CVGIP: GRAPHICAL MODELS AND IMAGE PROCESSING
Vol. 56, No. 6, November, pp. 462–478, 1994

Building Skeleton Models via 3-D Medial Surface/Axis Thinning Algorithms[1]

TA-CHIH LEE AND RANGASAMI L. KASHYAP

Knowledge-Based Systems Laboratory, Department of Electrical Engineering, Purdue University, West Lafayette, Indiana 47907

AND

CHONG-NAM CHU

Department of Mechanical Design & Product Engineering, Seoul National University, Seoul, Korea

Received July 16, 1993; revised July 14, 1994; accepted August 10, 1994

In this paper, we present an efficient three-dimensional (3-D) parallel thinning algorithm for extracting both the medial surfaces and the medial axes of a 3-D object (given as a 3-D binary image). A new Euler table is derived to ensure the invariance of the Euler characteristic of the object, during thinning. An octree data structure of 3 × 3 × 3 lattice points is built to examine the local connectivity. The sets of "simple" points found by different researchers are compared with the constructed set. Different definitions of "surface" points including ours are given. By preserving the topological and the geometrical conditions, our algorithm produces desirable skeletons and performs better than others in terms of noise sensitivity and speed. Pre- and postprocessors can be used to remove additional noise spurs. Its use in defect analysis of objects produced by casting and forging is discussed. © 1994 Academic Press, Inc.

1. INTRODUCTION

In the past, a number of thinning or skeletonization algorithms for 2-D digital images have been developed to extract shape features for image processing purposes. Those thinned figures have applications ranging from biological cell studies to character recognition (see[1] for a recent survey of 2-D skeletonization algorithms). Thinning schemes are useful because they reduce large amounts of image data to thin-line patterns (skeletons) so that shape analysis can be implemented. Although there exist many 2-D thinning algorithms, there are only a few studies done for the 3-D case. There are three primary reasons. First 3-D digital data was not readily available in the past. Second, 3-D topological properties were harder to address due to the higher dimensionality. Third, useful applications of 3-D skeletons were not apparent. However, these factors are no longer true. 3-D digital images (or objects) can now be generated through computed tomography [2] or computer graphics [3]. Concepts such as holes, Euler characteristic, connectivity, and simple points have been studied for 3-D domain by various researchers [4–6]. In this paper, we present an efficient parallel thinning algorithm for extracting both the medial surfaces and the medial axes (we refer to both of these as the "skeleton") of a 3-D digital object. The skeletons can be used for casting and forging defect analysis.

In 3-D Euclidean space, the skeleton (or the medial surface) of a geometry is the locus of the centers of all inscribed maximal spheres of the object where these spheres touch the boundary at more than one point [1, 6, 10, 11]. It can be considered as the simplified characteristic of a geometry and is used to reduce the feature search space of a geometric model. In the discrete space, there exist two basic methods to extract the skeleton from an object. One is to use the distance transformation. The other method is to use the thinning procedure that repetitively deletes the border points of an object satisfying topological and geometrical constraints until a smaller set of connected points is acquired. This set of connected points is only an approximation to the "true" skeleton in the Euclidean space. We will only consider the thinning operation in this paper. The topological requirement is to preserve the number of connected objects, cavities, and holes in the original shape [4–6]. In our algorithm, Euler characteristic and connectivity are preserved to guarantee the invariance of these topological properties. The geometrical condition, on the other hand, is used to ensure the desired width and location of the skeleton. Surface and arc end point conditions are given, and depending on these end point condition, either the medial surface or the medial axis can be extracted by the thinning operation. In addition to the topological and the geometrical con-

[1] This work was supported by the National Science Foundation under Grant CDR 8803017 to the Engineering Research Center for Intelligent Manufacturing Systems. The authors also acknowledge the help of Sanjeev Trika in preparing the manuscript.

1049-9652/94 $6.00
Copyright © 1994 by Academic Press, Inc.
All rights of reproduction in any form reserved.

straints, a good thinning operation should produce a skeleton that represents significant features of a geometry and that is insensitive to boundary noise. It is possible that few spurs still remain in the skeleton because of the small variations on the surfaces of the object. In this case, we can either use a pre-processor such as a 3-D digital filter to smooth out noises in 3-D digital images before applying the thinning process or use a post-processor to remove spurs in the skeleton afterwards by thresholding on the length of each skeleton arc.

This paper is organized as follows. In Section 2, properties of 3-D digital topology pertaining to thinning operations are discussed. An Euler table is derived to determine the invariance of the Euler characteristic, and an octree data structure of $3 \times 3 \times 3$ lattice points is built to examine the local connectivity. Various definitions of a "simple" point are also examined. Section 3 describes our thinning algorithm which can be used to extract either the medial surface or the medial axis. A new definition of a "surface" point is given. Comparisons are made among various thinning algorithms. In Section 4, skeleton models are constructed by using pre- and postprocessors. Applications of skeleton models for casting and forging defect analysis are presented in Section 5.

2. DIGITAL TOPOLOGY

2.1. *Some Basic Definitions*

The study of the topological properties such as adjacency, connectivity, and Euler characteristic is called digital topology. In this section, some basic definitions of digital topology are given. Further details can be found in [4, 5]. The object space used to represent a geometry is stored in a 3-D binary array of size $k_{\max} \times j_{\max} \times i_{\max}$: $Z^3 = \{v = (k, j, i) | (k, j, i) < (k_{\max}, j_{\max}, i_{\max})\}$, in which every volume element (or voxel), v, is represented by its centroid point (k, j, i), and the point has the value of either 0 or 1 (we will use "voxel" and "point" interchangeably in this paper). Objects of interest in the domain Z^3 are represented by the nonempty subset S which consists of all the points with the value 1. The complement of S, $\bar{S}$, consists of all the points with the value of 0. A point $v = (k, j, i)$ in S is called a "border" point if at least one of its 6-neighbors is in $\bar{S}$ [10, 11]. For convenience, it is assumed without loss of generality that all boundary points of Z^3 (i.e., $\{(k, j, i) | k = 0$ or $k_{\max}, j = 0$ or $j_{\max}, i = 0$ or $i_{\max}\}$) belong to $\bar{S}$. Each point v has two most common types of neighbors:

(1) 6-connected neighbors, $N_6(v = (k, j, i)) = \{(t, s, r) | |t - k| + |s - j| + |r - i| = 1\}$; and

(2) 26-connected neighbors, $N_{26}(v = (k, j, i)) = \{(w, v, u) | \max(|w - k|, |v - j|, |u - i|) = 1\}$.

In other words, $N_6(v)$ are the direct adjacent neighbors of point v, and $N_{26}(v)$ are those 6-neighbors plus all the diagonal and the corner neighbors within a $3 \times 3 \times 3$ neighborhood, as illustrated in Fig. 1a. Two other ways used in this paper to represent this neighborhood, $N_{26}(v)$, are shown in Figs. 1b and 1c. For simplicity, $N(v)$ is used to denote $N_{26}(v)$ if not specified otherwise, and $N(v)$ together with v is denoted by $\{N(v) \cup v\}$.

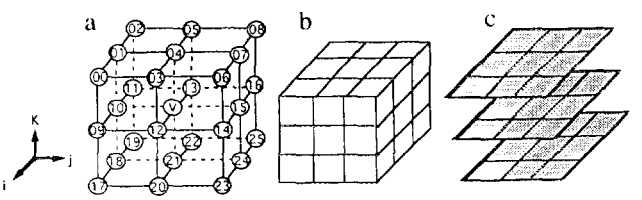

FIG. 1. (a) Indices of the 26-neighborhood of a point v, $N(v)$; (b) and (c) are two other ways of representing $N(v)$.

A 6-(26-) path of length n in S is a sequence of points, $P_{6\text{-}(26\text{-})} = \{v_i \in S \mid 0 \le i \le n\}$, such that v_i is 6-(26-) adjacent to its preceding point, v_{i-1}, for all i. Two points v_i and v_j are 6-(26-) connected in S if there exists a 6-(26-) path, $v_i, \ldots, v_j$, such that all of the points are in S. The set of points which are connected under 6-(26-) connectivity in S are called 6-(26-) connected components (or objects) of S. Similar definitions can be stated for $\bar{S}$. Opposite types of connectivity are usually used for S and $\bar{S}$ to avoid ambiguous situations (see [5] for examples). In this paper, 26- and 6-connectivity are used for S and $\bar{S}$, respectively. An arc (or a simple arc) is a 6-(26-) path such that each point on the 6-(26-) path is 6-(26-) adjacent to just two other points on the path, with the exception that the two end points of the arc have only one neighbor. On the other hand, a curve (or sometimes is called a closed curve) is a 6-(26-) path that has exactly two neighbors 6-(26-) adjacent to every points on the path. There exists exactly one component of $\bar{S}$ which is called the background that contains the boundary points of Z^3. The other components of $\bar{S}$ are completely surrounded by S, and are called cavities of S. Definition of a hole in a 3-D digital image is different from a cavity in that it is not completely surrounded by S. A 3-D hole can be thought of as the tunnel in a torus [4]. An object with n holes has a surface topologically equivalent to an n-hole torus. In other words, if an object has a hole, then the object cannot be continuously deformed to a point without splitting the surface. However, counting the number of 3-D holes is not an easy task because it is quite hard to distinguish a hole from the background. Fortunately, we can describe a 3-D hole differently by using the well-known Euler formula that relates the number of objects, cavities, and holes in a simple and consistent manner.

2.2. *Euler Characteristic*

The 3-D Euler characteristic $\chi(S)$ is defined by the global formula

$$\chi(S) = O(S) - H(S) + C(S), \tag{1}$$

where $O(S)$, $H(S)$, and $C(S)$ are the numbers of connected objects, holes, and cavities of S, respectively [4–6]. Similar to the 2-D case, a local formula $G(S)$ can be used to reduce the complexity of calculating $\chi(S)$ by considering predefined neighborhoods, $N(v)$, for each v in S. Since 3-D objects can be represented either as surface patches (Fig. 1b) or lattice of points (Fig. 1a), the Euler characteristic can be computed locally based on either surface characteristic or the lattice structure of S.

Using the surface patches of S, ∂S, Lobregt *et al.* [7] devised a table look-up method to compute the Euler characteristic within $N(v)$ for 3-D objects. The authors divide $N(v)$ into eight overlapping $2 \times 2 \times 2$ cubes (or octants), labelled as $N^2(v)$. The Euler characteristic is computed by summing the contribution of each octant (i.e., $G = \sum G_6(\partial S \cap N^2(v))$). The advantage of using this approach is that there are only 2^8 (=256) possible configurations for each octant; hence, $G_6(\partial S \cap N^2(v))$ can be precalculated and stored in a table resulting in an efficient computation. For 26-connected images, $G_{26}(\partial S \cap N^2(v))$ was found to be equivalent to $G_6(\partial S \cap N^2(v))$. The Euler characteristic of ∂S for the 22 basic configurations of an octant is given in column 4 of Table 1. The remaining 256 configurations can be generated through symmetry operations on the 22 octants.

When considering a 3-D object as a finite collection of points, a similar definition of the Euler characteristic of S can be derived from algebraic topology (see [4–8] for details) as

$$G_6(S) = v - e + f - \text{oct}, \tag{2}$$

where v, e, f, and oct are the total number of vertices, edges, faces, and octants in S, respectively. It was shown in [4] that the global and the local computation of Euler characteristic are actually equivalent, i.e., $\chi(S) = G(S)$. By the Poincaré duality, it is well known that $\chi(S) = \frac{1}{2}\chi(\partial S)$ in algebraic topology [8] where $\chi(\partial S)$ is the Euler characteristic of the surface ∂S. We will show that this relation also holds in digital topology. Let us first consider an octant shown in Fig 2a, and consider only 6-connected objects.

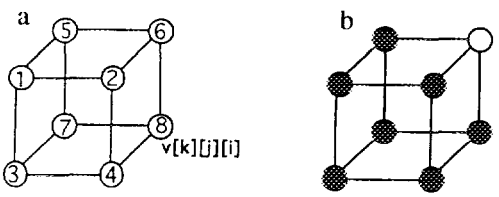

FIG. 2. (a) A $2 \times 2 \times 2$ cube or octant, $N^2(v)$; (b) octant with binary configuration 1111 1011 (the solid spheres represent points with value 1 and the empty spheres represent points with value 0).

As we come across each octant of the lattice structure of S, each vertex, edge, and face of the octant is encountered eight times, four times, and twice, respectively, but only once for each octant. Therefore, from Eq. (2), the Euler formula is an octant $N^2(v)$ becomes

$$G_6(S \cap N^2(v)) = \frac{v_i}{8} - \frac{e_i}{4} + \frac{f_i}{2} - \text{oct}_i. \tag{3}$$

Column 5 of Table 1 is the value of the Euler characteristic for the lattice representation of S using Eq. (3). The example in Fig. 2b is for the binary configuration 1111 1011 (decimal 251). $G_6(S \cap N^2(v))$ for this example is $\frac{7}{8} - \frac{9}{4} + \frac{3}{2} - 0 = \frac{1}{8}$. Note that $G_6(\partial S \cap N^2(v))$, the Euler characteristic for the surface representation of S, is $\frac{1}{4}$ (from Table 1).

TABLE 1
Euler Table for Surface and Lattice Representation

i	Binary 1234 5678	n	$G_6(\partial S\rfloor = G_{26}(\partial S)$	$G_6(\partial S\rfloor = G_{26}(\partial S)$
1	0000 0000	0	0	0
2	0000 0001	1	1/4	1/8
3	0000 0011	3	0	0
4	0000 1001	9	1/2	1/4
5	1000 0001	129	1/2	1/4
6	0000 0111	7	−1/4	−1/8
7	1000 0011	131	1/4	1/8
8	0010 1001	41	3/4	3/8
9	0000 1111	15	0	0
10	0010 1011	43	−1/2	−1/4
11	1000 1011	139	−1/2	−1/4
12	0100 1011	75	0	0
13	1100 0011	195	0	0
14	0110 1001	105	1	1/2
15	1110 1001	233	−1/4	−1/8
16	1101 0011	211	−3/4	−3/8
17	1111 0001	241	−1/4	−1/8
18	1101 1011	219	−3/2	−3/4
19	1111 1001	249	−1/2	−1/4
20	1111 0011	243	0	0
21	1111 1011	251	1/4	1/8
22	1111 1111	255	0	0

PROPOSITION 1. *The formula* $\chi(S) = \frac{1}{2}\chi(\partial S)$ *holds in digital topology.*

Proof. Column 4 of Table 1 lists the $G_{26(6)}(\partial S \cap N^2(v))$ values derived based on the results in [7]. We have calculated the $G_{26(6)}(S \cap N^2(v))$ values in column 5 of the table. It is apparent that $G_{26(6)}(S \cap N^2(v)) = \frac{1}{2}G_{26(6)}(\partial S \cap N^2(v))$. In [9], the authors have shown that the 22 configurations of an octant in Table 1 is complete. Hence, since the Euler characteristic is an additive property and $G(S) = \chi(S)$ [4, 5],

$$\chi(S) = \sum_i G_{26(6)}(S \cap N_i^2(v))$$

$$= \tfrac{1}{2}\sum_i G_{26(6)}(\partial S \cap N_i^2(v)) = \tfrac{1}{2}\chi(\partial S). \quad \text{Q.E.D.}$$

In order to preserve topological properties for thinning operations, we are required to show that a point v within $N(v)$ is invariant in the sense of the Euler characteristic, i.e., $\partial G_6(S \cap N(v)) = G_6(S \cap N(v)) - G_2(S \cap \{N(v) \cup v\}) = 0$, where δ denotes the change of the Euler characteristic. Consequently, from Eq. (3) and the fact that $\delta v_i = -1$, we get

$$\delta G_6(S \cap N^2(v)) = \delta \text{oct}_i - \frac{\delta f_i}{2} + \frac{\delta e_i}{4} - \frac{1}{8}, \tag{4}$$

where δoct_i, δf_i, and δe_i are the changes in the number of octants, faces, and edges that contain the point v, respectively (for example, δf_i = number of faces if v is not included, - number of faces if v is included). Using an idea similar to that in [7], an Euler table for 6-connected objects can be derived from Eq. (4). It is listed in column 2 of Table 2. Using the same example in Fig. 2b for an illustration, $\delta G_6(S \cap N^2(v))$ is $0 - \frac{1}{2} + \frac{2}{4} - \frac{1}{8} = -\frac{1}{8}$. For simplicity, the entries in the table have been multiplied by 8. Note that Table 2 cannot be reduced to the 22 basic configurations as in Table 1 because point v is fixed at the binary position 8 (see Fig. 2a), and none of Table 2's even entries are available. The apparent advantage of using Table 2 over Table 1 is that the effort in computation of Euler invariance is reduced by one-half. For 26-connected objects we can complement the 6-connected and use the formula $\delta G_{26}(S) = \delta G_6(\bar{S})$ to determine the invariance of the Euler characteristic. The results are listed in column 3 of Table 2.

2.3. *Simple Point and Connectivity*

In the past, simple points have been used for thinning operations to preserve topological properties such as the number of connected objects, cavities, and holes [6, 7, 10–14]. A discussion of various definitions of "simple" points follows. To determine a simple point, Shrihari, Udupa, and Yau partition $\{N(v) \cup v\}$ (i.e., $N(v)$ along with v) and $N(v)$ such that all the points within each partition are connected [12]. If the number of the two partitionings are the same, then v is simple. In other words, path connectivity should be preserved. However, as illustrated in Fig. 3, this condition is necessary but not sufficient. After removing v, the path connectivity is preserved in $N(v)$ but a hole (shown as uvw) is created during the process. Therefore, their algorithm does not preserve digital topology. Lobregt, Verbeek, and Groen define simple points as border points which are invariant in the sense that removing them preserves Euler characteristic [7]. However, using this characterization alone is also insufficient because $\delta G(S \cap N(v)) = \partial \chi(S \cap N(v)) = 0$ and $\delta C(S \cap N(v)) = 0$ imply $\delta O(S \cap N(v)) =$

TABLE 2
Euler Table for Preserving Euler Characteristic

n	$8\,\delta G_6$	$8\,\delta G_{26}$	n	$8\,\delta G_6$	$8\,\delta G_{26}$	n	$8\,\delta G_6$	$8\,\delta G_{26}$	n	$8\,\delta G_6$	$8\,\delta G_{26}$	n	$8\,\delta G_6$	$8\,\delta G_{26}$
1	-1	1	53	3	1	105	-1	5	157	3	1	209	1	-1
3	1	-1	55	1	-1	107	1	3	159	1	-1	211	3	1
5	1	-1	57	1	3	109	1	3	161	-1	-3	213	-1	1
7	3	1	59	-1	1	111	-1	1	163	1	-1	215	1	-1
9	-1	-3	61	3	1	113	1	-1	165	1	3	217	1	3
11	1	-1	63	-3	-1	115	-1	1	167	3	1	219	3	1
13	1	-1	65	-1	-3	117	-1	1	169	-1	1	221	-1	1
15	-1	1	67	1	3	119	-3	-1	171	1	-1	223	-3	-1
17	1	-1	69	1	-1	121	1	3	173	1	3	225	-1	1
19	3	1	71	3	1	123	-1	1	175	-1	1	227	1	3
21	3	1	73	-1	1	125	-1	1	177	1	-1	229	1	3
23	5	-1	75	1	3	127	-7	-1	179	-1	1	231	3	1
25	1	3	77	1	-1	129	-1	-7	181	3	1	233	-1	5
27	3	1	79	-1	1	131	1	-1	183	1	-1	235	1	3
29	3	1	81	1	-1	133	1	-1	185	1	3	237	1	3
31	1	-1	83	3	1	135	3	1	187	-1	1	239	-1	1
33	-1	-3	85	-1	1	137	-1	-3	189	3	1	241	1	-1
35	1	-1	87	1	-1	139	1	-1	191	-3	-1	243	-1	1
37	1	3	89	1	3	141	1	-1	193	-1	-3	245	-1	1
39	3	1	91	3	1	143	-1	1	195	1	3	247	-3	-1
41	-1	1	93	-1	1	145	1	-1	197	1	-1	249	1	3
43	1	-1	95	-3	-1	147	3	1	199	3	1	251	-1	1
45	1	3	97	-1	1	149	3	1	201	-1	1	253	-1	1
47	-1	1	99	1	3	151	5	-1	203	1	3	255	1	-1
49	1	-1	101	1	3	153	1	3	205	1	-1			
51	-1	1	103	3	1	155	3	1	207	-1	1			

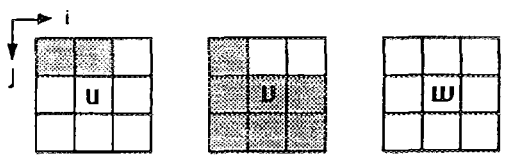

FIG. 3. A hole is created after the removal of v, even though path connectivity is preserved.

$\delta H(S \cap N(v))$. Hence, it is not possible to determine the number of objects and holes remain the same. Figure 4 illustrates an example where the removal of v not only creates a hole but also an additional object which results in no change of the Euler characteristic. However, this is certainly not desirable because it changes the topology of the original object. In addition, if the deletion of v removes a hole, then v is a nonsimple point [5]. For that reason, a more stringent characterization of a simple point is needed.

Kong and Rosenfeld in their survey paper on digital topology [5] discuss five different definitions of "topology preservation" of a 3-D thinning algorithm. One of which is a characterization of a simple point given by Morgenthaler [6] that has been known for being a complete set. A border point v is classified as "simple" if and only if its removal does not change the number of connected objects and holes for both S and $\bar{S}$. Simple points, as defined in [6, 13] have the following property:

PROPOSITION 2. *A border point v is a "simple" point if and only if*

$$\text{(a) } \delta G(S \cap N(v)) = 0; \text{ and} \tag{5}$$

$$\text{(b) } \delta H(S \cap N(v)) = 0; \text{ or} \tag{6}$$

$$\text{(c) } \delta O(S \cap N(v)) = 0 \tag{7}$$

Proof. Since Morgenthaler's definition of a simple point has been proven to be complete, we will show that the two definitions are actually equivalent. First, we will show the "if" part. In [13], the authors show that if $\delta G(S \cap N(v)) = 0$ where $v \in S$ is a border point and $\delta O(S \cap N(v)) = 0$ then $\delta O(\bar{S} \cap N(v)) = 0$. A similar derivation shows that $\delta H(S \cap N(v)) = 0$ implies $\delta H(\bar{S} \cap N(v)) = 0$ given $\delta G(S \cap N(v)) = 0$. We also know that $\delta \chi(S \cap N(v)) = \delta O(S \cap N(v)) - \partial H(S \cap N(v)) + \partial C(S \cap N(v))$ from Eq. (1) where "δ" denotes the "change" operator defined earlier. If $\delta G(S \cap N(v)) = 0$, then $\delta O(S \cap N(v)) = \delta H(S \cap N(v))$ because $\delta C(S \cap N(v)) = 0$ for a border point v and $\delta G(S \cap N(v)) = \delta \chi(S \cap N(v))$. Therefore, if Eqs. (5) and (6) or Eqs. (5) and (7) are satisfied, then $\delta O(S \cap N(v)) = 0$, $\delta O(\bar{S} \cap N(v)) = 0$, $\delta H(S \cap N(v)) = 0$, and $\delta H(\bar{S} \cap N(v)) = 0$. In other words, $O(S \cap \{N(v) \cup v\}) = O(S \cap N(v))$, $O(\bar{S} \cap \{N(v) \cup v\}) = O(\bar{S} \cap N(v))$, $H(S \cap \{N(v) \cup v\}) = H(S \cap N(v))$, and $H(\bar{S} \cap \{N(v) \cup v\}) = H(\bar{S} \cap N(v))$.

The "only if" part is straightforward. Since $O(S \cap \{N(v) \cup v\}) = O(S \cap N(v))$ and $H(S \cap \{N(v) \cup v\}) = H(S \cap N(v))$, then $\delta O(S \cap N(v)) + \delta H(S \cap N(v)) = \delta G(S \cap N(v)) = 0$ provided v is a border point. It follows since $\delta H(S \cap N(v)) = 0$ or $\delta O(S \cap N(v)) = 0$ are also true. Q.E.D.

Tsao and Fu use two checking windows in $N(v)$ to determine Eq. (6) [10, 13]. Checking windows WI, WJ, and WK are defined for each principle planes, $x = i$, $y = j$, and $z = k$ within $N(v)$, respectively. Each border point has two windows associated with it. However, as illustrated in Fig. 5, this approach does not provide a sufficient condition. That is, not all of the simple points can be detected using two checking windows. It can be seen that v is characterized as a nonsimple point due to two non-simple windows WJ and WK (i.e., points in both windows are disconnected after delecting v), but in fact v should be a simple point.

In [11], Gong and Bertrand use some predicates based on Morgenthaler's definition to determine simple points. However, these new predicates have the same drawback of insufficiency. The same configuration in Fig. 5 illustrates an example where points P_l and P_m violate condition $2c$ and $2d$ in their definition of a simple point (points P_d, $P_{\bar{d}}$, P_i, P_l, P_j, P_k, P_m are defined in [11]). Malandain and Bertrand in [14] give another simpler definition of a simple point based on Morgenthaler's characterization. The definition requires that condition in Proposition 3 to be satisfied. In addition, the number of 18-connected $\bar{S}$ components needs to be 1 after removing v. Even though the algorithm for computing connected components in $N(v)$ is straightforward, their set of simple points is still insuffi-

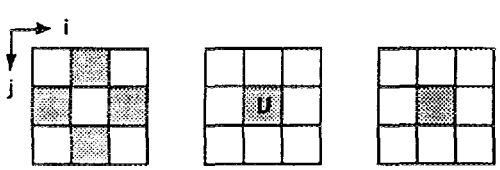

FIG. 4. An example illustrating that removing a point v does not affect the Euler characteristic, but does modify the digital topology.

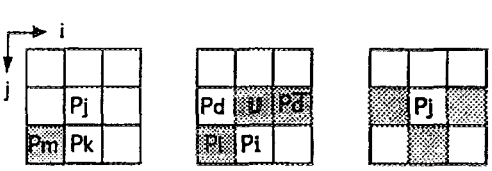

FIG. 5. v is a simple point, but fails to be characterized in [10, 11, 13].

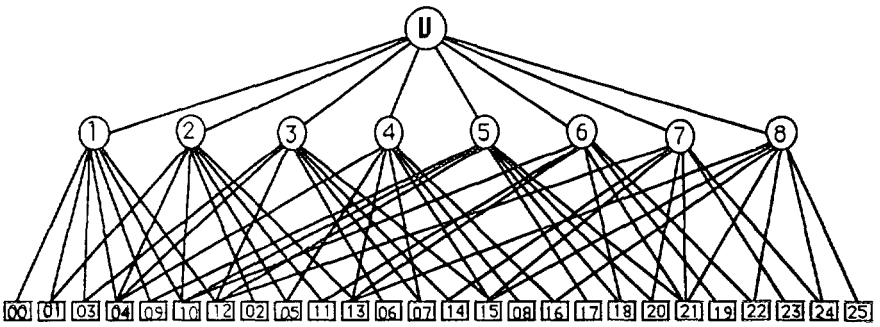

FIG. 6. Adjacency tree of $N(v)$.

cient because this definition can not detect the creation of holes (see Fig. 3 for an example). Hence, removing v will change the Euler characteristic.

Instead of trying to determine 3-D holes by using checking windows as in [10, 13], we can alternatively use Eq. (7) to test "simplicity" of a point by determining the number of changes of connected objects in $N(v)$. Since the center point $v \in S$ and all of its 26-neighbors are connected to it, $O(S \cap N(v))$ is always 1 before the deletion of v. Therefore, if $O(S \cap N(v))$ is 1 after the removal of v, then Eq. (7) is met. Hence, we have

PROPOSITION 3. *$\delta O(S \cap N(v)) = 0$ if and only if $O(S \cap N(v)) = 1$.*

Proof. See the proof for Proposition 1 in [6].

To determine connectivity in a more efficient manner, we first propose an octree-type data structure for $N(v)$. The underlying idea of an octree representation is to recursively partition a cubic volume into eight sub-cubes called octants until homogeneity exists in each of those octants (i.e., all voxels in a given octant either belong to S or $\bar{S}$) [15]. The octree data structure can be represented as a tree in which each non-terminal node of the three has eight children, and the leave nodes of the tree denote data points. Exploiting the recursive nature of the octree structure, we develop a labeling algorithm to determine the number of 26-connected objects in $N(v)$. In our data structure (called the adjacency tree of $N(v)$) the topmost node of the tree is v, the center point of $N(v)$. By subdividing $N(v)$ in Fig. 1a into eight overlapping octants as shown in the second level of the tree, adjacency relationships among those octants are manifested by connecting overlapping points in them (see Fig. 6). The following algorithm employs the adjacency tree data structure to calculate the number of connected objects in $N(v)$.

The labeling algorithm consists of two procedures, *$N(v)$_Labeling* and *Octree_Labeling*. The function of *$N(v)$_Labeling* is to label $N(v)$ so that the number of connected objects is determined. First, a local copy of $N(v)$ is made for the purpose of labeling. An initial label other than the value of 1 is assigned. We use 2 for such label. Indexing through the 26 neighbors of v, a call to the second procedure *Octree_Labeling* is made for each point that has the value 1. The second procedure *Octree_Labeling* is used to label corresponding octants of the point. The main structure of this function is based on the octree scheme where there are eight octants. For each octant, there are seven points which are under investigation excluding v itself. The order of the examination is dependent on the number of adjacent octants of the particular point. If the point has a 1 value, it is set to the current label and the corresponding octants are labeled by recursively invoking procedure *Octree_Labeling* with their octant indices and with the same label. For instance, in the case of octant 1 in Fig. 6, if cube [4] is 1 (i.e., cube with respect to index 4 in the bottom line of Fig. 6), it is set to the current label and procedure calls *Octree_Labeling*(2, label), *Octree_Labeling*(3, label), and *Octree_Labeling*(4, label) are made. It turns out that for each octant, there exists one vertex point that requires no procedure call, three edge points that need one call, and three face points that need three calls to their adjacent octants. We illustrate an example in Fig. 7 with a configuration of 10100000010100000100011000 0 for $N(v)$. The labeling order by applying the algorithm is the cubic index 0, 9, 20, 21, 11, 2, and 16. Compared with serial scheme in [16], the advantage of using the proposed algorithm is

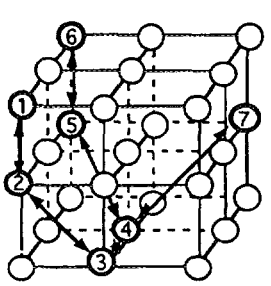

FIG. 7. An example illustrating the order of points being labeled by the proposed algorithm.

that there is no need for an equivalence table and a second-pass of updating.

ALGORITHM: *N(v)_Labeling*
INPUT: The coordinate of v of a 3-D binary image.
OUTPUT: Number of connected objects in $N(v)$.
- Store a local copy of $N(v)$ as cube[26] for labeling;
- Set label = INIT_LABEL;/*INIT_LABEL $\neq$ 1 */

FOR each point in cube with value = 1 DO
- Determine the octant index of the point;/* second level of the adjacency tree */
- Call *Octree_Labeling* with the current octant index and label;
- label = label + 1;

end FOR;
Return (label − 2); /* subtract one for INIT_LABEL = 2 and subtract one for overcounting */
end ALGORITHM.

ALGORITHM: *Octree_Labeling*
INPUT: Octant index, label.
OUTPUT: labeled cube
FOR the particular octant DO
- IF the value of points in the octant = 1 THEN
  - Set label to those points;
  - Call *Octree_Labeling* with adjacent octant indices and the current label;

end FOR;
end ALGORITHM.

We conclude this section with a depiction of the different characterizations of a simple point that have been discussed so far in Table 3. Note that $\delta G = \delta G(S \cap N(v))$, $\delta O = \delta O(S \cap N(v))$, $\delta \overline{O} = \delta O(\overline{S} \cap N(v))$, $\delta H = \delta H(S \cap N(v))$, and $\delta \overline{H} = \delta H(\overline{S} \cap N(v))$. There are 2^{26} (=67,108,864) different possible configurations of voxels around a voxel v in $N(v)$. It was asserted in Proposition 2 that our characterization of a simple point is equivalent to the one given by Morgenthaler [6]. Both Morgenthaler and our methods classify v as simple in 38.72% (=25,984,552) of these configurations. Definitions suggested in [7, 12] are overcharacterization of a simple point. Even though the predicates given by Gong and Bertrand [11] are simple, their definition only captures 0.10% (=65,536) of the total 2^{26} configurations. Tsao and Fu's hole detection approach using two checking windows [10, 13] has 19.70% (=13,220,446), but still falls short of the true simple point set. Therefore, their thinning algorithms based on such characterization will tend to require more thinning iterations and longer running time.

TABLE 3
Different Methods of Characterizing a "Simple" Point

Authors	Methods	% of 2^{26} (=67,108,864)	Set	Comments
Morgenthaler	$\delta O = 0$ $\delta \overline{O} = 0$ $\delta H = 0$ $\delta \overline{H} = 0$	38.72% (=25,985,144)	A	Complete set of simple points
Lee, Chu, and Kashyap	$\delta G = 0$; $\delta O = 0$	38.72% (=25,985,144)	B	$A = B = C \cap D$
Shrihari, Ydupa, and Yau	$\delta O = 0$	90.36% (=60,638,936)	C	$A, B, \subset C$
Lobregt, Verbeek and Groen	$\delta G = 0$;	40.07% (=26,890,744)	D	$A, B \subset D$
Tsao and Fu	$\delta G = 0$; and 2 windows	19.7% (=13,220,088)	E	$E \subset A, B$
Gong and Bertrand	Simple predicates	0.10% (=65,536)	F	$F \subset E$

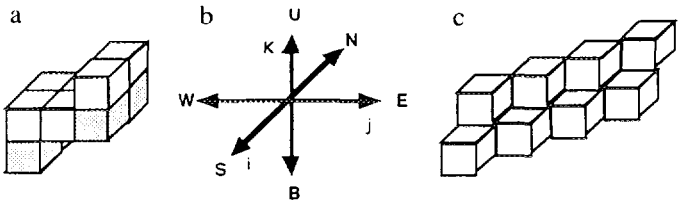

FIG. 8. Examples illustrating the need for 6 subcycles and for an additional constraint of a thinning operation (white voxels are deletable and dark ones are nondeletable).

3. 3-D PARALLEL THINNING OPERATIONS

As mentioned in the last section, points under consideration for removal by a thinning algorithm are simple border points of an object. Removing these points preserves the topological properties. There are many situations where simultaneous removal of simple border points will cause complete elimination of the original object (see Figs. 8a and 8c). The common way to eliminate this problem is to divide each thinning iteration into 6 subcycles according to six different types of border points [6, 10, 11]. A border point $v = (k, j, i)$ is of type N(north), S(south), W(west), E(east), U(up) or B(bottom) if the cubic index 13, 12, 10, 15, 4, or 21 is zero, respectively (see Figs. 1a and 8b). However, since the thinning process is carried out in parallel, the problem persists. The object in Fig. 8a or 8c serves as an example again where all of the white points are deletable because they are simple U-border points. Dark points are nondeletable because they are not U-border points. However, removing these white points simultaneously causes the object in Fig. 8a to be separated into two and the object in Fig. 8c is completely eliminated. In order to prevent situations such as these, further condi-

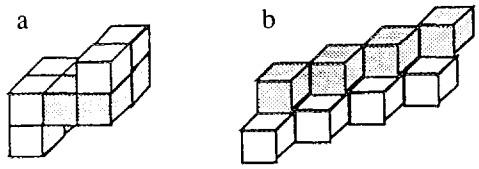

FIG. 9. (a) and (b) are the thinning results for Figs. 8a and 8c if sequential rechecking condition is used (white voxels are deletable and dark ones are nondeletable).

tions need to be checked before a simple point can be removed.

Tsao and Fu in [10, 13] use two checking windows for each thinning subcycle in an attempt to overcome this problem. To extract the skeleton, simple border points are removed if the deletion does not affect the path connectivity in the two checking windows. However, such computation is still not sufficient. For example, in Fig. 8c all the simple border points have connected checking windows. But, removing them simultaneously will cause the object to be completely eliminated.

To solve this problem, we use a sequential re-checking procedure in our algorithm to preserve connectivity. Let R denote the set of simple border points of a certain type and $Q = S - R$; i.e., the set of remaining points if R is removed from S. After all simple border points have been labeled, each point $v \in R$ is reexamined for path connectivity in $N(v)$. If points in $R \cap N(v)$ and $Q \cap N(v)$ (i.e., $O(\{R \cap N(v)\} \cup \{Q \cap N(v)\}) = 1$) are still connected, then v is removed. Since this process is repeated for the six different directions, the location of the medial line is not offset by a large amount. Shown in Figs. 9a and 9b are the thinning results for Figs 8a and 8c if sequential rechecking and end-point conditions, which is discussed next, are used.

The thinning procedure repetitively deletes the border points of an object that satisfy topological and geometric constraints. Above, we discussed the topological constraints. The geometrical constraints that have been discussed so far are important to ensure the thinness and the location of the skeleton. Depending on the desired result, either the medial surfaces or the medial axes, different geometric constraints can be used. In the next section, we survey a number of characterization for a digital surface point. A new definition for a surface point is then stated.

3.1. *Medial Surface/Axis Thinning (MST and MAT)*

The idea of medial surface thinning (or MST) is to acquire surfaces that are approximately located at the center line of S by using the thinning operation. To do that, we need a definition for surface points (or surface end points) so that an object is thinned into a skeleton whose thickness is one voxel thick in at least one of the three principle directions. There are a number of characterizations of digital surfaces. Tsao and Fu in [10] use two checking windows to check for a surface point. A point is classified as a surface point if the number of 1's in either window is less than 2. For example, all the points in Fig. 8c are classified as surface points since both the checking windows have only one 1. In [6], a point v is called a surface point if v or one of its direct neighbors is in a thin octant. An octant $N^2(v)$ is thin if v and every direct adjacent point in $N^2(v)$ is 6-adjacent to an $\bar{S}$ point in $N^2(v)$. Using Fig. 8c as an example again, all the points are in thin octants; hence they are all surface points. Morgenthaler and Rosenfeld [17] and Reed [18] call v a surface point if v is 6-adjacent to exactly two $\bar{S}$ components and every point 26-adjacent to v is adjacent to both of these two $\bar{S}$ components. Thus, they correctly classify the voxel v in Figs. 10a and 10b as surface points. But, it turns out that this definition does not produce desirable results for edge points of a surface. For example, in Figs. 10c and 10d, point v is deletable by their procedure (clearly, it should be identified as a surface end point). This classification excessively shrinks an object into a nonplane like skeleton. Gong and Bertrand also give an operational definition of a surface point [11]. A point v is not a border point of a surface if number of 1's in $N(v)$ is greater than 8 or if the number is between 4 and 7 such that at least one of the eight octants has three 6-adjacent points to v; otherwise v is a surface point. However, the reasoning behind this characterization is not clear.

From Figs. 10a and 10b, we note that a digital surface

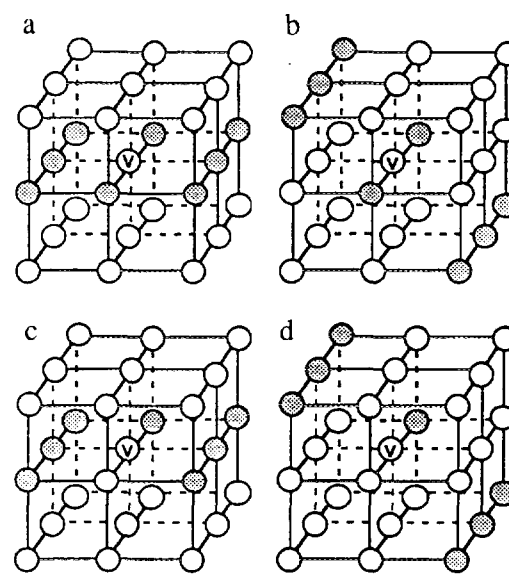

FIG. 10. (a) and (b) are surface points defined by [17, 18]; (c) and (d) are examples showing that their definitions do not work (*V*s in these examples should be all surface end points).

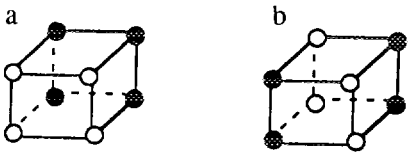

FIG. 11. (a) and (b) are two types of surfaces.

consists of two kinds of octants (as shown in Fig. 11a and 11b) which have binary configurations 11110000 (=240) and 10100101 (=165). Two other symmetric rotations are 10101010 (=170) and 11001100 (=204). Using these kinds of octants, however, will yield similar results for medial surface as in [17, 18] since edge points of a surface are not preserved. In order to preserve surface edge points, we relax the condition so that an octant may have less than three points. We now give a characterization to identify surface points that maintain the edges of the surface.

DEFINITION 1. A point v is a surface point if $\forall i$, $i \in \{1, \ldots, 8\}$,

$$\text{(a) } \text{index}[N_i^2(v)] \in \{240, 165, 170, 204\}; \text{ or} \tag{8}$$

$$\text{(b) } |N_i^2(v)| < 3. \tag{9}$$

where $N_i^2(v)$ denotes the ith octant of $N(v)$, index$[\cdot]$ is the index in base 10 of the binary configuration, and $|\cdot|$ is the number of points in the octant.

We can easily show that if a point v satisfies either Eq. (8) or Eq. (9), then v is contained in a thin octant [6]. Thus, the set of surface points is a subset of the surface points of [6]. Also, the set is a superset of the surface points as characterized in [17, 18] since their surface points are required to satisfy Eq (8).

Medial axis thinning (or MAT) differs from medial surface thinning in that the extracted skeleton consists of arcs and/or curves instead of surfaces that approximate the center line of S. Tsao and Fu in [10] follow a similar approach as in [12] by deriving an object's medial axes from its medial surfaces. An end point is defined as a point such that on the two checking windows no opposite 8-neighbor exists. However, as shown by the example in Fig. 8c, using this end-point condition does not thin the object to a desired skeleton since all the points have no opposite 8-neighbor in its checking windows; instead, it will preserve the surface (see Fig. 14e). In a later paper [13], Tsao and Fu give an operational definition of a medial axis end point which extracts the skeleton directly. Each subcycle of their thinning algorithm consists of two steps. Every simple border point is labeled at the first stage. In the second stage, nonend points are then determined. We believe that the simple definition of end point of an arc, as defined in Section 2.1, is sufficient to extract the medial axis if the topological conditions are satisfied.

Based on the topological and the geometrical conditions discussed thus far, we now present our thinning operation T_{lck}.

Let T_{lck} denotes a thinning operation with the sequence of directions U, B, N, S, W, and E (see Fig. 8b). T_{lck} repetitively deletes one type of border points of S in parallel (or simultaneously) satisfying the following conditions until no more points can be removed:

(C1) $\delta G(S \cap N(v)) = 0$

(C2) $O(S \cap N(v)) = 1$

(C3) $O(\{R \cap N(v)\} \cup \{Q \cap N(v)\}) = 1$ where R is the set of border points that satisfies (C1) and (C2) and $Q = S - R$.

(C4) (i) v does not meet the surface end point condition (Eqs. (8) and (9) in Definition 1), or

(ii) $|Q \cap N(v)| \geq 2$ where $|Q \cap N(v)|$ is the number of points in $\{Q \cap N(v)\}$.

3.2. *Experimental Results*

In this section, several examples that demonstrate the effectiveness of the proposed algorithm are given (see Figs. 12, 13, and 14). These examples are constructed using a geometric modeler called Pro/Engineer (a product of Parametric Technology Corp., MA) [19], followed by a 3-D discretizer. Pro/Engineer is a parametric, feature-based solid modeling system with Boundary Representation (B-rep) as its internal representation. Because parts are created using dimension-driven parameters and solid features (holes, slots, fillets, drafts, etc.), modifications of a geometry can be achieved with ease for "what-if" analyses. For more details about the operation of this modeler, refer to [19].

In order to carry out the thinning operation, we need to represent the volume of these objects with an array of voxels Z^3 (refer to Section 2.1 for basic definitions). The discretization process that converts Pro/Engineer's internal B-rep into voxels is done by a ray tracing and a filling operations. Rays are traced from three different planes, XZ, XY, and YZ for an accurate geometric representation. For each tracing plane, increments of resolution size are added in the two principle directions. Pro/Engineer modeler provides a function called "pro_ray_x_model" that finds intersecting points between a direction vector and a geometric entity. Each intersecting point is rounded into an integer coordinate to represent the boundary point of the object. In order to obtain a complete voxel model, a filling algorithm is needed to fill the interior of the object (refer to [20] for details).

In Fig. 12a, three L-shaped objects are assembled together using Pro/Engineer's "assemble" function, and it is converted to its voxel representation, shown in Fig. 12b, using the proposed digitizer. Since thinning operation is a non-linear process and it preserves topology, the

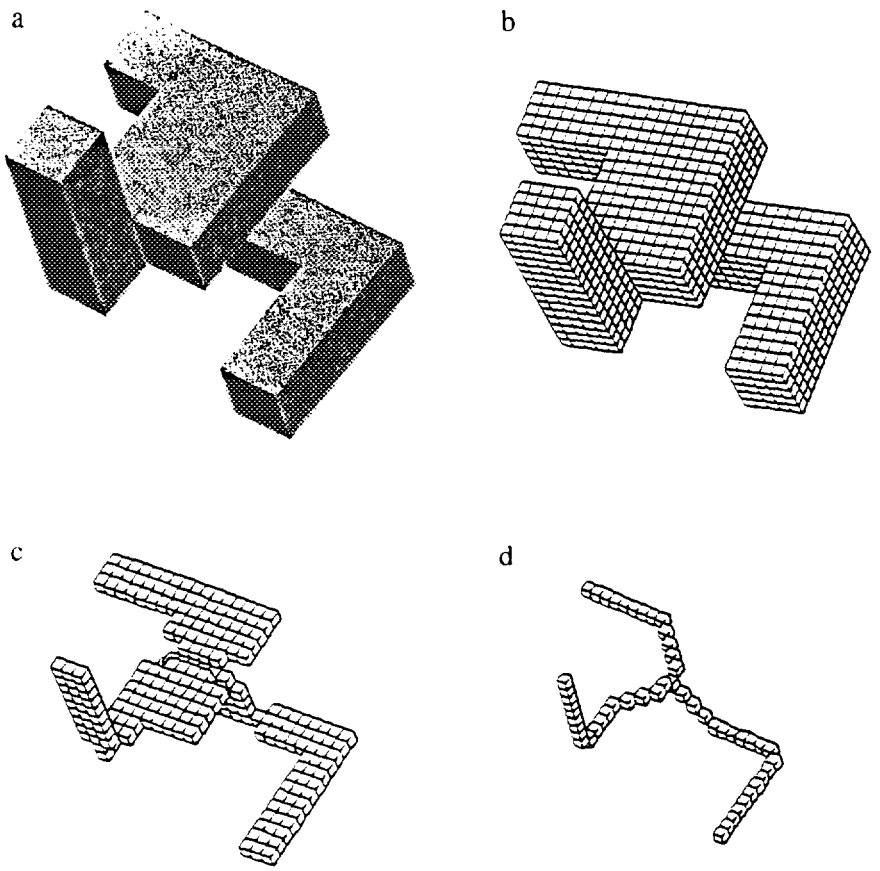

FIG. 12. (a) An object consisting three L-shaped components designed using the Pro/Engineer modeler; (b) its voxel representation; (c) and (d) are its medial surface and medial axis using T_{lck}, respectively.

skeleton of this set of unioned objects consists of skeleton of each L-shaped object connecting to each other. Figures 12c and 12d are the medial surface and the medial axis, respectively, for the part in Fig. 12b. It is clear from the result that the medial surface and the medial axis comprise of surfaces and arcs, respectively.

In general, there are two kinds of boundary noise: impulse and digitization noise. In Fig. 13a, random impulse noise is added to the surface of the three L-shaped object in Fig. 12b. A connecting rod designed using the Pro/Engineer solid modeler is displayed in Fig. 14a. The voxel representation of the connecting rod has digitization noise introduced at its boundary (Fig. 14b). By comparing the medial surface results using T_{lck} (the proposed algorithm) and T_{tf}, (the thinning algorithm in [10]) we see that T_{lck} produces slightly cleaner medial surfaces than T_{tf} (Figs. 13b and 13c, and Figs. 14c and 14d). This observation is much more obvious in the case of the medial axes. Clearly, T_{lck} produces more desirable medial axes than T_{tf} as shown in Figs. 13d and 13e, and Figs. 14e and 14f. In Fig. 14e, T_{tf} tends to preserve surfaces rather than arcs or curves, as mentioned in the last section.

In Fig. 15, the number of deleted points by the two algorithms is plotted against the number of thinning iterations for the thinning results in Fig. 14. For the medial surface extraction, it was found that T_{lck} required more iterations than T_{tf}, but with comparable CPU time on a Sun 4 workstation and comparable thinning results (see Figs. 14c and 14d). The "iteration" measurement used here is defined as a single application of the thinning process in all six (U, B, N, S, W, and E) directions. However, the main difference lies in the medial axis extraction. T_{tf} required 61 thinning iterations with 16,372 CPU seconds whereas our algorithm required only 7 iterations with 28.4 CPU seconds. This result shows the speed efficiency of our algorithm over Tsao and Fu's [10]. It was shown at the end of Section 2 that Tsao and Fu's simple point characterization only has 19.7% of the total simple point set. Hence, their algorithm would require more thinning iterations than ours because their set of deletable points in each iteration is smaller than ours. It should be pointed out, however, that thinning iteration is not the only factor that would affect the computation speed, the computation cost of determining simple nonend point should also be considered.

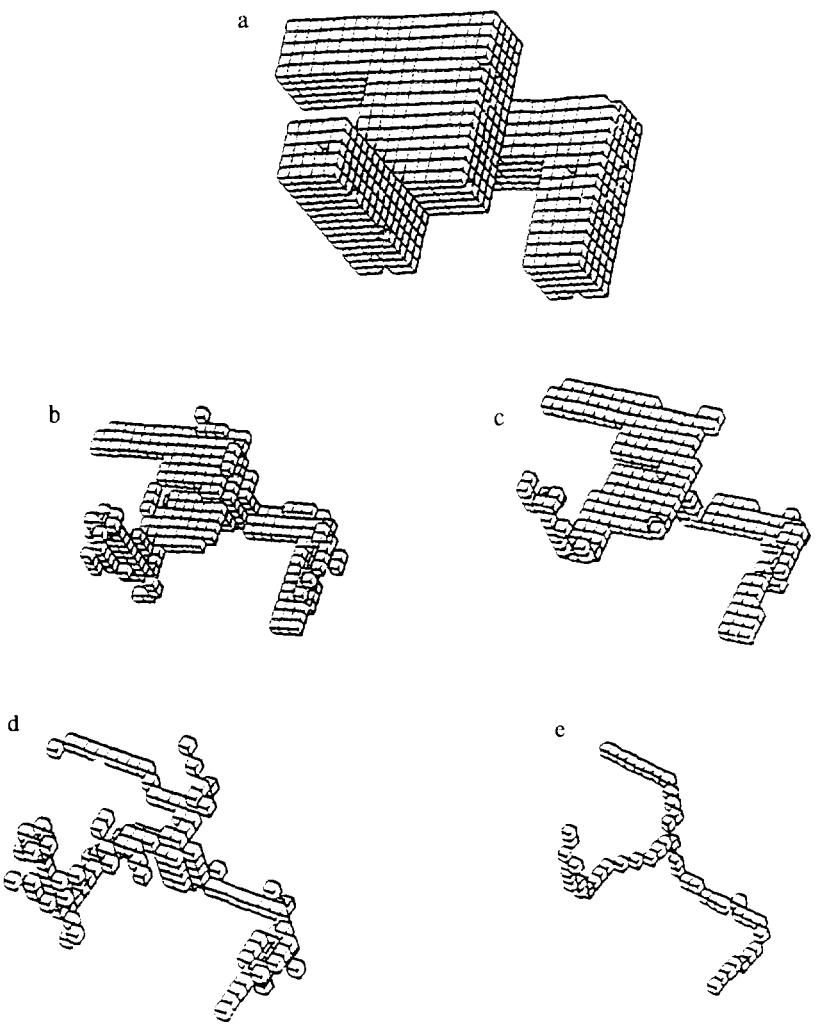

FIG. 13. (a) Noise added to the three L-shaped object in Fig. 12b; (b, c) Extracted medial surfaces using T_{tf} and T_{lck} respectively; (d, e) extracted medial axes using T_{tf} and T_{lck}, respectively.

4. SKELETON MODELING

Most of the thinning algorithms such as the ones in [6, 10, 11, 14] really do not consider the noise problems associated with thinning operations. Although these algorithms satisfy the topological and the geometrical constraints, the extracted skeletons usually do not reflect the intended geometric features in the original shape (in this section, we will use "skeleton" and "medial axis" interchangeably). In [13], Tsao and Fu characterize the effect of noise as being either undesired skeleton branches created due to noisy boundary, small holes, and/or cavities. The noise problems can be reduced by (1) preprocessing the noisy input image. (2) postprocessing the extracted skeleton to remove unwanted branches, and (3) smoothing during the thinning operation. It was shown in the last section that the proposed thinning algorithm has a smoothing effect on the resulting skeleton. In this section we consider both the pre- and post-processors. A robust skeleton is constructed using the following six-stage procedure:

(1) Preprocess the noisy input image by a 3-D digital filter, (e.g., maximum and minimum operators in [13, 21]),
(2) extract skeleton using T_{lck}.
(3) obtain a distance map of the skeleton,
(4) classify skeleton points,
(5) remove noise in the skeleton by thresholding, and
(6) construct the skeleton model.

There are many digital filters that are designed for the purpose of smoothing digital images with noise. One of the simplest local filters is the combination of maximum and minimum operators as discussed in [13, 21]. For a 3-D binary image Z^3, the local maximum and minimum operators are defined as: $S_{max}(Z^3) = \{v | v \in S \text{ or } v \in N(v') \text{ for some } v' \text{ in } S\}$ and $S_{min}(Z^3) = \{v | v \text{ not} \in S \text{ or } v \in N(v')$

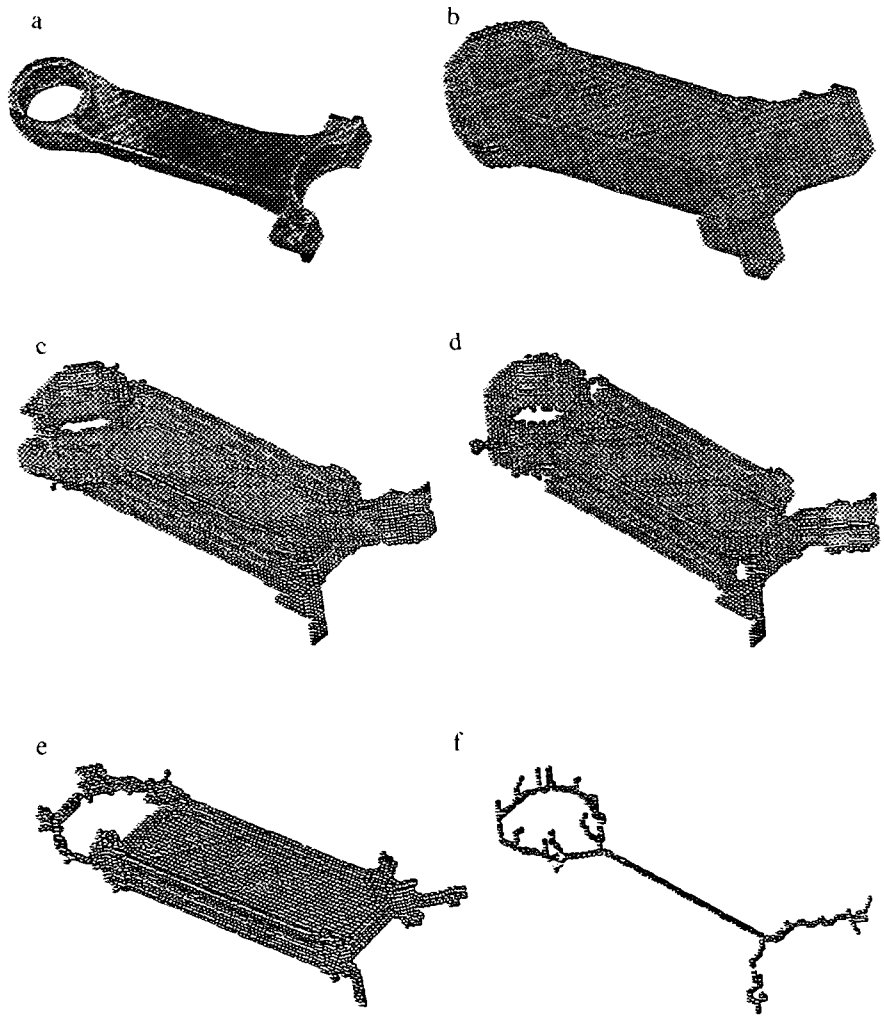

FIG. 14. (a) A connecting rod designed using the Pro/Engineer modeler, (b) its voxel representation with digitization noise; (c, d) extracted medial surfaces using T_{tf} and T_{lck}, respectively; (e, f) extracted medial axes using T_{tf} and T_{lck}, respectively.

for some v' not in S}. The ordering of the maximum and the minimum operators results in two types of filters: FU1 and FU2. They are defined as $FU1^n(Z^3) = \min^n(\max^n(Z^3))$ and $FU2^n(Z^3) = S^n_{\max}(S^n_{\min}(Z^3))$ where n is the order of FU1 and FU2 and is the number of times the operator is applied to the 3-D image Z^3. It turns out that $FU1^n$ tends to expand objects such that all the small holes or concavities with thickness less than $2n$ are eliminated. The operator, $FU2^n$, on the other hand, shrinks objects, and convexities with thickness less than $2n$ are eliminated. This is very similar to the discussion on the expanding and shrinking operators for 2-D images in [21]. The noisy 3-D objects in Figs. 13b and 14b are used as examples to test the effectiveness of the two filters. Since most of the noise on the boundary of Fig. 13b are concavities with thickness of one, it is obvious that applying first-order FU1 to the original shape produces a better skeleton than first-order FU2, as shown in Figs. 16a and 16b. However, in order

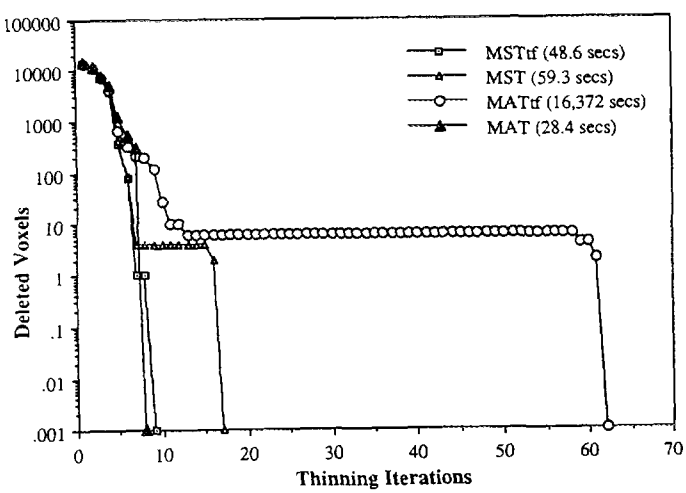

FIG. 15. Number of thinning iterations vs deleted points for the connecting rod (41,399 voxels).

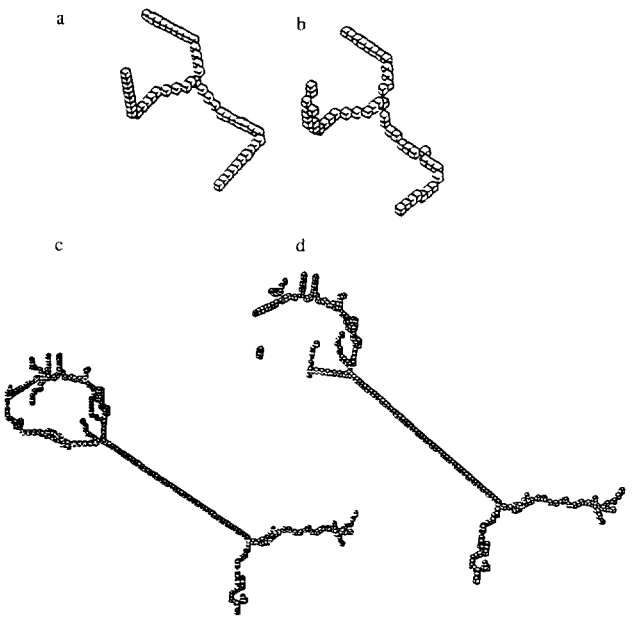

FIG. 16. (a, b) Extracted skeletons of the noisy three L-shaped object after applying first-order FU1 and FU2; (c, d) skeletons of a connecting rod after applying third-order FU1 and FU2.

to produce a desirable skeleton, we need to have some priori knowledge about the noise such as the corresponding contaminated voxels. Furthermore, not only the computation effort of applying FU1 and FU2 are large for objects consisting of large data points, the operators will not always remove spurious skeleton branches completely (Fig. 16c) and sometimes even disconnect the object if the order of the filter is not chosen correctly (Fig. 16d). Therefore, these kinds of operators need to be used with caution, and we must rely on some sort of post-processing techniques to overcome the problems with noise. After the skeleton is extracted using T_{lck}, the distance mapping algorithm expands each skeleton point to its neighboring points until the object's boundary is reached. The Euclidean distance between the first boundary point reached and the skeleton point is computed and assigned to the skeleton point. In addition to the distance information, each point of the skeleton needs to be classified according to the number of 26-neighbors it has. The classification scheme is used not only to remove noise spurs, but it forms the basis for building the skeleton model. However, it turns out that using the number of 26-adjacent neighbors does not always classify each skeleton point correctly. An example is shown in Fig. 17a where we denote regular and "T" as points with two and three 26-neighbors, respectively. For clarity we give this example in 2-D, but it is easily extended to 3-D cases. In order to build a skeleton model, there should have been only one "T" point rather than three. To correct this problem, we can classify a point with three regular neighbor points as a "T" point and label the point as "T" type immediately, and any 26-adjacent neighbors of this "T" point will be considered as regular points. In addition, we will classify a point with four regular neighbors as an "X" point. The result of using this simple scheme is shown in Figs. 17b and 17c. However, this scheme is insufficient for other types of classifications such as the one shown in Fig. 17d. In this example, we will need a more robust verification procedure that can determine the number of arcs connecting to the "T" point.

After verifying each skeleton point, we threshold the result by removing those branches with length less than a pre-defined length l (the post-processor). Shown in Fig. 18a is the skeleton in Fig. 14f with noise that has been effectively removed by using this simple operator with $l = 10$. However, the operator does not always yield desirable results because the predefined length must be determined in advance. After removing those noisy skeleton edges (or branches), each skeleton point needs to be reclassified to ensure the correctness of the model (see Fig. 18b).

Based on the adjacency and the radial distance information, we construct a skeleton model of Fig. 18b in Tables 4 and 5. Table 4 shows the result by applying the classification procedure to the skeleton in Fig. 18b. The "Distance" column represents the Euclidean distance between the skeleton point and its nearest boundary point. For clarity, we omit points with exactly two 26-neighbors (i.e., regular points). An Edge table is constructed for retaining

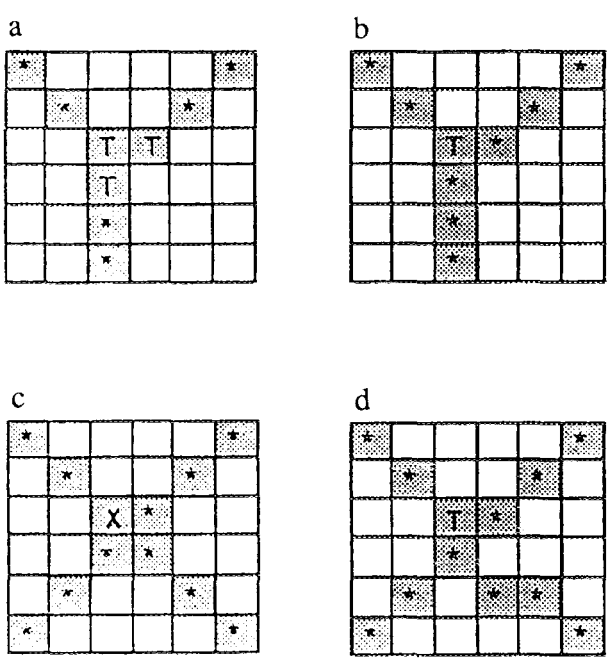

FIG. 17. (a) An example of misclassified point type (* are regular points); (b, c) corrected classifications; (d) an example where the proposed scheme is insufficient.

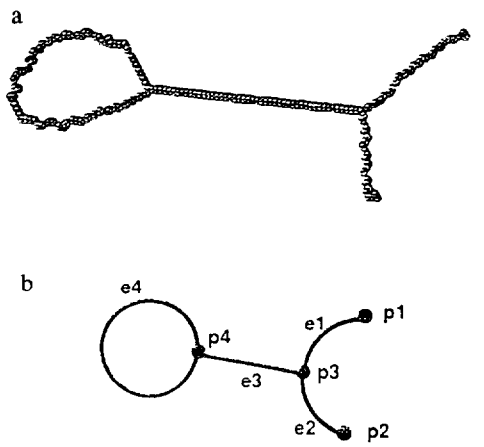

FIG. 18. (a) Skeleton in Fig. 14e with noise removed by the thresholding operation; (b) skeleton with points and edges classified.

the adjacency information (see Table 5). Both the "Point" and the "Edge" tables constitute the so-called skeleton model.

5. APPLICATIONS OF THE SKELETON MODEL IN MANUFACTURING

In the past decade, many commercial computer-aided engineering (CAE) analysis software tools have developed for engineering analysis purposes. However, these facilities are often not fully integrated, and they require human interaction, not to mention the lengthy preparation and simulation time that might be required. Part of the reason is that part descriptions of 3-D CAD models are not represented in a form which can be utilized directly in manufacturing analysis. In order to overcome these problems, geometric reasoning based on a high-level representation needs to be carried out. The skeleton model discussed in the last section can be considered as a simplified representation of a geometry that preserves topology. In this section, we propose to use the skeleton model to predict casting and forging defects.

5.1. *Casting Defect Analysis*

Casting is one of the most popular methods for achieving a desired complex shape in the metal processing industry. Parts (e.g., connecting rods) are made by pouring hot liquid metal into the mold cavity. When solid, metal is the shape of the part [22]. Due to its simplicity, the solidification modulus (SM) value has been widely used to determine the relative freezing time among various segments of a casting for evaluating global casting soundness [22, 23]. The SM calculation is based on the subdivision of a complex part into simple subcomponents and its value is defined as the ratios of the volume over the surface area of each subcomponent. Intuitively, regions of a casting with a low SM value cool faster than those with high SM value because a low-SM region has a greater surface area than a region with high SM value. Therefore, it is assumed that the SM of a section must be greater than its neighboring SM in order to act as a feeder to its neighboring sections; otherwise, the flow of liquid metal will be blocked. By comparing SM among subdivided regions, the location for the riser (a riser is defined as a reservoir of liquid metal [23]) can be determined. The task of dividing a part into sub-components is, however, not fully determined. Since most complex castings are made up of a combination of simple geometries, such as T, X, L, ribs, bosses, recognition of these geometries can be considered as a first step in solving this subdivision problem.

Kotschi and Loper [23] have generated graphical data for T and X sections of castings by using computer analysis based on the SM concept. Consider the T shape shown in Fig. 19 used in their study. The subdivision is placed at a length equal to the thickness of each arm away from the radius R. The order of solidification of these regions can be determined from its SM values. Using the heuristics provided by them and the distance mapping information in the skeleton model, we can determine the thickness of each arm and locate places to sub-divide castings into subcomponents for SM analysis.

Let us take the connecting rod in Fig. 14b as an example. Using our medial axis thinning operation T_{lck} and the proposed postprocessing technique we extracted a clean skeleton in Fig. 18a. Though the radius information is available in the skeleton model, it cannot be considered as the true thickness of the arms in the connecting rod due to variations of its cross sections. Instead, we should compute the SM value for the cross section of each skeleton point in the component. The averaged SM value of

TABLE 4
Point Table for the Medial Axis of the Connecting Rod

Point number p_i	Coordinate (x, y, z)	Distance	Point type
p_1	(6, 4, 11)	4.1	E
p_2	(44, 6, 11)	4.6	E
p_3	(26, 22, 10)	7.2	T
p_4	(26, 78, 10)	7.3	T

TABLE 5
Edge Table for the Medial Axis of the Connecting Rod

Edge number	End points (p_i, p_j)	Length
e_1	(p_1, p_3)	28.5
e_2	(p_2, p_3)	25.1
e_3	(p_3, p_4)	48.0
e_4	(p_4, p_4)	84.5

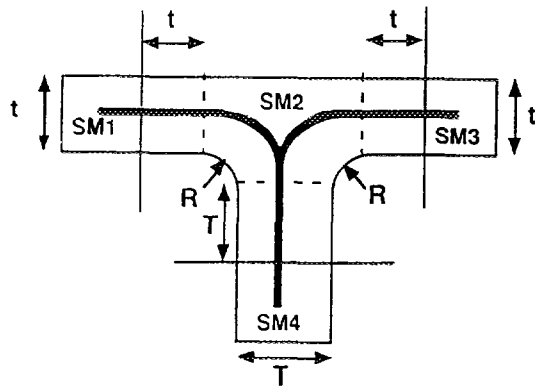

FIG. 19. T section composed of a supporting arm of thickness T and a cross arm of thickness t, joined with fillets of radius R [23]. The skeleton is drawn in dark lines in the center of the T section.

each arm is used in the equation SM = $2/T$ to obtain the averaged thickness t or T [24]. We then determine a cutting plane offset by the averaged thickness $t + 1/2\ T$ from the "T" junction point as shown in Fig. 19. SM values are computed for each subcomponent, and risers can be recommended for components that have large SM values compared to its neighboring components. See Fig. 20 for an illustration of such computation. Details can be found in [24].

5.2. *Forging Defect Analysis*

Forging is a manufacturing process that deforms heated metal parts into a specific pattern using a hydraulic press or a power hammer [25]. The primary objective of forging design is to guarantee sufficient metal flow so that initial part geometry can be obtained without any defects. Two of the most important design features in forging are the rib and the web [25]. A rib is a projection from the web that is either located at the periphery or within a forging part in a direction parallel to the ram stroke. The web of a forging is the relatively thin section of the forging that lie between ribs and other forged elements extending from the surfaces of the web (see Fig. 21 for the rib and the web of a connecting rod). The ease of manufacturing a rib depends mainly on the ratio of its height and width. A deeper and thinner rib usually demands greater forging pressure. An important rule of thumb in forging design is that rib width should never be more than the web thickness; otherwise, the flow of the material filling the rib may cause a void in the web directly below the base of the rib [25]. Moreover, there are limits on the minimum web thickness because thin webs require greater forging pressure and tend to cause defects. To apply these heuristics for defect detection, ribs and webs must first be identified in a forging design.

Marefat and Kashyap in [26] propose an attributed adjacency graph and a subgraph matching scheme for feature detection (see [26] for other approaches). Although their subgraph matching approach can be extended to determine forging features such as ribs and webs for parts consisting of planar surfaces, it is difficult to determine these features for surfaces with free form (or sculptured) nature because of the representational difficulties. In fact, many of the forging geometries consist of free-form surfaces rather than polygonal faces (e.g., the connecting rod in Fig. 21). Medial surface model, however, is invariant to surface curvature and can be used to detect rib and web features for free-form objects. After the medial surface is extracted using our thinning operation, surface patches need to be determined for the model. The problem of forming surface patches boils down to the task of grouping points that lie on the same surface. The approach in [27] can be used to group points that belong to each surface

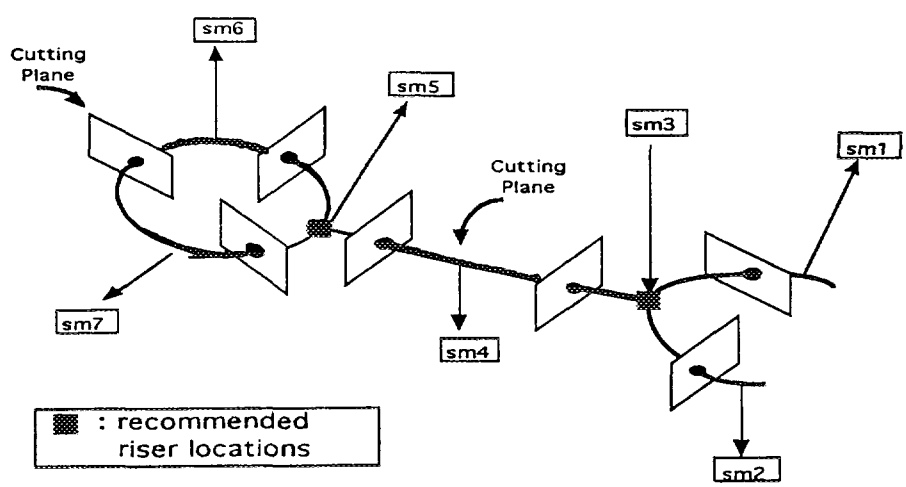

FIG. 20. Computations of section modulus and recommendation of riser locations [24].

patch by using principle direction frames or by using similar Gaussian and mean curvature. To obtain these local characteristics requires estimation of tangent and curvature information at these points.

After surface points have been classified into different pathces, a topological graph showing the relationship between those patches can be formed. In Fig. 22a, topological graphs for a web–rib–web object is illustrated. The surface normal of each surface patches is in one of the principal directions. Using the normal direction, the medial surface that correspond to the rib can be determined as perpendicular to the forging direction (surface *S*2). A web, on the other hand, has normal direction parallel to the forging direction (surfaces *S*1 and *S*3). With the radius of the associated medial surface model, an average thickness can be acquired for each surface patch. Multiple web–rib–web regions can also be detected by using a simple graph matching technique as shown in Fig. 22b. It is easily seen from Fig. 22 that medial surface is invariant with surface curvature because polygonal objects have the same medial surface models as its free-formed counterpart. With this knowledge, a wider coverage for different rib and web geometry can be achieved.

6. CONCLUSION

We have presented a parallel thinning algorithm that can be used to extract either the medial surface or the medial axis of a 3-D object. The preliminary results have shown that our algorithm produces more desirable skeletons than existing algorithms. Pre- and postprocessors can be used to reduce noise spurs in the skeleton model. The constructed models are proposed for predicting forging and casting defects. Because of its descriptive power of free-formed surfaces, skeleton model, in our belief, has great potential in providing manufacturing features that are needed in an integrated design and manufacturing environment. However, further work is certainly needed in several areas in order to fully establish such integration. More robust pre- and postprocessors are needed. A verification scheme that can handle possibly misclassified points in medial axis is also warranted. A closer study of digital surfaces is essential for building a noise-free medial surface model.

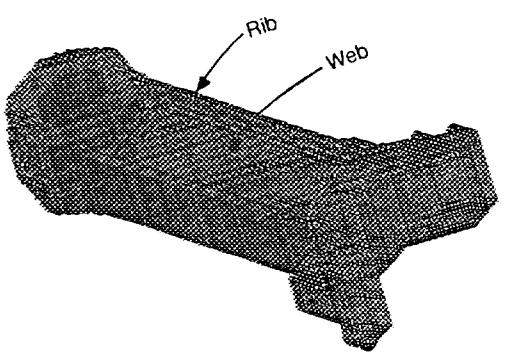

FIG. 21. Rib and web of the connecting rod in Fig. 14b.

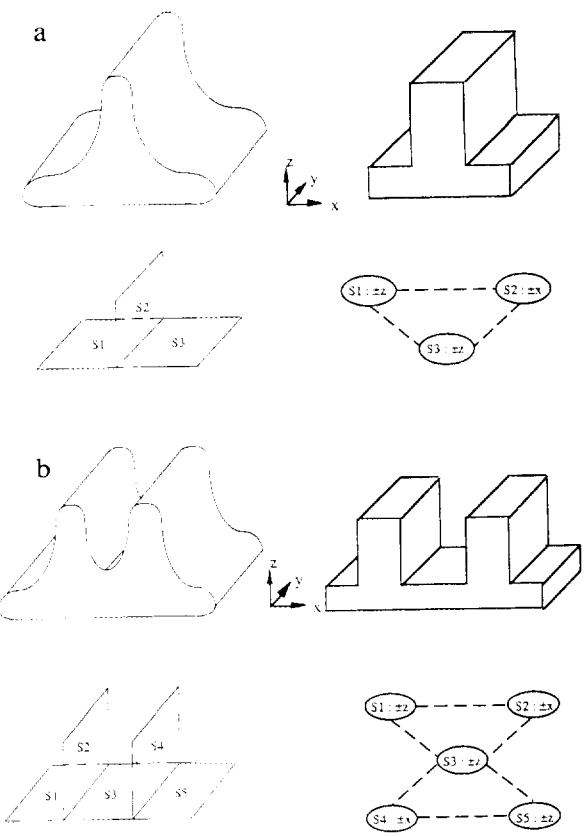

FIG. 22. Topological graph for ribs and webs.

REFERENCES

1. L. Lam, S. W. Lee, and C. Y. Suen, Thinning methodologies—a comprehensive survey, *IEEE Trans. Pattern Anal. Mach. Intell.* **14**,(9), 1992, 869–885.
2. S. N. Srihari, Representation of three-dimensional digital images, *ACM Comput. Surveys* **13**,(4), 1981, 399–424.
3. A. Kaufman, Efficient algorithms for 3D scan-conversion of parametric curves, surfaces, and volumes, *Computer Graphics* **21**, 1986, 303–329.
4. D. G. Morgenthaler, *Three-Dimensional Digital Topology: The Genus*, Technical Report TR-980, Computer Vision Laboratory, Computer Science Center, Univesity of Maryland, College Park, 1980.
5. T.Y. Kong and A. Rosenfeld, Digital topology: Introduction and survey, *Comput. Vision Graphics Image Process.* **48**(3), 1989, 357–393.
6. D. G. Morgenthaler, *Three-Dimensional Simple Points: Serial Erosion, Parallel Thinning, and Skeletonization*, Technical Report TR-1005, Computer Vision Laboratory, Computer Science Center, University of Maryland, 1981.

7. S. Lobregt, W. Verbeek, and F. C. A. Groen, Three-dimensional skeletonization: Principle and algorithm, *IEEE Trans. Pattern Anal. Mach. Intell.* **2,** 1980, 75–77.
8. C. R. F. Maunder, *Algebraic Topology*, Cambridge Univ. Press, Cambridge, 1980.
9. T. Y. Kong and A. W. Roscoe, A theory of binary digital pictures, *Comput. Vision Graphics Image Process.* **32,** 1985, 221–243.
10. Y. F. Tsao and K. S. Fu, A parallel thinning algorithm for 3-D pictures, *Comput. Graphics Image Process.* **17,** 1981, 315–331.
11. W. Gong and G. Bertrand, A simple parallel 3-D thinning algorithm, in *IEEE 10th International Conference on Pattern Recognition*, 1990, pp. 188–190.
12. S. N. Srihari, J. K. Udupa, and M. M. Yau, Understanding the bin of parts, in *Proceedings of the IEEE Conference on Decision Control*, 1979, pp. 44–49.
13. Y. F. Tsao and K. S. Fu, A 3D parallel skeletonwise thinning algorithm, in *IEEE Conference on Pattern Recognition and Image Processing*, 1982, pp. 678–683.
14. G. Malandain and G. Bertrand, Fast characterization of 3D simple points, in *IEEE International Conference on Pattern Recognition*, 1992, pp. 232–235.
15. C. L. Jackins and S. L. Tanimoto, Oct-trees and their use in representing 3D objects, *Comput. Graphics Image Process.* **14,** 1980, 249–270.
16. C. M. Park and A. Rosenfeld, *Connectivity and Genus in Three Dimensions*, Technical Report TR-1005, Computer Vision Laboratory, Computer Science Center, University of Maryland, 1971.
17. D. G. Morgenthaler and A. Rosenfeld, Surfaces in three-dimensional digital images, *Inform. and Control* **51,** 1981, 227–247.
18. G. M. Reed, On the characterization of simple closed surfaces in three-dimensional digital images, *Comput. Graphics Image Process.* **25,** 1984, 226–235.
19. *Pro/Engineer User's Guide*, Parametric Technology Corporation, Waltham, MA, 1987–1992.
20. T. C. Lee, *CAD/CAM Integration via Skeleton-Based Modeling*, M.S. thesis, Department of Electrical Engineering, Purdue University West Lafayette, IN, 1993.
21. A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, 2nd ed., Vol. 2, Academic Press, New York, 1982.
22. P. F. Wieser, *Steel Castings Handbook*, Steel Founders' Society of America, 1980.
23. R. M. Kotschi and C. R. Loper Jr., Design of T and X sections for casting, *AFS Trans., 1974, 535–542.*
24. C. N. Chu, R. L. Kashyap, I. C. You, and T. C. Lee, *CAD/CAM Integration for Casting Design*, Technical Report TR-ERC 91-1, Engineering Research Center for Intelligent Manufacturing Systems, Purdue University, 1991.
25. S. A. Sheridan and P. M. Unterweiser, *Forging Design Handbook*, American Society for Metals, Metals Park, OH, 1972.
26. M. Marefat and R. L. Kashyap, Geometric reasoning for recognition of three-dimensional object features, *IEEE Trans. Pattern Anal. Mach. Intell.* **12,**(10), 1990, 949–965.
27. P. T. Sander and S. W. Zucker, Tracing surfaces for surfacing traces, in *First Intern. Conf. on Computer Vision*, 1987, pp. 241–249.