal{N}(p)$, namely, *W-surface*, *S-surface*, *N-surface*, *T-surface*, and *B-surface* are defined in a similar way. Exactly one *s*-point belongs to each surface of $\mathcal{N}(p)$. A *NE-edge* of $\mathcal{N}(p)$ is defined as a set {*NE*, *TNE*, *BNE*} of exactly three points. Eleven other edges of $\mathcal{N}(p)$, namely, *SE-edge*, *TE-edge*, *BE-edge*, *NW-edge*, *SW-edge*, *TW-edge*, *BW-edge*, *TN-*

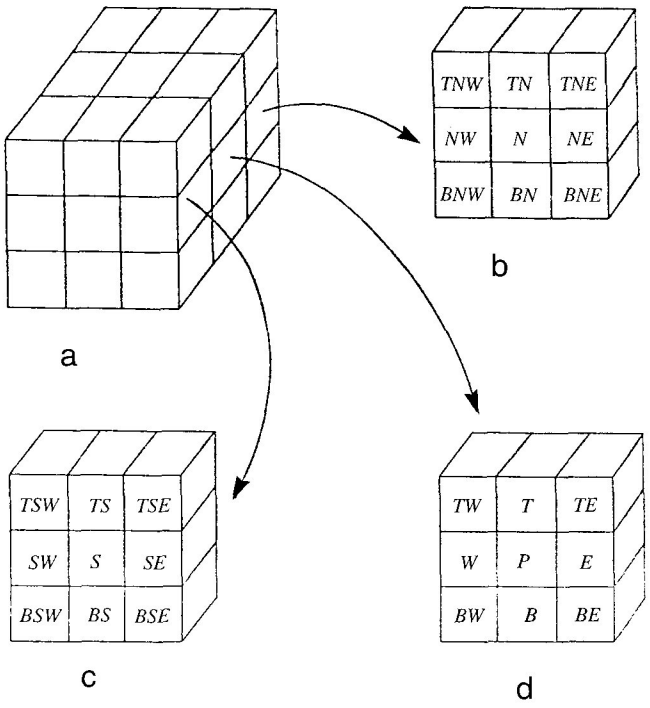

FIG. 1. $3 \times 3 \times 3$ neighborhood of a point p. (Clockwise from upper left: $3 \times 3 \times 3$ neighborhood; back plane; middle plane; front plane.)

edge, *BN-edge*, *TS-edge*, and *BS-edge* are defined in a similar way. Exactly one *e*-point belongs to each edge of $\mathcal{N}(p)$. Two *s*-points $a, b \in \mathcal{N}(p)$ are called *opposite* if they are not 26-adjacent, otherwise they are called *nonopposite* *s*-points.

III. THEORETICAL DISCUSSION

The concept of a simple point [6, 11, 18–20] is well established for both 2D and 3D image spaces and is useful in many binary image processing applications. The development of the simple point concept deals with the question of whether or not binary transformation of a point changes the topology of the image. However, in the present work, we compute the topological changes in terms of the changes in the numbers of black components, tunnels, and cavities within $\mathcal{N}(p)$. Let us consider two situations arising due to deletion of a point p: (1) creation of two black components, and (2) creation of a tunnel in $\mathcal{N}(p)$. Using a characterization of simple points we can detect that in both the situations the black points are nonsimple points. On the other hand, the present contribution distinguishes these two cases and efficiently computes the numbers of black components and tunnels generated in the $3 \times 3 \times 3$ neighborhood of the deleted point. This work facilitates the classification of points as edge points, inner points of arcs and surfaces, and different types of junction points in a surface skeleton for possible segmentation. Also, the study is applied to compute the Euler number of an object (see Section IV).

Let $\mathcal{P} = (\mathcal{V}, 26, 6, \mathcal{B})$ be an image under consideration. Then for every point p, we define two images in $\mathcal{N}(p)$ as follows:

$$\hat{\mathcal{N}}(p) = (\mathcal{N}(p), 26, 6, (\mathcal{N}(p) \cap \mathcal{B}) - \{p\})$$

$$\check{\mathcal{N}}(p) = (\mathcal{N}(p), 26, 6, (\mathcal{N}(p) \cap \mathcal{B}) \cup \{p\}).$$

Note that $\hat{\mathcal{N}}(p)$ and $\check{\mathcal{N}}(p)$ are 3D digital images with image space $\mathcal{N}(p)$ (the $3 \times 3 \times 3$ neighborhood of p). Moreover, p is always white in $\hat{\mathcal{N}}(p)$ (i.e., p is deleted), while p is always black in $\check{\mathcal{N}}(p)$ (i.e., p is added). For any other point of $\mathcal{N}(p)$, its color in $\hat{\mathcal{N}}(p)$ and $\check{\mathcal{N}}(p)$ is the same as the color of corresponding point in $\mathcal{P}$. Thus, our work boils down to the estimation of differences in the numbers of black components, tunnels, and cavities in $\hat{\mathcal{N}}(p)$ and $\check{\mathcal{N}}(p)$. The definitions of black components and cavities are well established. Kong and Rosenfeld [6] pointed out the difficulty of defining a tunnel. However, a precise definition of the number of tunnels can be provided. For example, the number of tunnels in a polyhedral set is the rank of its first homology group [9]. From Saha and Chaudhuri [20] we state the following result on $\hat{\mathcal{N}}(p)$:

1. $\hat{\mathcal{N}}(p)$ contains exactly one black component,
2. the number of tunnels in $\hat{\mathcal{N}}(p)$ is zero,
3. $\hat{\mathcal{N}}(p)$ contains no cavity.

For the definition of the number of tunnels in $\hat{\mathcal{N}}(p)$ we use some of the following notations:

$W_s(p)$ the set of white *s*-points in $\hat{\mathcal{N}}(p)$

$W_e(p)$ the set of white *e*-points in $\hat{\mathcal{N}}(p)$

$W_{se}(p)$ $W_s(p) \cup W_e(p)$.

DEFINITION 1. The number of tunnels in $\hat{\mathcal{N}}(p)$ is zero when all *s*-points are black, otherwise the number of tunnels in $\hat{\mathcal{N}}(p)$ is equal to one less than the number of 6-components of $W_{se}(p)$ that meet $w_s(p)$.

From the above definition of the number of tunnels in $\hat{\mathcal{N}}(p)$ it may be noted that $\hat{\mathcal{N}}(p)$ may contain at most five tunnels. From Definition 1 the corollary given below immediately follows.

COROLLARY 1. *The number of tunnels in $\hat{\mathcal{N}}(p)$ is independent of the color of the v-points.*

Here, we shall develop an efficient algorithm to compute the change in the numbers of black components, tunnels, and cavities in $\hat{\mathcal{N}}(p)$ that may occur due to binary transformation of p. As we have already mentioned that the numbers of black components, tunnels, and cavities in $\hat{\mathcal{N}}(p)$ are always 1, 0, and 0, respectively, we have to compute these numbers only in $\check{\mathcal{N}}(p)$. To do so we use three important properties of $\check{\mathcal{N}}(p)$.

Property 1. Let x be an *s*-point and $y \neq x$ be a point in the *x*-surface of $\check{\mathcal{N}}(p)$. Then a point $q \in \check{\mathcal{N}}(p)$ is 26-adjacent to x if it is 26-adjacent to y.

Property 2. Let x be an *e*-point and $y \neq x$ be a point in the *x*-edge of $\check{\mathcal{N}}(p)$. Then a point $q \in \check{\mathcal{N}}(p)$ is 26-adjacent to x if it is 26-adjacent to y.

Property 3. Let x be an *s*-point and y be an *e*-point in the *x*-surface of $\check{\mathcal{N}}(p)$. Then a 6-path of *s*- and *e*-points that joins two *s*-points must contain x if it contains y.

To verify the correctness of Property 3 the readers should note that in this paper paths are sequences of *distinct* points.

PROPOSITION 1. *If an s-point x is black in $\hat{\mathcal{N}}(p)$ then the number of black components in $\hat{\mathcal{N}}(p)$ is independent of the color of other points of the x-surface.*

PROPOSITION 2. *If an e-point x is black in $\hat{\mathcal{N}}(p)$ then the number of black components in $\hat{\mathcal{N}}(p)$ is independent of the color of other points of the x-edge.*

Proposition 1 and Proposition 2 may be proved using

Property 1 and Property 2, respectively. Also, the following proposition may be proved using Definition 1, Corollary 1, and Property 3.

PROPOSITION 3. *If an s-point x is black in $\hat{N}(p)$ then the number of tunnels in $\hat{N}(p)$ is independent of the color of other points of the x-surface.*

Let $\xi(p)$, $\eta(p)$, and $\delta(p)$ denote the numbers of black components, tunnels, and cavities in $\hat{N}(p)$, respectively. Since $\hat{N}(p)$ has only one interior point, it follows from the definition of cavity that $\hat{N}(p)$ may contain at most one cavity and it occurs when all s-points are black [20]. Thus,

$$\delta(p) = \begin{cases} 1 & \text{if six } s\text{-points are black;} \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, $\delta(p)$ is a function of s-point configuration. Moreover, as described in Proposition 1 and Proposition 3, $\xi(p)$ as well as $\eta(p)$ are independent of the color of the points of the x-surface when x is black. We define an x-surface as a *dead surface* of $\hat{N}(p)$ if the s-point x is black. A v-point or an e-point is called an *effective point* of $\hat{N}(p)$ if it does not belong to any dead surface.

COROLLARY 2. *With a known s-point configuration of $\hat{N}(p)$, we can compute $\xi(p)$, $\eta(p)$, and $\delta(p)$ from the effective point configuration of $\hat{N}(p)$.*

COROLLARY 3. *With a known s-point configuration of $\hat{N}(p)$, we can compute $\eta(p)$ from the effective e-point configuration of $\hat{N}(p)$.*

Corollary 2 is a straightforward consequence of Propositions 1 and 3 while Corollary 3 is a straightforward consequence of Corollary 1 and Proposition 3.

Using Proposition 2, Corollary 1, and the definition of $\delta(p)$ we see that $\xi(p)$, $\eta(p)$, $\delta(p)$ are independent of the color of the points of the x-edge when x is black. We define an x-edge as a *dead edge* of $\hat{N}(p)$ if the e-point x is black. A v-point is defined as an *isolated point* of $\hat{N}(p)$ if it neither belongs to any dead surface nor belongs to any dead edge.

COROLLARY 4. *Let y be a black isolated point of $\hat{N}(p)$. Then $\{y\}$ is a black component of $\hat{N}(p)$.*

COROLLARY 5. *Let $\hat{N}(p)$ contain six white s-points. Then the number of black components in $\hat{N}(p)$ is equal to the number of 26-components in $B_e(p)$ plus the number of black isolated points (where $B_e(p)$ is the set of black e-points in $\hat{N}(p)$).*

Corollary 4 follows from the definition of an isolated point while Corollary 5 is a straightforward consequence of Proposition 2 and Corollary 4.

Now we shall describe the possible geometric classes of s-point configurations of $\mathcal{N}(p)$. Two s-point configurations belong to the same geometric class iff one can be transformed to the other by some three-dimensional rotation in multiples of 90° about different axes (with p as origin). Possible geometric classes of s-point configurations and corresponding numbers of effective points (n_e) are as follows.

Class 0: Six s-points are black ($n_e = 0$).

Class 1: Five s-points are black ($n_e = 0$).

Class 2: Two pairs of opposite s-points are black ($n_e = 0$).

Class 3: One pair of opposite s-points and two other nonopposite s-points are black ($n_e = 1$).

Class 4. One pair of opposite s-points and another s-point are black ($n_e = 2$).

Class 5: Three nonopposite s-points are black ($n_e = 4$).

Class 6: One pair of opposite s-points is black ($n_e = 4$).

Class 7: Two nonopposite s-points are black ($n_e = 7$).

Class 8: One s-point is black ($n_e = 12$).

Class 9: No s-point is black ($n_e = 20$).

As discussed earlier, $\xi(p)$, $\eta(p)$, and $\delta(p)$ are functions of effective point configuration of $\hat{\mathcal{N}}(p)$. Thus, if $n_e = 0$, as in Classes 0–2, we can at once know $\xi(p)$, $\eta(p)$, and $\delta(p)$. In other cases, as in Classes 3–9, where $n_e > 0$, we can use a look_up_table. For a given s-point configuration there are 2^{n_e} possible effective point configurations. An effective point configuration can be thought of as an n_e-bit binary number. For example, consider a Class 5 situation with e_0, e_1, e_2, e_3 denoting the four effective points. Then a four-bit binary number is generated such that its ith bit denotes the color of e_i (i.e., 1 when e_i is black and 0 otherwise). For example, a four-bit binary number 1010 denotes an effective point configuration where e_0 and e_2 are white while e_1 and e_3 are black. For each such effective point configuration there is an entry in the look_up_table which contains the values of $\xi(p)$, $\eta(p)$, and $\delta(p)$. Since there are only six s-points in $\mathcal{N}(p)$, $\hat{\mathcal{N}}(p)$ can contain at most five tunnels and it occurs when all s-points are white and all e-points are black (see Definition 1). Also $\hat{\mathcal{N}}(p)$ can contain at most eight black components and it occurs only when all v-points are black and all other points are white. Moreover, for Classes 1–9, $\delta(p)$ is always zero. Thus, for each entry of the look_up_table we use one byte whose lower order four bits give the value of $\xi(p)$ and higher order four bits give $\eta(p)$.

Since any two different s-point configurations belonging to the same geometric class can be transformed to each other by some three-dimensional rotation in multiples of 90° about different axes (with p as origin), a single look_up_table can be used for different s-point configurations belonging to the same geometric class. Here we shall describe

how one look_up_table can be used by two different s-point configurations belonging to the same geometric class. Let an s-point configuration γ belong to Class i and a look_up_table is generated for γ with its effective points ordered as $e_0, e_1, \ldots, e_n$. Let us consider another s-point configuration γ' belonging to Class i. Since γ and γ' belong to the same geometric class, γ can be transformed to γ' by some three-dimensional rotation in multiples of 90° about different axes (with p as origin). Let us denote that rotation by μ. Then the same look_up_table can be used for γ' with its effective points ordered as $e_0', e_1', \ldots, e_n'$, where e_i', $0 \leq i \leq n$ is obtained from e_i after the rotation μ. For example, if a s-point configuration contains the black s-points N, T, E then we may have a look_up_table where its effective points are ordered as $e_0 = SW$, $e_1 = BW$, $e_2 = BS$, and $e_3 = BSW$. On the other hand, if another s-point configuration contains the black s-points N, B, E then the same look_up_table can be used if we make the effective points ordered as $e_0 = TW$, $e_1 = SW$, $e_2 = TS$, and $e_3 = TSW$. We denote the ordering of effective points for an s-point configuration γ as an ordered set $EFO(\gamma)$. Note that the s-point configuration with black s-points N, B, E as well as its ordered effective points can be obtained by rotating the s-point configuration with black s-points N, T, E and its ordered effective points about the x-axis by 90° in clockwise direction (with p as origin). As discussed above, only one look_up_table is necessary for each Class i, $3 \leq i \leq 9$. Let these tables be called LT_i, $3 \leq i \leq 9$. Thus, the total storage for the look_up_tables of Class i, $3 \leq i \leq 8$ is as follows (considering one byte for each entry).

$$2^1 + 2^2 + 2^4 + 2^4 + 2^7 + 2^{12} \text{ bytes} \simeq 4 \text{ Kbytes}$$

Unfortunately, the memory space required for the look_up_table of Class 9 is as high as 2^{20} bytes, i.e., 1 Mbytes. Hence, a straightforward use of look_up_table described above is not efficient for Class 9. To generate a different look_up_table for Class 9 that needs less space we use Corollaries 3 and 5. When all s-points are white, there may be two cases:

Case 1: All e-points are white.

Case 2: At least one e-point is black.

If Case 1 is true then $\eta(p) = 0$, $\delta(p) = 0$, the number of 26-components in $B_e(p) = 0$, and all v-points are isolated points. Thus, we can compute $\xi(p)$ by finding the number of black v-points.

If Case 2 is true, let us consider that an e-point x is black. Thus, there can be at most 2^{11} possible configurations of other e-points and hence the look_up_table LT_9 needs 2^{11} entries. An ordered set of these eleven e-points is used to calculate their configuration value. We denote this ordered set as $EEO(x)$. The address of an entry of LT_9 corresponds to a distinct configuration of these eleven e-points. Thus, to compute $\xi(p)$ and $\eta(p)$ (here, $\delta(p) = 0$), the following information is stored at each entry of LT_9.

1. the number of tunnels in $N_{18}(p)$,
2. the number of 26-components in $B_e(p)$,
3. the set of isolated points in $N_{18}(p)$.

Since, the e-point x is black, at most six v-points can be isolated points depending on the configuration of eleven other e-points (i.e., two v-points belonging to the x-edge are never isolated point). To denote the set of isolated points by a bit pattern we use an ordered set, say $ISO(x)$, of the above mentioned six v-points. Thus, each entry of LT_9 needs two bytes. The higher order four bits of the first byte contain the value of $\eta(p)$, while the lower order four bits of the same byte contain the number of 26-components in $B_e(p)$. The lower order six bits of the second byte denote the set of isolated points (sixth and seventh bits are always zero). For example, if the first and second bytes of certain entry of LT_9 are 00110010 and 00010001, then $\eta(p) = 3$ and the number of 26-components in $B_e(p)$ is 2, while zeroth and fourth points of $ISO(x)$ are isolated points.

The look_up_table LT_9 now needs only 2×2^{11} bytes, i.e., 4 Kbyte, instead of 1 Mbyte in its earlier form. Unlike LT_i, $3 \leq i \leq 8$, LT_9 does not contain the value of $\xi(p)$. During run time a one-byte word w is generated to denote the color of the points of $ISO(x)$. For example, 00101101 denotes that the zeroth, second, third, and fifth points of $ISO(x)$ are black. Thus a bitwise AND operation between w and the second byte of a corresponding entry of LT_9 gives the set of black isolated points in $N_{18}(p)$.

An e-point can be rotated in multiples of 90° about different axes (with p as origin) to reach another e-point. Thus, we can use single look_up_table for different e-points in the same fashion we described for different s-point configurations belonging to the same geometric class. However, here we need the rotational transformation on both $EEO(x)$ and $ISO(x)$.

III.A. *The Algorithm*

It is understood that our procedure for computing $\xi(p)$, $\eta(p)$, and $\delta(p)$ has two parts, which are: (1) *a priori knowledge*, and (2) *run-time computation*. The a priori knowledge includes the precalculation and storage of all the look_up_tables LT_i, $3 \leq i \leq 9$, in separate files. Also, for each s-point configuration γ belonging to Class i, $3 \leq i \leq 8$, the ordered set of effective points $EFO(\gamma)$ is precalculated as discussed earlier and implicitly stored in the program to compute the entry value of LT_i. Similarly, in the case of Class 9, for every e-point x, $EEO(x)$ and $ISO(x)$ are precalculated and implicitly used in the program. In addi-

tion, for all Class i, $0 \le i \le 2$, where the number of effective point is zero, $\xi(p)$, $\eta(p)$, and $\delta(p)$ are predetermined.

It should be mentioned that the a priori knowledge is independent of the input image to be processed. Thus, once generated, the knowledge can be used for any image later on.

During the run time, for any point p of an input 3D image $\mathcal{P}$, we can generate $\mathcal{N}(p)$ and use the following procedure (*topo_para*) to compute the topological parameters $\xi(p)$, $\eta(p)$, and $\delta(p)$.

```
procedure topo_para (N(p))
get the s-point configuration γ from N(p);
switch(geometric class of γ)
Case 1: /* γ belongs to Class 0 */
        ξ(p) = 1; η(p) = 0; δ(p) = 1;
Case 2: /* γ belongs to Class 1 */
        ξ(p) = 1; η(p) = 0; δ(p) = 0;
Case 3: /* γ belongs to Class 2 */
        ξ(p) = 1; η(p) = 1; δ(p) = 0;
Case 4: /* γ belongs to Class 3–8 */
        let γ belong to Class i, 3 ≤ i ≤ 8;
        get the configuration value of EFO(γ), say j;
        ξ(p) = value of lower four bits of jth entry of LT_i;
        η(p) = value of higher four bits of jth entry of LT_i;
        δ(p) = 0;
Case 5: /* γ belongs to Class 9 */
        if all e-points are white in N(p) then
            ξ(p) = number of black v-point;
            η(p) = 0;
            δ(p) = 0;
        else            /* at least one e-point is black */
            get a black e-point x;
            get the configuration value of EEO(x), say i;
            generate a one-byte word w to denote the con-
            figuration of ISO(x);
            r = result of bitwise AND operation between
            w and the second byte of ith entry of LT_9;
            ξ(p) = value of lower four bits of the first byte
            of ith entry of LT_9 + number of 1-valued bits
            in r;
            η(p) = value of higher four bits of the first
            byte of ith entry of LT_9;
            δ(p) = 0;
end procedure topo_para;
```

IV. APPLICATION

From the theoretical development described above, we get the expressions for $\xi(p)$, $\eta(p)$, and $\delta(p)$ corresponding to an object point p. These parameters are useful in different image processing techniques such as thinning, 3D surface segmentation, and Euler number computation. Here we shall describe their application to the classification of different types of points such as surface edge point, arc end point, surface or arc inner point, and different types of junction points in a three dimensional surface skeleton representation. The results of this classification method are applied for a meaningful segmentation of 3D surface. Finally, we describe an algorithm to compute the Euler number of a 3D digital object.

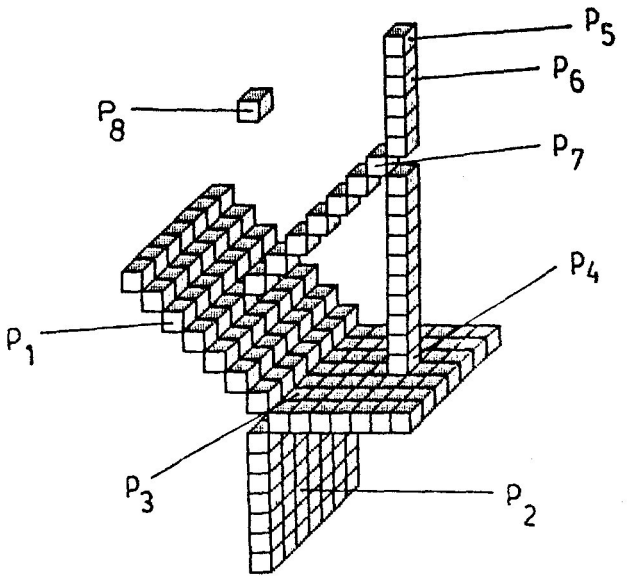

FIG. 2. Demonstration of different types of points in a surface skeleton representation (see the text for detail).

IV.A. *Point Type Classification*

In a 2D skeleton we can imagine only three types of points, namely, arc end point, arc inner point, and junction point of arcs. However, in a 3D surface skeleton, different types of junction points may occur (e.g., junction of surfaces, junction of surfaces and arcs, and junction of arcs). As an example, let us consider a surface skeleton representation shown in Fig. 2, where different points of importance are marked as p_i, $i = 1, 2, \ldots, 8$. These points, along with their $\xi(p)$, $\eta(p)$, and $\delta(p)$ values, are described below.

- p_1 is an edge point of surface (*SE*-type),

$$\xi(p_1) = 1,\ \eta(p_1) = 0,\ \delta(p_1) = 0;$$

- p_2 is an inner point of surface (*S*-type),

$$\xi(p_2) = 1,\ \eta(p_2) = 1,\ \delta(p_2) = 0;$$

- p_3 is a junction point of surfaces (*SS*-type),

$$\xi(p_3) = 1,\ \eta(p_3) = 2,\ \delta(p_3) = 0;$$

- p_4 is a junction point of surface and arcs (*SC*-type),

TABLE 1
Initial Decision Table for Skeleton Point Classification

$\xi(p)$	$\eta(p)$	$\delta(p)$	Point type	Name assigned
0	0	0	*I*-type	N_1
1	0	0	*SE*-type or *CE*-type	N_2
2	0	0	*C*-type	N_3
>2	0	0	*CC*-type	N_4
1	1	0	*S*-type or *CC*-type	N_5
>1	≥1	0	*SS*-type or *SC*-type or *CC*-type	N_6
1	>1	0	*SS*-type or *SC*-type or *CC*-type	N_7
1	0	1	*SS*-type or *SC*-type or *CC*-type	N_8

$$\xi(p_4) = 2,\ \eta(p_4) = 1,\ \delta(p_4) = 0;$$

- p_5 is an arc end point (*CE*-type),

$$\xi(p_5) = 1,\ \eta(p_5) = 0,\ \delta(p_5) = 0;$$

- p_6 is an inner point of arc (*C*-type),

$$\xi(p_6) = 2,\ \eta(p_6) = 0,\ \delta(p_6) = 0;$$

- p_7 is a junction point of arcs (*CC*-type),

$$\xi(p_7) = 3,\ \eta(p_7) = 0,\ \delta(p_7) = 0;$$

- p_8 is an isolated point (*I*-type),

$$\xi(p_8) = 0,\ \eta(p_8) = 0,\ \delta(p_8) = 0.$$

From the above example it may appear that $\delta(p)$ does not have any discriminating power. However, as seen from Table 1, there is a situation (last row of the table) where $\delta(p)$ takes a value 1 and hence discriminates against other cases. By analyzing all feasible combinations of $\xi(p)$, $\eta(p)$, and $\delta(p)$ we develop Table 1 for automatic detection of different types of points.

It can be found from Table 1 that N_1, N_3, N_4 signify unique point type while N_2, N_5, N_6, N_7, N_8 signify two or more point types. Thus, using Table 1 we can get a partial classification of skeleton points in one scan. However, from this partial classification we can arrive at unique classification using another scan. During this second scan we observe the 26-neighborhood of any point not uniquely decided by Table 1 and use Table 2 for unique classification. Thus, in Table 2 only N_2, N_5, N_6, N_7, N_8 points are considered.

At the end of second scan the classification process is completed except for a small problem illustrated by Figs. 3 and 4. In Figs. 3a and 3c the hidden points immediately below the vertical 6 × 6 rectangle are all white points. Also, in Figs. 3b and 3d the hidden points immediately below the vertical 5 × 6 rectangle are all white points. In Figs. 3a–3b, the *SS*-line (a 26-path of *SS*-type points) should be extended to reach surface edges as shown in Figs. 3c–3d. In Fig. 4, the arc-line (a 26-path of *C*-type or *CC*-type points) meets the surface and we have to find out the junction point between the arc-line and the surface. On the other hand, in another example shown in Fig. 5, the *SS*-line should not be extended at two ends. However, as discussed below, it is possible to identify these cases and act accordingly.

Extension of SS-lines. At first we define a function D of two points p, q as follows:

$$D(p, q) = \begin{cases} 0 & \text{if } p = q; \\ 1 & \text{if } p \text{ is an } s\text{-point of } N_{26}(q); \\ 2 & \text{if } p \text{ is an } e\text{-point of } N_{26}(q); \\ 3 & \text{if } p \text{ is a } v\text{-point of } N_{26}(q); \\ \infty & \text{if } p \notin N_{26}(q); \end{cases}$$

Let S denote a set of points. Using the above function we define q as one of the nearest points of p in S if $\forall r \in S$, $D(p, r) \geq D(p, q)$. It should be noted that a point may have more than one nearest point in a set of points.

Let S_{SE} denote the set of all *SE*-type points in a surface skeleton. Let p be an end point of a *SS*-line (an end point has at most one 26-adjacent *SS*-type point). Let SS_p denote the set of all *SS*-type points in $N_{26}(p)$ excluding p, and let S_p denote the set of all *S*-type points in $N_{26}(p)$ excluding p. We observe every *SE*-type point $q \in (N_{26}(p) - N_{26}(SS_p)) \cap S_{SE}$ and flag according to the following algorithm.

TABLE 2
Final Decision Table for Skeleton Point Classification

Name	Neighborhood analysis	Point type
N_2	Exactly one ASP	*CE*-type
N_2	More than one ASPs	*SE*-type
N_5	All ASPs are N_3 or N_4	*CC*-type
N_5	Not all ASPs are N_3 or N_4	*S*-type
N_6	All ASPs are N_3 or N_4	*CC*-type
N_6	Not all but some ASPs are N_3 or N_4	*SC*-type
N_6	No ASP is N_3 or N_4	*SS*-type
N_7	All ASPs are N_3 or N_4	*CC*-type
N_7	Not all but some ASPs are N_3 or N_4	*SC*-type
N_7	No ASP is N_3 or N_4	*SS*-type
N_8	All ASPs are N_3 or N_4	*CC*-type
N_8	Not all but some ASPs are N_3 or N_4	*SC*-type
N_8	No ASP is N_3 or N_4	*SS*-type

Note. In this table ASP is an abbreviation of "26-adjacent skeleton point."

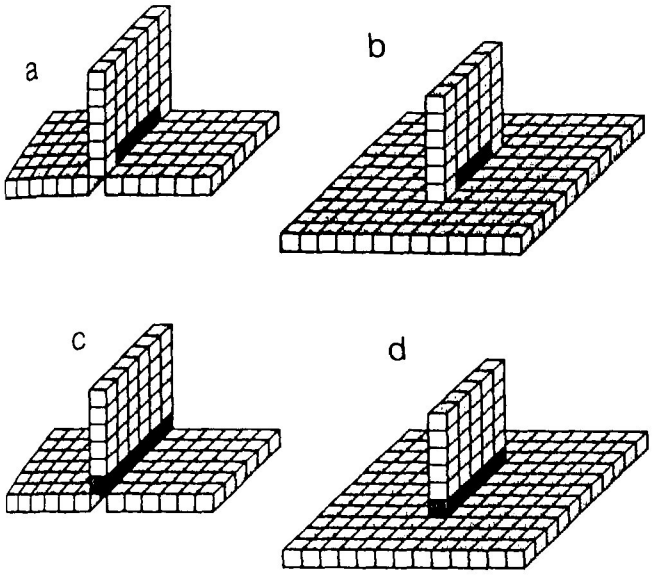

FIG. 3. Extension of SS-lines. (a, b) The SS-line (shown black) obtained using Table 2; (c, d) the SS after the extension process.

if $S_{SE} \cap \mathcal{N}^*(q)$ contains more than two 26-components then
    flag q;
else
if the number of tunnels in $S_{SE} \cap \mathcal{N}^*(q)$ is greater than zero then
    flag q;
else
if $S_{SE} \cap \mathcal{N}^*(q)$ contains less than two 26-components then
    if $\exists r \in S_p \cap \mathcal{N}^*(q) - \mathcal{C}(SS_p)$ such that $D(q, r) < D(q, p)$ then
        select one of the nearest points of q in $S_p \cap \mathcal{N}^*(q) - \mathcal{C}(SS_p)$, say t;
        flag t;

Note that after the last "else" statement the flagged point (if at all) is not q but a point t nearest to q in $S_p \cap \mathcal{N}^*(q) - \mathcal{C}(SS_p)$. Finally, all flagged points are renamed as SS-type points. If the algorithm is executed on Figs. 3a and 3b then we obtain Figs. 3c–3d.

Finding junctions between arc and surface. Let p be a C-type or CC-type point and let S be the set of S-type, SC-type, and SS-type points in $\mathcal{N}(p)$. Let $S_1, S_2, \ldots, S_n$ be the 26-components of S. For each S_i such that S_i contains

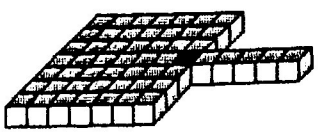

FIG. 4. The SC-type junction point (shown black) cannot be detected using Table 2.

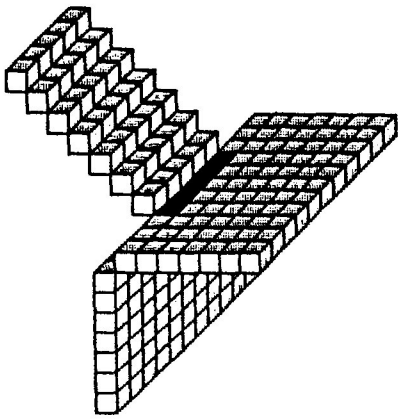

FIG. 5. An example where the extension process is not needed.

no SC-type or SS-type points, one of the nearest points of p in S_i is declared as a SC-type point. The SC-type junction point in Fig. 4 is detected by this algorithm.

IV.B. *Surface Skeleton Segmentation*

Here, we shall describe a method of segmentation of a surface skeleton representation based on the classification method described in Section IV.A. Let S denote the set of all skeleton points and let J denote the set of all SS-type, SC-type, CC-type points (i.e., all junction points). Let $S' = S - (\mathcal{E}(J) \cup J)$. The set of 26-components of S' represents different segmented surfaces and arcs of the suface skeleton. However, some undesired situations may occur for some surface representations. These situations along with their solutions are described below.

For a surface representation shown in Fig. 6a an undesired tunnel is created in S' as shown in Fig. 6b. Moreover, in Fig. 7a the tail-like part is removed as shown in Fig. 7b. To solve these problems we use two more steps.

Step 1. Let $S'_1, S'_2, \ldots, S'_n$ be all 26-components of S'. Each S'_i is extended to ES'_i as follows:

$$ES'_i = S'_i \cup (\mathcal{E}(S'_i) \cap S).$$

Step 2. Each extended component ES'_i is further extended to reach final components FS'_i as follows:

$$FS'_i = ES'_i \cup (\mathcal{E}(ES'_i) \cap J).$$

Finally all FS'_i's as well as all 26-components of $J - \cup_{i=0}^{n} FS'_i$ are declared as segmented parts. Figures 6c and 7c demonstrate the final outputs obtained using these steps.

IV.C. *Computation of the Euler Number*

Most of the existing methods for computing the Euler number of an object is based on computing the expression

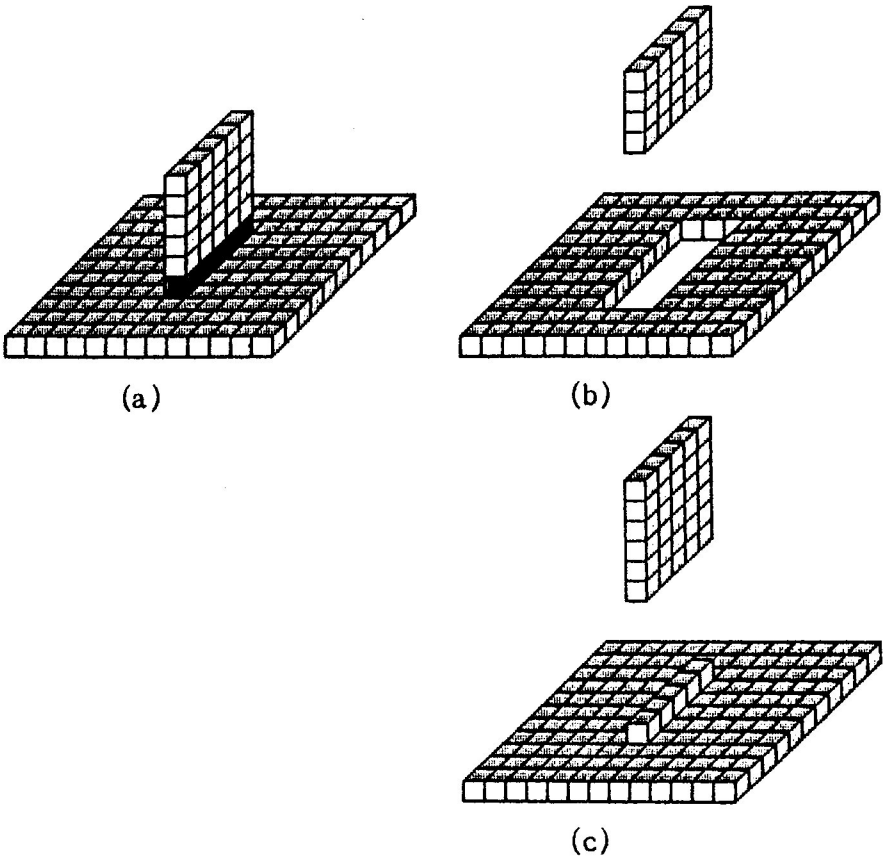

FIG. 6. An example where an undesired tunnel is created in S' (see the text). (a) Original surface representation with the junction points shown black; (b) 26 components of S'; (c) final segments.

$\Sigma(-1)^k * C_k$, where C_k is the number of k-dimensional simplex in an object. However, the Euler number of a 3D object is also equal to the number of object components minus the number of tunnels plus the number of cavities in the object [8]. We know that deletion of a point from an object may change the Euler number of the object. Also, the change in the Euler number of the object that may occur due to the deletion of a point is equal to the change in the Euler number of object points in the $3 \times 3 \times 3$ neighborhood of the deleted point.

Let us consider a 3D digital image $\mathcal{P} = (\mathcal{Y}, 26, 6, \mathcal{B})$ and we want to compute the Euler number of $\mathcal{P}$. Due to the deletion of a point $p \in \mathcal{B}$, the change in the Euler number in the $3 \times 3 \times 3$ neighborhood of p equals

the Euler number of $\mathcal{N}(p)$ − the Euler number of $\hat{\mathcal{N}}(p)$.

Again, the Euler number of $\mathcal{N}(p)$ = the number of black components in $\mathcal{N}(p)$ − the number of tunnels in $\mathcal{N}(p)$ + the number of cavities in $\mathcal{N}(p) = 1 - 0 + 0 = 1$. Thus, the change in the Euler number due to the deletion of $p = 1$ − the Euler number of $\hat{\mathcal{N}}(p) = 1 - \xi(p) + \eta(p) - \delta(p)$. The algorithm for computing the Euler number of $\mathcal{P}$ is as follows.

```
begin Euler_number(P = (Y, 26, 6, B))
E(P) = 0;
for each point p ∈ B do
    begin
        E(P) = E(P) + 1 − ξ(p) + η(p) − δ(p);
        B = B − {p};
    end
return(E(P));
end Euler_number.
```

Our method for computing the Euler number is locally defined and massive parallelization may be introduced using the concept of subfields [3].

V. RESULTS AND DISCUSSION

Segmentation done by the present approach on some images are illustrated in Figs. 8–10. Figures 8a–10a are three binary images displayed by 3D surface rendering.

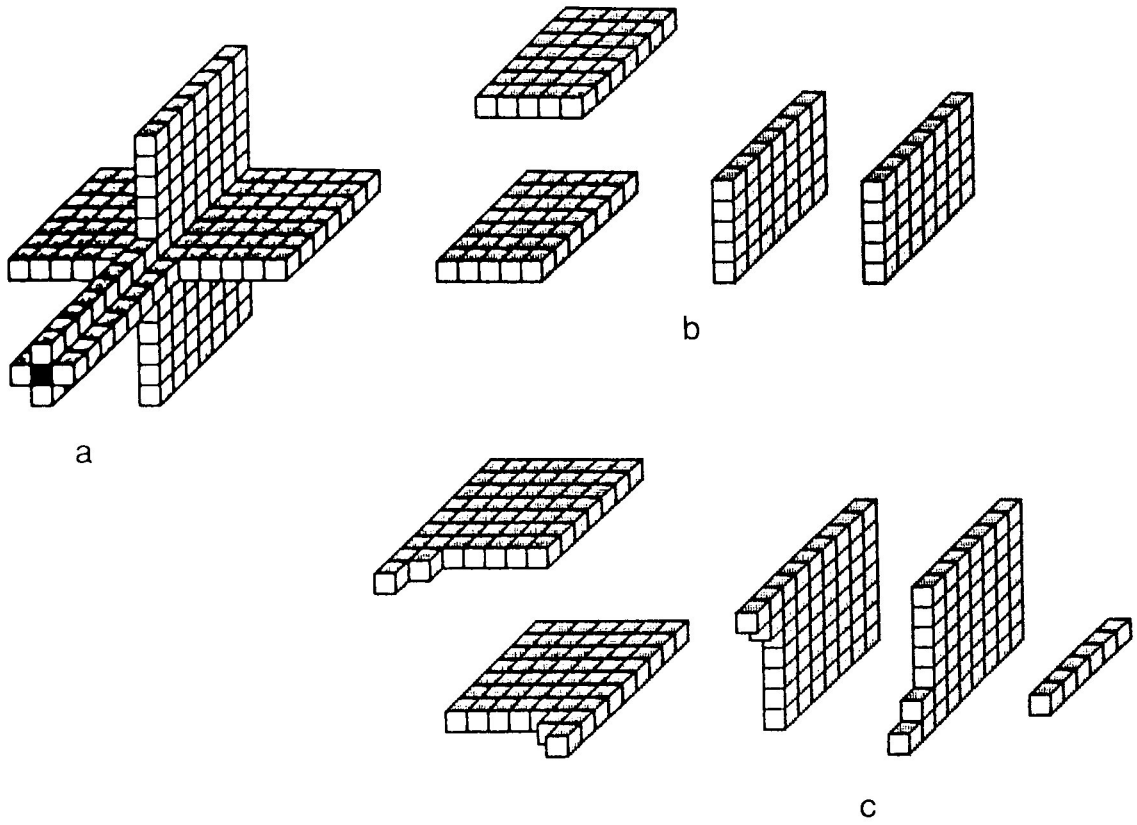

FIG. 7. An example where the tail-like part is lost in S'. (a) Original surface representation with the junction points shown black; (b) 26 components of S'; (c) final segments (the right-most segment represents the tail-like part).

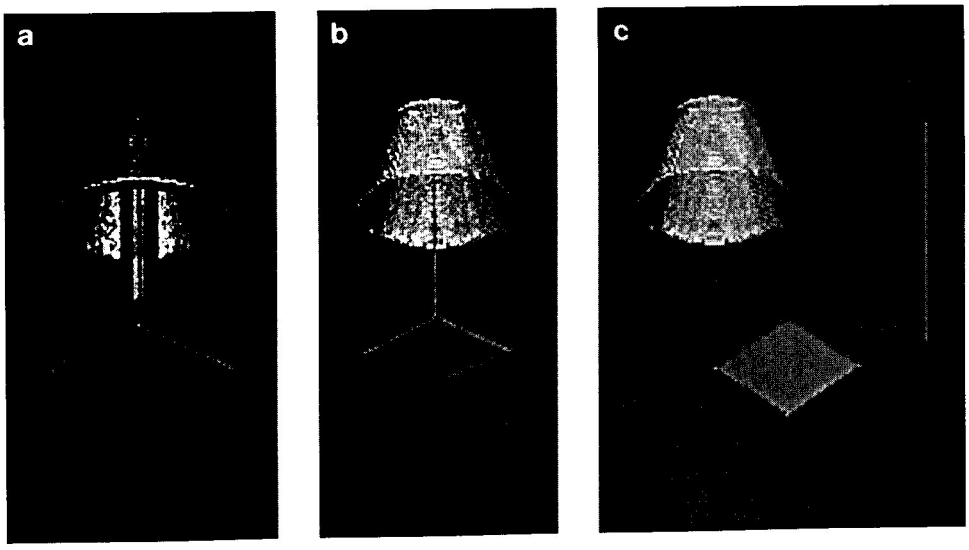

FIG. 8. Results of segmentation. (a) Original object; (b) surface skeleton representation by the thinning algorithm of Saha and Chaudhuri [20]; (c) segmented parts.

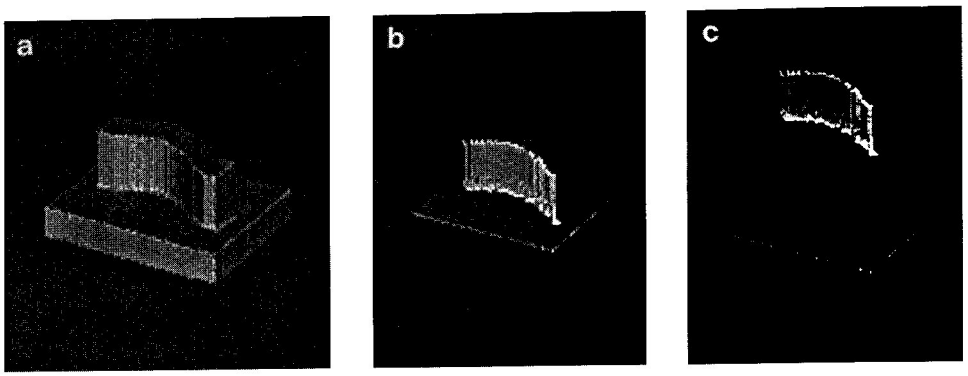

FIG. 9. Results of segmentation. (a) Original object; (b) surface skeleton representation by the thinning algorithm of Saha and Chaudhuri [20]; (c) segmented parts.

The minimum sizes of the rectangular parallelepiped to enclose these images are $71 \times 114 \times 71$, $79 \times 40 \times 60$, and $77 \times 77 \times 29$, respectively. Here the background is made black to produce a better visual effect. Figures 8b–10b are three surface skeleton representations obtained by applying the thinning algorithm of Saha and Chaudhuri [20] on the images of Figs. 8a–10a, respectively. In Fig. 10 the original image and the skeleton are displayed from different angles of view. Figures 8c–10c demonstrate the segmented parts obtained from Figs. 8b–10b, respectively, using the method described in Sections IV.A and IV.B.

The segmentation process is based on observing the topological junction points. Nontopological segmentation is not possible by this method. Figure 11 is an example where the two surfaces can not be separated. In this case the segmentation could be done by noting the abrupt change in surface normal direction.

The proposed approach can segment a surface skeleton representation into surfaces and arcs containing no junction. More specifically, a segmented part is one of the following: a simple surface patch (topologically equivalent to a rectangular sheet), a simple closed surface (topologically equivalent to a hollow sphere), a simple curve (topologically equivalent to a straight line segment), a simple closed curve (topologically equivalent to a circle). The segmented parts (along with the depth information derived from a thinning algorithm) can be used to represent an object by a set of simple geometric features (restricted by a predefined feature set).

Malandain *et al.* [13] also considered segmentation of a 3D surface using topological features. They have made some classification of points in $\mathcal{A}^3$ where every point type classification is unique and applied the same classification in digital domain. The unique classification table creates some undesired situations, some of which are described by them. Also, Fig. 12 of [13] has a junction of surfaces which is not a curve but a surface of three-point width which will create further problems in segmenting the surfaces. Such a situation does not arise by our approach. Also in [13] no mention was made of how to compute efficiently the numbers of adjacent object components (called C^* in [13]) and adjacent background components (called $\overline{C}$ in [13]) of a point in its $3 \times 3 \times 3$ neighborhood. On the otherhand, our algorithm *topo_para* is an efficient approach to compute the parameters related to topological segmentation that is characterized by Table 1. The concept

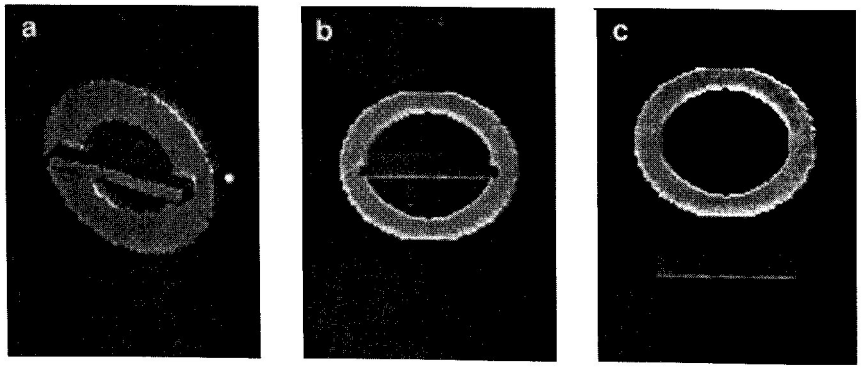

FIG. 10. Results of segmentation. (a) Original object; (b) surface skeleton representation by the thinning algorithm of Saha and Chaudhuri [20]; (c) segmented parts.

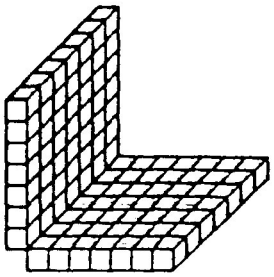

FIG. 11. An example where two surfaces can not be segmented by the proposed method.

of using 18-neighborhood in computing the number of 6-adjacent background components of a point in its $3 \times 3 \times 3$ neighborhood was first proposed by Saha *et al.* [18].

The change in the numbers of black components, tunnels, and cavities within $\mathcal{N}(p)$ when p is deleted is useful in many image processing applications. The most naive and trivial approach to detecting the change is to prepare a look_up_table for all possible black/white combinations in its $3 \times 3 \times 3$ neighborhood which needs 2^{26} bytes, i.e., 64 Mbytes of memory to store the look_up_table as compared to the 8 Kbytes needed by our approach. Only for Class 9 does our algorithm *topo_para* need the configuration of all the 26 points in the $3 \times 3 \times 3$ neighborhood. For Classes 0–8 it needs the configuration of fewer points.

ACKNOWLEDGMENTS

The authors thank Professor D. Dutta Majumder, Dr. S. K. Parui, and Ms. Mahua Dutta for their interest and valuable suggestions about this work. The authors also express their thanks to the unknown referees who read the manuscript so carefully and suggested several improvements.

REFERENCES

1. H. Bieri and W. Nef, Algorithms for the Euler characteristic and related additive functionals of digital objects, *Comput. Vision Graphic Image Proces.* **28,** 1984, 166–175.
2. C. R. Dyer, Computing the Euler number of an image from its quadtree, *Comput. Graphics Image Process.* **13,** 1980, 270–276.
3. M. J. E. Golay, Hexagonal parallel pattern transformations, *IEEE Trans. Comput.* **C-18,** 1969, 733–740.
4. G. T. Herman and D. Webster, A topological proof of a surface tracking algorithm, *Comput. Vision Graphics Image Process.* **23,** 1983, 162–177.
5. T. Y. Kong, Digital topology with applications to image processing, *Doctoral dissertation, University of Oxford*, 1986.
6. T. Y. Kong and A. Rosenfeld, Digital topology: Introduction and survey, *Comput. Vision Graphics Image Process.* **48,** 1989, 357–393.
7. T. Y. Kong and A. W. Roscoe, Continuous analogs of axiomatized digital surfaces, *Comput. Vision Graphics Image Process.* **29,** 1985, 60–86.
8. T. Y. Kong, A. W. Roscoe, and A. Rosenfeld, Concepts of digital topology, *Topology Appl.* **46,** 1992, 219–262.
9. T. Y. Kong and A. W. Roscoe, Characterizations of simply-connected finite polyhedra in 3-space, *Bull. London Math. Soc.* **17,** 1985, 575–578.
10. S. Lobregt, P. W. Verbeek, and F. C. A. Groen, Three-dimensional skeletonization: Principle and algorithm, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-2,** 1980, 75–77.
11. D. G. Morgenthaler, Three-dimensional simple points: Serial erosion, parallel thinning and skeletonization, Tech. Rep. TR-1005, Computer Vision Laboratory, University of Maryland, 1981.
12. D. G. Morgenthaler and A. Rosenfeld, Surfaces in three-dimensional digital images, *Inform. Control* **51,** 1981. 227–247.
13. G. Malandain, G. Bertrand, and N. Ayache, Topological segmentation of discrete surfaces, *Int. J. Comp. Vision* **10,** no. 2, pp. 183–197, 1993.
14. G. M. Reed and A. Rosenfeld, Recognition of surfaces in three-dimensional digital images, *Inform. Control* **53,** 1982, 108–120.
15. A. Rosenfeld, Connectivity in digital pictures, *J. Assoc. Comput. Mach.* **17,** 1970, 146–160.
16. A. Rosenfeld, Three-dimensional digital topology, *Inform. Control* **50,** 1981, 119–127.
17. A. Rosenfeld, A characterization of parallel thinning algorithms, *Inform. Control.* **29,** 1975, 286–291.
18. P. K. Saha, B. Chanda, and D. D. Majumder, Principles and algorithms for 2D and 3D shrinking, Tech. Rep. TR/KBCS/2/91, N.C.K.B.C.S. Library, Indian Statistical Institute, Calcutta, India, 1991.
19. P. K. Saha, B. B. Chaudhuri, B. Chanda, and D. D. Majumder, Topology preservation in 3D digital space, *Pattern Recognit.* **27,**(2), 1994 295–300.
20. P. K. Saha and B. B. Chaudhuri, Detection of 3D simple points for topology preserving transformations with application to thinning, *IEEE Trans. Pattern Anal. Mach. Intell.* **16,** 1994, 1028–1032.
21. P. K. Saha and B. B. Chaudhuri, Simple point computation and 3D thinning with parallel implementation, Tech. Rep. TR/KBCS/1/93, N.C.K.B.C.S. Library, Indian Statistical Institute, Calcutta, India, 1993.
22. J. I. Toriwaki, S. Yokoi, T. Yonekura, and T. Fukumura, Topological properties and topology-preserving transformation of a three-dimensional binary picture, in *Proceedings, 6th International Conference on Pattern Recognition, 1982*, pp. 414–419.
23. Y. F. Tsao and K. S. Fu, A Parallel Thinning Algorithm for 3D Pictures, *Comput. Graphics Image Process.* **17,** 1981, 315–331.
24. Y. F. Tsao and K. S. Fu, A 3D parallel skeletonwise thinning algorithm, in *Proceedings, IEEE PRIP Conference, 1982*, 678–683.
25. K. Voss, Images, objects, and surfaces in Z^n, *Int. J. Pattern Recognit. Artif. Intell.* **5,** 1991, 797–808.