CVGIP: GRAPHICAL MODELS AND IMAGE PROCESSING
Vol. 54, No. 5, September, pp. 420–437, 1992

Generating Skeletons and Centerlines from the Distance Transform

C. WAYNE NIBLACK

IBM Almaden Research Center, San Jose, California

PHILLIP B. GIBBONS

*AT&T Bell Laboratories, Murray Hill, New Jersey**

AND

DAVID W. CAPSON

McMaster University, Hamilton, Ontario, Canada

Received January 28, 1991; accepted May 21, 1992

We describe an algorithm for generating connected skeletons of objects in a binary image. The algorithm combines essentially all desirable properties of a skeletonization method: (1) the skeletons it produces have the same simple connectivity as the objects; it is based on a distance transform, and can use any "natural" distance metric (in particular those giving a good approximation to the Euclidean distance), resulting in skeletons that are both (2) well-centered and (3) robust with respect to rotation; the skeletons allow the objects to be reconstructed either (4) exactly or (5) approximately to within a specified error; (6) for approximate reconstruction, the skeletons are insensitive to "border noise" without image prefiltering or skeleton postpruning; (7) the skeletons can be thin; (8) the algorithm is fast, taking a fixed number of passes through the image regardless of the width of the objects; and (9) the skeletons have a pleasing visual appearance. Several of these properties may conflict. For example, skeletons cannot always be both thin and allow exact reconstruction, and our algorithm can be run to give priority to either property. This paper describes the skeletonization algorithm, discusses the tradeoffs involved, and summarizes the formal proofs of its connectivity and reconstructability properties. Because the algorithm is fast, robust, flexible, and provably correct, it is ideally suited for many of the applications of skeletonization—data compression, OCR, shape representation, and binary image analysis. The quality of the skeletons produced is demonstrated with numerous examples. © 1992 Academic Press, Inc.

1. INTRODUCTION

We are concerned with generating skeletons of objects in a binary image, where both the skeletons and the objects are defined on a rectangular grid. A good definition of a skeleton is surprisingly difficult. We define a skeleton of an object to be a subset S of the object that (1) has the same simple connectivity[1] as the object, (2) allows the object to be reconstructed to within a specified, known error along its border, where the reconstruction is done using only S and the values from a distance transform along S, and (3) is thin in the sense that any pixel in S is either required for the reconstruction or removing it changes 8-connectivity (causes a break or adds a hole). Several comments on this definition: First, the three parts of the definition essentially say that the skeleton is (1) connected, (2) centered, and (3) thin. These are the most basic properties of a skeleton, and are used by various authors, e.g. [1, 2]. Second, the definition is relative to the definition of the distance transform, and changing the distance transform can change the resulting skeleton. Moreover, it does not imply a unique skeleton even for a given distance transform. Third, it is well known that the three characteristics—connected, centered, and thin—are often conflicting for shapes on a digital grid. This definition makes the common assumption that connectivity must be preserved, whereas reconstructability and thinness can be relaxed. Thicker skeletons are permitted where necessary for connectivity, or to achieve a specified precision in reconstruction. Conversely, approximate reconstruction is permitted so that a desired degree of thinness can be achieved. Requiring exact reconstruction typically leads to thick (two-pixel-wide) skeletons in places of even width in an object [3] or to other solutions,

* Part of this work was completed while the author was at Stanford supported by NSF Grant CCR-86-10181, IBM Micro Grant 442427-57449, and ONR/DARPA Contracts N00014-87-K-0828 and N00014-88-K-0166.

[1] Connectivity means the skeleton has the same number of components as the object. Simple connectivity means it also has the same number of holes. Alternatively we may say, as in [1], that the skeleton and object are *homotopic*.

1049-9652/92 $5.00

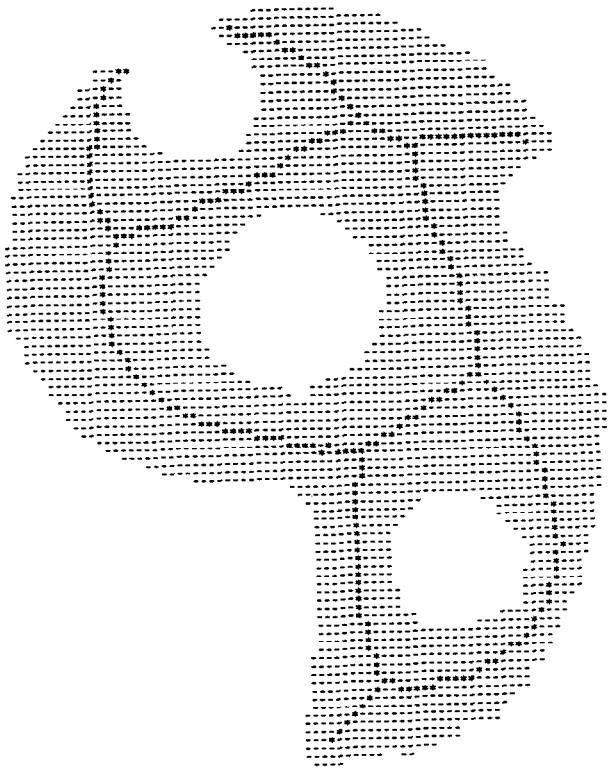

FIG. 1. Example image and skeleton. Object pixels are represented by -'s if they are not in the skeleton and by *'s if they are in the skeleton. The object is taken from [6] © 1985 IEEE.

such as skeletons that can go between the pixels, for example on a "derived grid" [4, 5]. Moreover, requiring exact reconstruction often leads to skeletons with many hairs, particularly in objects with irregular borders.

The three properties mentioned (connected, centered, and thin) are not the only desirable properties of a skeleton. Others include robustness to rotation, allowing exact reconstruction when desired, fast computability, and visually pleasing appearance, e.g., few unnecessary hairs, jags, or "zippers" (zig-zag lines in a smooth section of the object). The algorithm we describe attempts to combine all of these. As we will show, we are able to prove the connectedness and reconstructability properties, and we take steps to achieve thinness. Although we cannot guarantee perfect thinness (we will discuss this later), we still use the term "skeleton" to describe the result. Figure 1 gives an example of a skeleton produced by the algorithm.

Many methods for skeletonization have been presented in the literature and they can be grouped into several classes. One class repeatedly peels off the border of an object in a sequence of raster scans. An early paper is by Hilditch [7] who also describes a method for completing one entire peel in a single raster pass. Her method is sequential in the sense that the processing at each pixel requires knowledge of previously processed pixels in the same pass. Many related algorithms have been presented, and both [8] and [9] summarize and compare numerous methods.

Parallel methods (those which do not require results from previously processed pixels within the same iteration and are therefore suitable for parallel and pipelined architectures) are also common [10–12]. In their basic form, these require four subpasses to peel off one complete border layer and still preserve topology, where each subpass peels from one direction—top, bottom, left, and right. Two subpass methods are possible by combining the top and left, and bottom and right passes [13–16], or by alternately thinning two "subfields" of the image [17]. A one subpass method has also been described [18].

Another class of (sequential) algorithms also repeatedly peels off border pixels, but does so by maintaining a list of border pixels of the object, and processing the list, moving randomly through the image along the object border, as opposed to moving in a raster scan. Pixels along the object border are removed except for special pixels, often called contour points or multiple pixels, that must be kept to preserve connectivity. When only these points remain, the skeleton has been constructed. Such algorithms are usually faster than raster algorithms when implemented on a sequential computer. Examples are [19–24]. For all the above methods, the connectivity of the final skeleton is ensured by defining the peeling conditions so that no pixel is removed that breaks local connectivity.

Related to the parallel methods are morphological skeleton methods, based on the set operations of erosion and dilation using a structuring element. The result is a morphological skeleton, which is somewhat different from the skeleton as defined above in that it is typically not connected. It does allow for object reconstruction [25, 26]. An exception which does produce a connected skeleton is [1], where specific structuring elements are used, giving an algorithm similar to the parallel peeling methods above.

A final class of algorithms is those based on the distance transform and the related medial axis transform (MAT)[2] [27]. These are also sequential methods, and one of their main advantage is that they are noniterative. The skeleton is produced in a fixed number of passes through the image regardless of the object sizes. It is well known that the set of local maxima in the distance transform produces a skeleton-like representation of an object, but that it is not in general connected. The main problem addressed by distance transform and MAT-based algorithms is how to suitably reconnect the local maxima to

[2] In this paper, we use the common, computational definition that the medial axis transform is the set of local maxima in the distance transform.

produce a skeleton. The algorithm we describe is of this type.

An important choice is these algorithms is the metric used in the distance transform. This will directly affect the centering of the skeleton and its sensitivity to rotation. We use metrics which provide a good approximation to the Euclidean distance, for example the "chamfer 2–3," or 2–3, metric [28]. This metric assigns a distance of 2 between horizontally or vertically adjacent pixels, and 3 between diagonally adjacent pixels. This keeps the ratio close to the optimal value [28] and still allows the computation to be done using fixed point arithmetic. Another (and better, according to [28]) is the "chamfer 3–4" or 3–4 metric, which we also use. Other common choices, but which we feel do not give the advantages of the 2–3 or 3–4 metrics, are the $d8$ (chessboard) and $d4$ (city block) metrics. Note also that by using a weighted distance transform, the method becomes more general than the morphological methods (e.g., [1, 25, 26]) using iterative morphology. This is true because a fixed structuring element effectively defines as constant the distance from all pixels in the element to the pixel on which the element is placed, and to give results comparable to a weighted distance transform, a "weighted structuring element" would be required.

Previous work on the distance transform-based and MAT-based skeletons has been reported by Montanari [29] (he used the term "quasi-Euclidean metric") although he was not concerned with the skeleton connectivity, Danielson [30] (who uses a nearly exact Euclidean distance metric), Arcelli and di Baja [6] (who use $d8$), Gong [5] ($d4$ metric on a derived grid), and Dorst [31] (who also uses the 2–3 metric). As compared to these methods, the main advantage of our approach is the combination of (1) complete proofs of both the connectivity and the hole-surrounding property of the resulting skeleton when any "natural" metric is used, (2) a proof of reconstructability, exact if required, but (preferably) to within a specified error tolerance, and (3) especially compared to methods using $d4$ or $d8$, insensitivity to rotation. When approximate reconstruction is allowed, typically to within one or two pixels, we can also add (4) natural immunity to border noise, and (5) thinness (meaning usually single-pixel-wide skeletons) without pre- or postprocessing. A preliminary version of this work [32] described an algorithm that had property (1), and this paper completes and extends it to include the remaining properties.

The remainder of this paper is organized as follows. We describe the algorithm in a series of three stages, each designed to achieve a particular property of skeletons: connectivity, reconstructability, and then thinness. First, in Section 2, we describe the basic algorithm for generating connected skeleton-like subsets of objects in binary images. The connectedness of the resulting subset is not obvious, and in Section 3 we summarize the proofs that the set of pixels selected has the same simple connectivity as the set of objects in the image. These proofs hold regardless of the (integer) distance metric used by the algorithm. Second, in Section 4, we describe an extension to this algorithm to allow for exact reconstructability, or approximate reconstructability to within a specified tolerance, and prove its properties. Third, in Section 5, we describe techniques for thinning the set of pixels selected. These techniques achieve visually pleasing results while maintaining a known error bound in the reconstruction. Section 6 summarizes the overall algorithm for producing skeletons. Section 7 describes how the algorithm may be modified to produce "centerlines," skeleton-like subsets of objects that result from combining the basic algorithm for achieving connectivity (Section 2) with the techniques for thinning (Section 5), but without the extensions for reconstructability. These have been useful in certain image analysis applications needing only topological information. Section 8 contains sample results, Section 9 discusses issues relevant to many skeletonization algorithms such as robustness to rotation and computational effort, and Section 10 has conclusions.

2. THE BASIC ALGORITHM

We begin with an informal description of the algorithm. We assume that the (rectangular) image is "padded" with background pixels so that no object touches the image border.

Similarly to other noniterative algorithms, we first compute the distance transform, then select a subset M of pixels (for now, this is the set of local maxima in the distance transform, but we will expand it later) and finally select pixels to connect the pixels in M. As part of this last step, we identify pixels, called *saddle points*, whose neighborhoods contain alternating humps and valleys. For example, a pixel with distance transform value x such that clockwise around its eight neighbor pixels there are pixels with values $>x$ (part of a hump), $<x$ (valley), $>x$ (hump), and $<x$ (valley) is a saddle point. We connect the set of pixels in M by climbing "uphill," i.e. along paths of increasing distance transform values, from saddle points and neighbors of local maxima.

In more detail, our algorithm strategy consists of selecting, in order, the following:

1. All pixels in M; that is, all local maxima in the distance transform.

2. All pixels along steepest uphill paths that start from a local maximum. It may seem odd that we can climb "uphill" from a local maximum, but this is possible given that the first step is to an equal-valued neighbor. See the example in Fig. 2(a).

3. All saddle pixels.

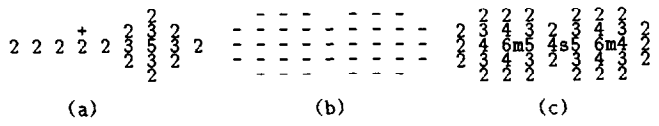

FIG. 2. (a) Example showing a local maximum (the pixel below the +) from which it is possible to climb "uphill" (toward the 5). Note the first step is to an equal-valued neighbor. (b) Example object. (c) Its distance transform using the 2–3 metric. The two (strict) local maxima are marked m and the sole 1 × 1 saddle point is marked s.

4. All pixels along steepest uphill paths that start from a maximum pixel on a hump in a saddle. The first step may be to an equal-valued neighbor.

The intuition as to why this strategy succeeds in connecting the pixels in M from a single object is as follows. Consider two local maxima, m_1 and m_2, with no local maxima between them. The distance transform values either (a) increase from m_1 to m_2 (or vice versa), after an initial step of equal value (again, this is the example of Fig. 2(a)), or (b) decrease from m_1 to some low point l and then increase to m_2. In this latter case, m_1 and m_2 can be connected by two uphill paths from l. A saddle represents a best choice for a low point since the valleys indicate that the saddle is centered with respect to the object, and the humps lead to distinct local maxima.[3]

Similar strategies are used in existing skeletonization algorithms (e.g., [5, 31]). Within this strategy, there are four important choices to be made that affect the connectedness, centering, reconstructability, rotation robustness, and border noise tolerance of the skeletons, as well as the running time of the algorithm:

- The metric used to compute the distance transform. We consider the 2–3 and 3–4 metrics to be preferable to the more common 1–1 or 1–2 metrics (i.e., $d8$ or $d4$).
- The definition of the set M. We show that when M is the set of local maxima, we obtain skeleton-like objects, while an expanded definition is needed to guarantee reconstructability.[4]
- The definition of a saddle point. We show that a relatively strict definition suffices to connect the local maxima in an object.
- The definition of a steepest uphill path. This includes the criteria for tie-breaking and when to initiate more than one climb out of a pixel.

In the remainder of this section, we present a precise description of our basic algorithm, focusing on the four areas discussed above. We use standard definitions for an *image border pixel* (a pixel with less than eight 8-neighbors), a *direct neighbor* or *D-neighbor* (a 4-neighbor), an *indirect neighbor* or *I-neighbor* (an 8-neighbor that is not a direct neighbor), a *local maximum* (a pixel with nonzero value x such that each of its 8-neighbors is at most x), a *strict local maximum* (each of its 8-neighbors is less than x), and a *nonmaximum* (a pixel not a local maximum). For a given metric, let d_D denote the distance between two D-neighbors and d_I denote the distance between two I-neighbors. If $d_D < d_I$, we say the metric is *weighted*. Let $d(p, q)$ denote the grid distance between pixels p and q as weighted by the distance metric; i.e., each vertical/horizontal step is distance d_D and each diagonal step is distance d_I. The *distance transform value* of a pixel p is $\min\{d(p, q) | q \text{ is not in any object}\}$.

Figures 2(b) and (c) show an example object and its distance transform values when the 2–3 metric is used. In this paper, we allow all "natural" distance metrics, i.e., all integer metrics such that $0 < d_D \leq d_I \leq 2d_D$. In practice, we prefer the 2–3 or 3–4 metrics.

A *witness* for a pixel p (with value x) is any D-neighbor with value $x - d_D$ or any I-neighbor with value $x - d_I$. Thus a witness for a pixel p is any 8-neighbor from which it derives its distance transform value, or, as in [34], any 8-neighbor pixel through which distance information "flows" to p.

We define humps, valleys, and saddle points as follows.

DEFINITION 1. Let $x > 0$ be the distance transform value of a pixel p and consider the cyclic sequence, N_p, of the eight neighbors around p. A *hump* of p is a consecutive subsequence, $n_1, n_2, \ldots, n_k$, $k \geq 1$, of N_p such that either

- the value of each member of the subsequence is greater than x but the value of the neighbor in N_p preceding n_1 and the neighbor in N_p succeeding n_k are both at most x, or
- the value of each member of the subsequence is x but the value of the neighbor in N_p preceding n_1 and the neighbor in N_p succeeding n_k are both less than x.

A *valley* of a pixel p is a consecutive subsequence of neighbors not in a hump of p but whose preceding and succeeding neighbors in N_p are each in a hump.

Each nonzero pixel has at least one witness, and any pixel with at least one neighbor with greater or equal distance transform value has at least one hump and one valley. Thus if a nonzero pixel is not a strict local maximum, then it has at least one hump and at least one valley.

One of our key definitions is that of a saddle point, which includes both a 1 × 1 saddle (similar, for example, to [31]) and a 2 × 2 saddle. The latter is less well known. Empirically, we found 2 × 2 saddles necessary to main-

[3] Except in pathological cases (see Section 8).

[4] This latter set, which we call N^* pixels, is discussed in Section 4.

tain connectivity as we developed the algorithms, and subsequently also found them necessary in the connectivity proofs.

DEFINITION 2. A *1 × 1 saddle point* is a nonmaximum pixel with distance transform value $x > 0$ that has more than one hump. A *2 × 2 saddle point* is a set of four nonmaxima pixels all with distance transform value $x > 0$ in a 2 × 2 such that the 2 × 2 has more than one hump, but none of the four member pixels is a 1 × 1 saddle point. (A hump of a 2 × 2 with value x is a consecutive subsequence of the 12 neighbors of the 2 × 2 satisfying the first criterion above for a hump.) A pixel is a *saddle pixel* if it is a 1 × 1 saddle point or a member of a 2 × 2 saddle point.

Clockwise in the neighborhood of a saddle point with value x there are values $>x$, $<x$, $>x$, $\leq x$ or values $>x$, $<x$, x, $<x$. A saddle point has at least four transitions in slope in its neighborhood pixel values. The pixel marked s in Fig. 2(c) is a 1 × 1 saddle point: each of its two humps consists of a direct neighbor with value 5.

Finally, we present the definition of a "climbing neighbor," used to define a steepest uphill path.

DEFINITION 3. The *climbing neighbor* for a nonzero pixel p that is not a local maximum is a maximum-valued neighbor, q, of p selected according to the following tie-breaking rule: Let R be a run of adjacent neighbors of p all with the maximum value such that the neighbor preceding and the neighbor succeeding the run are smaller-valued. (If there are several such runs, select one: the correctness proofs hold regardless of which run is selected.) If R contains an odd number of pixels, the climbing neighbor is the middle pixel of R. Else the climbing neighbor is the direct neighbor of p among the two pixels in the middle of R.

DEFINITION 4. The *hump climbing neighbor* for a hump is a maximum-valued neighbor among the pixels on the hump selected according to the above tie-breaking rule. The hump climbing neighbor for a hump of a 2 × 2 saddle is selected according to the same rules, except that in a 2 × 2, both pixels in the middle of an even run can be direct neighbors of saddle pixels. In this case, select either of the two direct neighbors.

If a nonzero pixel is not a local maximum, then it has a climbing neighbor (which is also a hump climbing neighbor for its hump). The distance transform value of the climbing neighbor for a pixel is greater than the value of the pixel itself. The distance transform value of a hump climbing neighbor for a saddle is greater than or equal to the value of the saddle itself. Figure 3 depicts the climbing neighbors for the object from Fig. 2(b), shown as a directed graph.

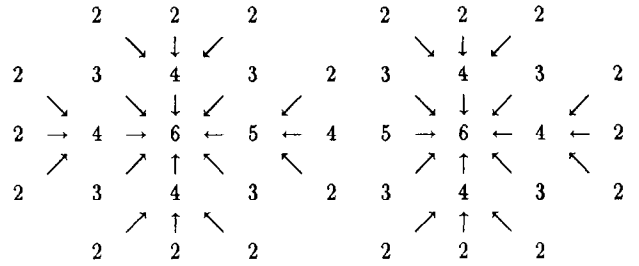

FIG. 3. Climbing neighbors for the object from Fig. 2(b), with respect to the 2–3 metric. An arrow points from each nonmaximum node to its climbing neighbor. The center pixel, with transform value 4, is a 1 × 1 saddle point with hump climbing neighbors, with value 5, to its left and right. Note that the three nonzero pixels in the center column each have two possible climbing neighbors, and we have arbitrarily selected for each the 5 on the left over the 5 on the right. The correctness proofs hold regardless of how such "ties" are broken.

Given these definitions, the following steps describe our basic algorithm:

BASIC ALGORITHM.

A1. Compute the distance transform using the desired metric.

A2. Select all pixels which are *local maxima* in the distance transform.

A3. For each local maximum, p, begin a steepest uphill climb from each direct neighbor of p of equal value that is not a local maximum. Each steepest uphill climb selects, one at a time, a sequence of pixels where the next pixel selected is the *climbing neighbor* of the previous pixel.

A4. Select all *saddle* pixels.

A5. For each saddle or 2 × 2 saddle, begin a steepest uphill climb from the *hump climbing neighbor* for each hump of the saddle. As in step A3, each steepest uphill climb selects, one at a time, a sequence of pixels where the next pixel selected is the *climbing neighbor* of the previous pixel.

3. THE CONNECTIVITY PROOFS

We now outline the proofs that the set of pixels selected by the algorithm, call them the set S, has the same simple connectivity as the set of objects in the image. The proofs show that S consists of a connected subset for each object in the image ("connectivity"), and that S loops around each hole in an object ("simple connectivity"). The full proofs, which are relatively long, can be found in a technical report [33]. The proofs hold for all natural distance metrics, i.e., all integer metrics such that $0 < d_D \leq d_I \leq 2d_D$.

The first result is stated as

THEOREM 1. *The basic algorithm selects an 8-connected subset of pixels for each 8-connected component of an image.*

The approach used in the proof is as follows, illustrated with the image in Fig. 2(b). We represent the image and specifically its distance transform values as a directed graph in which each nonzero pixel is a node. We define directed arcs from a pixel p to its 8-neighbors q as follows:

- If p is not a local maximum, there is an arc from p to its climbing neighbor, q. (Recall that any nonmaximum, even a saddle pixel, has a unique climbing neighbor.) This is shown in Fig. 3.
- If p is a (nonstrict) local maximum, there is an arc from p to all neighbors q with the same value as p that are either direct neighbors or local maxima.

Since both local maxima in our sample object are strict, no arcs of the second type are added, so the resulting graph for the object is as shown in Fig. 3.

Next we define "teams," essentially consisting of sets of pixels that can reach the same local maxima by following the directed arcs. Each pixel is on exactly one team. There are two teams in Fig. 3, those linked together on the left and those linked together on the right. Then, in the heart of the proof, we look along the boundary between two teams, specifically at a point of maximum value in the boundary. (The boundary between two teams consists of all pixels in either team with an 8-neighbor in the other team.) By a careful case analysis of the different configurations of possible distance transform values, and by using both 1×1 and 2×2 saddles, we are able to show that a saddle point must occur at (or at a neighbor of) this maximum point. Further, this saddle will have a hump climbing neighbor in each team. By selecting the saddles and climbing from their humps, as we do in steps A4 and A5 of the algorithm, we connect all teams within an object. The result is that the set S, from its construction, must have the same connectivity as the object.

The second major claim to be proved is that the set S surrounds each hole. We need several additional definitions:

DEFINITION 5. An *8-connected path* (*4-connected path*) of pixels in a grid is a sequence of distinct pixels, $p_1, p_2, \ldots, p_k$, $k \geq 1$, such that for all i, $1 \leq i < k$, pixels p_i and p_{i+1} are 8-neighbors (4-neighbors, respectively). An *8-connected cycle* is an 8-connected path such that the first and last pixels in the path are 8-neighbors, and the path does not contain all four pixels in a 2×2 neighborhood.

DEFINITION 6. The pixels in a cycle partition the remaining pixels in the grid into two sets: those in the interior of the cycle and those in the exterior. A cycle *encloses* a set of pixels, T, if all pixels in T are in the interior set of the cycle. A cycle *surrounds* a (4-connected) hole if the cycle encloses the hole but does not enclose any other zero-valued pixels.

DEFINITION 7. A *smallest-neighbor chain* for a pixel p is an 8-connected path to p from a zero-valued pixel such that each pixel in the path (other than p) is a smallest-valued neighbor of the next pixel in the path.

The claim is stated as:

THEOREM 2. *The basic algorithm selects a set of pixels, S, such that for each 4-connected hole, h, in the image there is an 8-connected cycle in S that surrounds h.*

The idea of the proof is as follows. If S does not surround each hole, there must be a hole for which there is a path out to background (or another hole) that does not cross S. Figure 4 gives an example, where u is a path from a hole to the background, not crossing S. We are able to show that if such a path exists, there must also exist a similar path v made up of two segments, each a

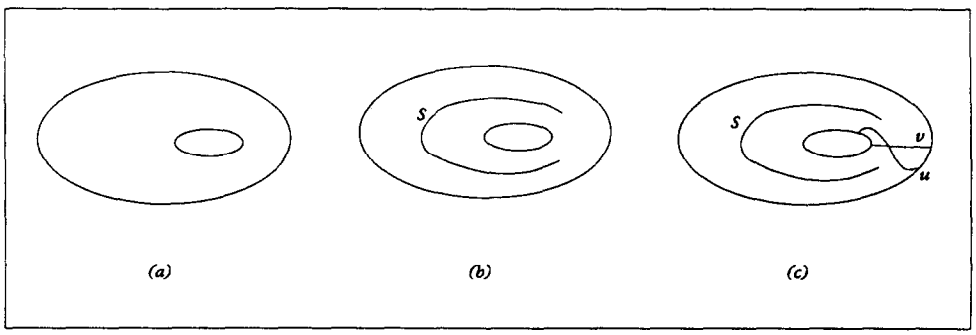

FIG. 4. (a) A sample object with a hole. (b) A skeleton S which does not surround the hole. (c) A path u from the hole to the background that does not cross S, and a shortest such path v, made up of two "smallest-neighbor chains."

smallest-neighbor chain, one ascending from the hole, the other ascending from the background. This is essentially a shortest path from the hole to the background not crossing S. See Fig. 4(c). Again, by a careful case analysis, we are able to show that, at its maximum point (where the two ascending chains meet), path v must necessarily cross points which are, by the nature of the distance transform and the definition of points in S, in the set S. That is, there is no path out that does not cross S, and thus all holes are completely surrounded by S.

Finally, we show that any loop in S does indeed surround a hole. This must be done carefully since special cases can arise.[5] We use the following definition:

DEFINITION 8. Let S be a set of pixels selected by the basic algorithm. Two indirect neighbors, u and v, in S are *linked* if v is a (hump) climbing neighbor for u, u is a (hump) climbing neighbor for v, or both u and v are local maxima. A *linked cycle* is an 8-connected cycle of pixels in S such that all pairs of pixels that are consecutive in the cyclic sequence for the cycle are either direct neighbors or linked indirect neighbors.

Then we are able to prove

THEOREM 3. *The basic algorithm selects a set of pixels, S, such that each linked cycle in S enclosing at least one pixel not in S encloses a hole in the image.*

The heart of the proof argues that if a pixel p is not in S, then there exists a smallest-neighbor chain to p that does not cross S. It follows that if a linked cycle in S encloses p, then the entire chain is enclosed within the cycle. In particular, the zero-valued pixel, z, at the head of the chain is enclosed within the cycle. Pixel z cannot be a background pixel, so it is part of some hole, h. By the previous theorem, the linked cycle encloses all of h, and thus any linked cycle in S encloses a hole in the image.

4. RECONSTRUCTABILITY

We now consider reconstructability. By this we mean the ability to reconstruct the object(s) using only the skeleton, and the values of the distance transform along the skeleton. This is a desirable property of the skeleton, and we show in this section how to extend the basic algorithm (in a way that preserves the connectivity) to achieve it. We begin with the following definitions:

DEFINITION 9. Let p be a pixel with distance transform value x. A *disk* or *ball* $B(p, x)$ of radius x associated with p is the set of (nonzero) pixels q such that the transform distance $d(p, q)$ from p to q is strictly less than x.

[5] These special cases are discussed in Section 9.

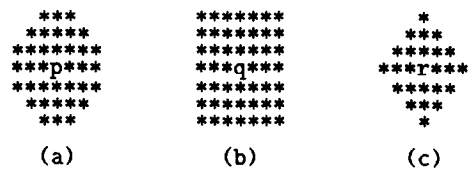

FIG. 5. Examples of disks under various distance transform metrics. (a) B(p, 8) for the 2–3 metric. (b) B(q, 4) for the $d8$ (1–1) metric. (c) $B(r, 4)$ for the $d4$ (1–2) metric. Since the 2–3 metric is better than the $d8$ or $d4$ metric at approximating the true Euclidean distance (e.g., its disk better approximates a round disk), skeletons based on the 2–3 metric are well-centered and more robust to rotation.

The disk for an object (nonzero) pixel is a *maximal disk* if it is not contained in the disk of any other pixel.

Examples of disks for various metrics are given in Fig. 5.

Let O be the object(s) in an image. Every pixel in the object is in at least one disk (its own), hence it is also in a maximal disk. Let $\{B_i\}$ be the set of maximal disks for the object. Then $O \subset \cup B_i$. Conversely, each maximal disk is completely within the object, so $\cup B_i \subset O$. Thus the union of the maximal disks is the object, or, stated another way, the object can be reconstructed from the set of maximal disks. This is a common result. See for example [27] or [34], who specifically treat the weighted distance transform. Next we prove the following lemma.

LEMMA 1. *If p with value x is a witness of q with value y, then $B(p, x) \subset B(q, y)$.*

Proof. Let r be a point in $B(p, x)$. Since p is a witness for q, $y = x + d(p, q)$. Then $d(r, q) \leq d(r, p) + d(p, q) < x + d(p, q) = y$, so $r \in B(q, y)$. ■

Thus the set of disks associated with witness pixels is always contained in other disks, so the set of nonwitness pixels suffices for exact reconstruction of the object. This led Arcelli and di Baja [34] to define a local maximum as a nonwitness pixel. (This is not the formal definition they give, but they point out that it is equivalent.) To avoid confusion, we keep the normal definition of local maximum (given in Section 2), but define a set of pixels, N, as follows:

DEFINITION 10. Let N be the set of pixels that are not a witness of any neighbor.

With this, we have that an object can be reconstructed from the disks associated with the set N. (This is Theorem 2 in [34].)

We now ask if this is a minimal set that allows reconstruction. The answer is no for several reasons. First, while a maximal disk is not contained in any other single disk, it may be contained in the union of several, as shown in Fig. 6(a) and (b). Second, another form of re-

FIG. 6. Examples showing that the set of maximal disks is not a minimal set for reconstruction. (a) An object. (b) The centers of maximal disks under many metrics (e.g., 2–3 or 3–4). The disk associated with the center pixel is not necessary for reconstruction. (c) A five pixel object. (d) Distance transform values for the 2–3 metric. All points are in N by Definition 10, but only the center pixel is needed for reconstruction.

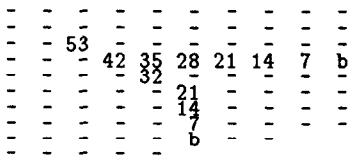

FIG. 8. (a) 7–11 distance transform. Points marked b are background, - are object. Only a portion of the object is indicated, and only selected values are shown. The point labeled 32 is not a witness, yet its disk is completely contained in the disk of the point labeled 53.

dundancy occurs because many pixels in N of value d_D (2, for our case of the 2–3 metric) are unnecessarily included. Figure 6(c) and (d) show a simple example. All object pixels are in N, but only the central pixel is necessary for reconstruction. Figure 7(a) shows a more complex case, the set N for the letter A object. The associated skeleton, part (b) of the figure, is "hairy" and thick. Third, in general, the pixels of N are not necessarily centers of maximal disks. This is not a well known result. Figure 8 shows an example using a "7–11" metric in which the disk of an N-pixel is completely contained in the disk of another N-pixel. As we show below, pixels in N are centers of maximal disks only if they are "sufficiently large."

Thus we are looking for a condition that allows us to identify a small (ideally minimal) set necessary for reconstruction. We know that, in general, the minimal subset for reconstruction is a strict subset of the centers of maximal disks, and these centers are a strict subset of N, the nonwitness pixels. So, starting from N, we want to exclude centers of nonmaximal disks. The next three theorems help us do this. The first says that to check whether the disk for a pixel is a (globally) maximal disk, it is sufficient to compare it solely with the disks associated with its 8-neighbors. (Thus a local test determines a global answer.) The second gives a sufficient condition for maximality within a pixel's 8-neighbors, and the third, essentially a result from number thoery, elaborates on the condition in the second theorem and gives an upper bound on the values for pixels that must be handled as special cases. The latter two theorems thus allow us to "weed out" the N pixels and still maintain reconstructability. The proof for Theorem 4 is a set of case analyses, and that for Theorem 6 is number theoretic. All proofs are given in [33], and, when applicable, hold for all natural metrics. In Theorem 6, gcd(m, n) means the greatest common divisor of integers m and n.

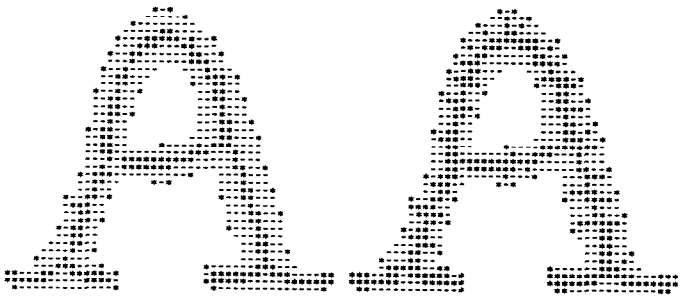

FIG. 7. (a) The N points. The object is taken from [4] © 1984 IEEE. The 2–3 metric is used. Many of these points are not needed for reconstruction, and hence our algorithm uses only a subset of these points. (b) A skeleton generated using the N points. Exact reconstruction is possible from this skeleton. The difference between this and the previous figure is that all N-pixels have been connected. This hairy skeleton contrasts with the skeleton generated by our algorithm (shown later in Fig. 13).

THEOREM 4. *Let p be a pixel with value $x > 0$. $B(p, x) \subset B(q, y)$ for some pixel q with value $y > 0$ if and only if $B(p, x) \subset B(r, z)$ for some pixel r, an 8-neighbor of p, with value $z > 0$.*

THEOREM 5. *Let d_D and d_I be positive integers. Consider a pixel p with value $x > 0$, and let m_1 and n_1 be nonnegative integers such that $x = m_1 d_D + n_1 d_I$. If p is not a witness, and if there exist nonnegative integers m_2 and n_2 such that $(m_1 - m_2)d_D + (n_1 - n_2)d_I = 1$, then $B(p, x) \not\subset B(q, y)$ for any q, an 8-neighbor of p with value y.*

THEOREM 6. *Let d_D and d_I be positive integers such that $gcd(d_D, d_I) = 1$ and $1 \le d_D \le d_I \le 2d_D$. Let $K = \{ad_D + bd_I: a, b \in Z, a, b \ge 0\}$, and let $x_0 = \min\{x \in K: \{x, x + 1, \ldots\} \subset K\}$. Then $x_0 = (d_D - 1)(d_I - 1)$.*

The idea behind Theorem 5 is that if x is a possible value for a pixel resulting from a "d_D–d_I" distance transform (that is why values m_1 and n_1 must exist), if p is not a witness, and if, when forming the disk centered on p of radius x, there are pixels around the border of the disk "just barely" in the disk (this is the condition $(m_1 - m_2)d_D + (n_1 - n_2)d_I = 1$), then these pixels will not be in the disk of any neighbor of p. This means no neighbor's disk will completely contain the disk at p and so the disk at p is maximal. Theorem 6 says that if x is "large" (that is, greater than $x_0 = (d_D - 1)(d_I - 1)$), this will always happen. This follows from the theorem since, if $x > x_0$,

then both x and $x - 1$ are possible values in a d_D–d_I distance transform, and thus there will be pixels "just barely" in the disk of radius x. Then the conditions of the theorem will hold, so $B(p, x)$ will be a maximal disk. On the other hand, if x is small (that is, smaller than $(d_D - 1)(d_I - 1)$), it may not be possible to find pixels "just barely" in the disk of radius x, so this case must be specially checked to see if p is a maximal disk.

Theorem 5 is a generalization of a result of Arcelli and di Baja for the 3–4 metric. (See Theorem 1 in [34]. Note that, as stated, their theorem does not extend to other metrics.) They show that if a pixel p with value $x > 6$ is not a witness, then its disk is maximal (for the 3–4 metric, $x_0 = (3 - 1)(4 - 1) = 6$). They also describe ways to test whether pixels with smaller values have maximal disks.

The condition given in Theorem 6 provides an upper bound on the values of an N-pixel for which we must explicitly check to see if it has a nonmaximal disk. For the 2–3 metric, this bound is 2, and for the 3–4 metric, it is 6.

We define a set N^* of points to be those with maximal disks, a set from which exact reconstruction of the image is possible.

DEFINITION 11. Let N^* be the set of all pixels that have maximal disks in the image.

For the 2–3 metric, an equivalent definition is as follows.

DEFINITION 12 (For the 2–3 metric). Let N^* be the set of pixels p such that (1) p is not a witness of any neighbor and (2) if x, the value of p, is equal to 2, p does not have a D-neighbor equal to 3 or an I-neighbor equal to 4.

For general (integer) metrics, a small table can be used to hold, for each sufficiently small transform value x (i.e., $x \in [d_D, (d_D - 1)(d_I - 1)]$), the minimum D-neighbor value, $D(x)$, such that the disk $B(p, x)$ is contained within the disk $B(q, y)$ of D-neighbor q with value y whenever $y \geq D(x)$. Similarly, for each small x the table holds a value $I(x)$ for I-neighbors. Then, when the set N^* is selected, pixels with value $x \in [d_D, (d_D - 1)(d_I - 1)]$ are explicitly checked to see if they have a neighbor whose ball contains their ball and, if so, they are not selected.

Summarizing the results of this section, we have that the objects can be reconstructed from the N^* pixels, and that

- For the particular case of the 2–3 metric, the N^* pixels (1) are not witnesses and (2) if equal to 2, do not have a D-neighbor equal to 3 or an I-neighbor equal to 4.
- For general metrics, the N^* pixels can be found as all nonwitness pixels with value greater than x_0, plus nonwitness pixels with values less than or equal to x_0 that are explicitly checked for maximality.

This set N^*, although not necessarily minimal, is a small set for reconstruction. The reconstructions itself is done using the reverse distance transform, which may be done in two raster passes through the image [27].

Remark. The results of this section are much simplified in the case $d_D = d_I = 1$ or $d_D = 1$, $d_I = 2$, i.e., the $d8$ and $d4$ metrics. In this case a pixel is the center of a maximal disk if and only if it is not a witness, and this is true if and only if it is a local maximum. Proofs for this are straightforward. See, for example, [5]. The disadvantage is that such metrics produce skeletons that are not robust with respect to rotation. See comments in [30].

4.1. *Discussion*

Using the N^* pixels defined above, the proofs of the connectivity properties referred to in Section 3 are still valid since they use the set of local maxima, a subset of N^*. The only change necessary to the basic algorithm of Section 2 is that we climb out of each N^* pixel if it is not a local maximum. If we do these climbs, which must eventually end on a local maximum, then this set is at least as connected as the results of step A3 of the basic algorithm, the continuation of the proof holds unmodified, and we know we have a connected result. Thus one can get exact reconstructability and still maintain connectivity.

However, this seemingly desirable goal having been reached, it turns out to have its disadvantages. First, the skeleton is no longer thin, but typically two pixels wide for even width objects or parts of objects. Second and more important, requiring exact reconstruction leads to skeletons that have many "hairs" and "runners." These occur not only on objects with noisy borders, in which case the number of runners can be very large, but also on objects with smooth, clean borders. This is true even when we use, for example, condition (2) in Definition 12, which excludes many runners. See, for example, Fig. 9. Similar examples are given by Pavilis [3] (he uses d_4) and Xia [23].

Thus, for pseudo-Euclidean metrics, one must accept either skeletons that have both hairs and thick sections, or approximate reconstruction. Essentially all authors choose the latter. SPTA [8], which basically uses d_4, allows reconstruction "similar to the original," and the authors describe the reconstruction as one such that "when the reconstructed pattern was superimposed on the original pattern, the SPTA on average gave a point-to-point match of 94%".[6] The MAT-based algorithm of

[6] SPTA also assumes a preprocessing step to smooth the input objects.

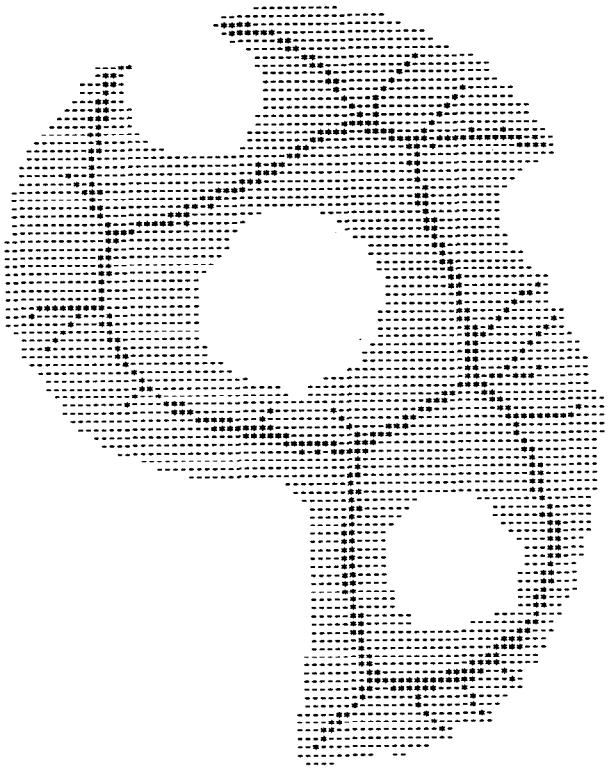

FIG. 9. Skeleton for an object with a smooth border. The skeleton has many hairs and runners out to points in N^*, points with maximal disks, in order to allow exact reconstruction. The object is taken from [6] © 1985 IEEE.

Arcelli and di Baja [6] produces a skeleton from which "the original figure can almost completely be recovered." Pavlidis [3], whose method can produce exact reconstruction, includes a switch in his algorithm to control whether the skeletons will allow reconstruction or be "nice." We do the same and allow the program to be run selecting either skeletons for exact reconstruction (as discussed above), or nicer skeletons that permit approximate reconstruction (as discussed in the next section).[7] As a third alternative, we can also generate "centerlines," based on the local maxima and not the complete set of N^* pixels, for potentially even nicer skeleton-like subsets of objects (but with no guarantees on reconstruction). Centerlines will be discussed in Section 7.

[7] Xia [23] claims exact reconstructability from his skeletons, but the example given to show this, Fig. 15(a) in his paper, appears to us to show that the image is *not* reconstructable. Not all pixels, for example, along the upper left edge of the letter "a" will be reconstructed, and this is true even though the skeleton is two pixels wide. Also, in order to produce skeletons which allow this reconstruction, an extra scan (per iteration?) is necessary to check for exceptions that prevent reconstructability.

4.2. *Approximate Reconstruction*

We now show how to generate skeletons that allow reconstruction to within a specified maximum error along the border. This is particularly useful given the "border noise" problem that may affect skeletonization algorithms (e.g., [1, 2, 6, 23]) in which small, often single pixel, irregularities along an object border generate runners and hairs to the border. Using approximate reconstruction, these are naturally omitted from the skeleton and no image prefiltering or skeleton postpruning are necessary. The processing requires an additional pass, but no heuristic rules for filtering or pruning are necessary.

The processing for approximate reconstruction is done after the N^* points are found. For each $p \in N^*$, we perform a steepest uphill climb, noting the accumulated distance D we move. We also note the difference Δ between y, the value of the pixel q to which we have climbed, and x, the value of p, the starting pixel. (For example, if the path were a witness chain, D would always equal Δ, but since $p \in N^*$, p will not be a witness of the first pixel in the climb, and hence $\Delta < D$.) At q, $D - \Delta$ is the error in reconstruction resulting from omitting p and using q instead. If q is a skeleton point, and $D - \Delta \leq d_D$, we can delete p and still reconstruct to within a one pixel error, and, in general, if $d - D \leq \varepsilon$ at q, we can reconstruct to within ε distance units even if we omit p. (Reconstruction to within ε distance units means that for any pixel p not reconstructed, there is a pixel q that is reconstructed such that $d(p, q) \leq \varepsilon$.) In practice, this turns out to be a conservative error bound. Examples will be shown in Section 8.

We must be careful not to iteratively delete skeleton points, i.e., delete p because the disk at q is sufficient, and subsequently delete q because the disk at r is sufficient. We keep track of "necessary" points to avoid this. The processing to trim the N^* points to allow for approximate reconstruction to within ε distance units becomes:

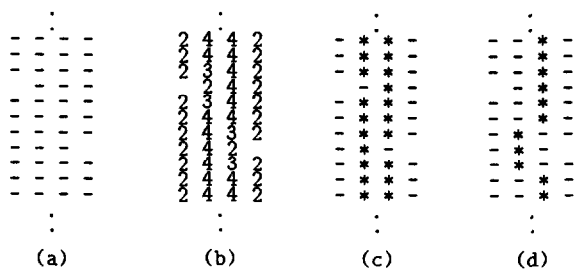

FIG. 10. Example of thick results from an even width object. (a) Object. (b) Distance transform using the 2–3 metric. (c) N^* points, a thick set. (d) N^* points remaining after an explicit thinning pass.

```
if p ∈ N*, p not required for connectivity of N* points, and p not marked "necessary"
    do until end of climb
        look for the steepest uphill step
        if a step is possible (i.e. we are not at a strict local maximum)
            let q with value y be the maximum neighbor
            D = distance stepped from p to q
            Δ = y − x (where x is the value of p)
            if D − Δ ≤ ε
                step to q
            else
                exit climbing
            endif
        else
            exit climbing
        endif
    enddo
    if we hit an N* point during the climb (let r be the last such point)
        remove p from skeleton
        put r in the skeleton (in case it was removed)
        mark r as "necessary"
    else
        mark p as "necessary"
    endif
endif
```

The remaining set of N^* pixels allows approximate reconstruction. Other skeleton points, such as saddles and climbing points, are not needed. However, if only the trimmed N^* points are used, the reconstructed object may not have the same connectivity as the original—it can break in regions near saddles. To guarantee connectivity of the reconstructed object, the saddle and climbing pixels must be added to the reconstruction result obtained using the (trimmed) N^* points. Alternatively we can reconstruct using the disks of all skeletal points and obtain even smaller reconstruction errors.

Approximate reconstruction as we have defined it is not the same as what may be called "partial reconstruction," i.e., reconstruction of a slightly thinner version of the object. For example, a partial reconstruction to within two pixels would be an object with the outer two layers of pixels removed. Consider an object that is a perfect disk (in the sense of disks defined earlier). The skeleton consists of the single point at the center. The processing for approximate reconstruction will not remove any skeletal pixels—there are no small hairs in the skeleton that protrude to account for small irregularities along the border, and the approximate reconstruction will be identical to the exact reconstruction. In contrast, a partial reconstruction could be done by reducing the value at the center pixel (and in general at each skeletal point) some t distance units, giving a reconstruction with t pixels removed all along the circumference.

5. THINNESS

According to the proofs in previous sections, selecting the four types of pixels (N^*, climb-from-N^*, saddles, and climb-from-saddles) results in a connected subset of an object from which the object can be reconstructed. In effect, we are saying that while the N^* points (or local maxima) of an object are disconnected, we can reconnect them by a set of hill climbing operations, first from the points themselves, and then from the saddle points, and further, if we have used the set N^* instead of the local maxima, we can guarantee reconstructability.

We have not shown that this subset is thin. In fact, it is not. One cause of the thickness is illustrated in Fig. 10, where for a simple even width object, the distance transform values contain two adjacent (and perhaps nonequal) N^* points.

We now describe modifications to the algorithm to produce thinness. These modifications conflict with other properties of the algorithm, in particular the reconstructability. In our implementation, the thinness modifications are selectable–they may be enabled or disabled and we will show examples with and without the modifications.

The modifications are embedded in the algorithm, and require an additional pass. An alternative is to simply thin the skeleton as a postprocessing step as in [6]. The advantage of the method we describe is that, where there

is a thickness, our method selects the larger of the two points if they are unequal (recall that a pixel can have a larger neighbor and still be an N^* point), so that the error in reconstruction is minimized; thinning as a postprocessing step (typically by peeling from, say, the top, right, bottom, and left) may result in slightly larger reconstruction errors. Still, thinning using a postprocessing peeling step is quite acceptable for many applications.

To produce a thin skeleton, we first note that by selecting skeletons for approximate reconstruction with one or two pixels error along the border, the (reduced) set of N^* pixels is almost always thin. If the climbing pixels and saddle pixels we add do not introduce thickness, the final skeleton will be almost always thin as well. However, we noted two problems when doing this. First, the processing of the N^* points for approximate reconstruction does not necessarily produce a thin set. There is no guarantee that by trimming out certain N^* points to obtain reconstruction to within, say 2, pixels, the resulting skeleton is thin, and counterexamples can be constructed that have thick places after the trimming. Second, the processing for approximate reconstruction can produce zig-zag, "zipper-like" patterns. Both of these situations tend to occur at bends or corners in the skeleton, or where several branches come together. To minimize these occurrences, we directly thin the set of N^* points (before the processing for approximate reconstruction) as follows:

```
if p ∈ N*
    if left neighbor of p is in N*
        if right neighbor of p is a witness for p (‡)
            let s be the smaller of p and its left neighbor (s = p in case of a tie (†))
            if s was not the "winner" in a previous comparison
                delete s
            endif
        endif
    endif
    if right neighbor of p is in N*
        if left neighbor of p is a witness for p (‡)
            let s be the smaller of p and its right neighbor (s = p in case of a tie (†))
            if s was not the "winner" in a previous comparison
                delete s
            endif
        endif
    endif
endif
```

A similar check is applied for top and bottom neighbors during the same pass. Figure 10(d) shows the result of applying this thinning procedure to the N^* points in Fig. 10(c).

Notes. (1) The two lines marked (†) are not symmetrical. If there is a tie, in one case the left point is deleted, and in the other, the right point is deleted. This reduces the jaggedness of the skeleton at certain corner configurations.

(2) The lines marked (‡) find N^* points where there is a thickness of the form

w p q

in which p and q are two adjacent N^* pixels and w is a witness of p that is opposite q. (For example, this is the case in many of the horizontal slices of the object in Fig. 10(b).) A thickness can also occur where the witness of p is not directly opposite q, but is any 8-neighbor of p that is not an 8-neighbor of q, such as u or v in

u
w p q
v

We tried this condition in the thinness check. It eliminates some rarely occurring thick places. However, it also introduces occasional one-pixel jags in the skeleton, and we felt that overall the simpler check is preferred.

As mentioned above, we must keep from adding "thickening" pixels to the skeleton during the climbing and finding-saddles steps. To do this, we include a "redundancy check" and do not add a pixel if it is redundant. A redundant pixel is one whose addition connects only pixels which are already (8-)connected. Examples are shown in Fig. 11. In part (a), the center pixel p is

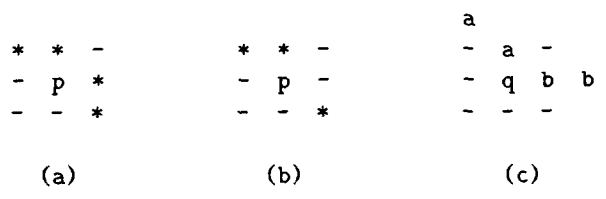

FIG. 11. (a) A redundant pixel, (b) a valid pixel, and (c) a redundant saddle.

redundant, whereas in part (b) it is not. One special case that arises is when a saddle point itself is redundant after its climb-from-saddle pixels have been added to the skeleton. An example is shown in Fig. 11(c). Pixel q is the saddle, the "a"s are one path out, and the "b"s are the second path out. To avoid these redundant pixels, we add the saddle point (subject to the redundancy check) after the climbs, and not before.

A difficult example for generating a thin skeleton is given in Fig. 12, taken from [6]. By using the N^* pixels, as opposed to the N pixels, and applying the above thinness check for redundant saddle pixels, we achieve the thin skeleton shown in the figure. In this particular example, no N^* pixels were removed during the thinness check, so the skeleton both is thin and allows exact reconstruction.

The combination of N^*-thinning and redundancy check removes almost all cases of thickness in the skeleton, but thick places are still possible. One example is identical to the case shown in Fig. 11(c), but in the case where q is an N^*-pixel rather than a saddle, and the a's and b's are two separate climbs to q. Pixel q is no longer necessary for connectivity, and it may be that, with the a's and b's, it is no longer necessary for reconstruction, so the skeleton is not perfectly thin at q. It is for cases such as this that we mentioned in the introduction that we could not guarantee perfect thinness. An additional pass could be used to remove these but the thinness we achieve is sufficient for many applications.

6. IMPLEMENTATION OF THE ALGORITHM

Including the extensions described in the previous sections, our algorithm for generating skeletons is:

A1. Compute the distance transform using the desired metric. A new image is created in which the value of each pixel represents (an approximation to) its shortest distance to the background using the selected metric (2–3, 3–4, etc.). This may be done by propagating local distances between, for our case, 8-connected pixels, and requires two passes through the image. In the first pass, the (approximate) distance of each pixel from the background, measured from the top and left, is computed. In the second, the distance from the bottom and right is similarly computed and, within this pass, the minimum of these two distances is assigned to the pixel. A method that is typically much faster is to compute the distance transform using a peeling approach, where the distance value for each pixel is inserted as the object boundary is peeled in. This method avoids both arithmetic and comparisons at each pixel. It is described for weighted distance transforms in [35].

A2. Select all N^* pixels. If a thin skeleton is requested, thin the N^* points using the procedure outlined in Section 5. If approximate reconstruction is requested, delete all N^* points not needed for reconstruction to within the desired tolerance, using the procedure outlined in Section 4.2.

A3. For each remaining p in N^*, begin a steepest uphill climb from (a) its climbing neighbor if p is not a local maximum, or (b) each direct neighbor of p of equal value that is not in N^*, if it is. (If thin skeletons are being generated, we do not begin a climb on a neighbor that was removed in the thinning.) Each steepest uphill climb selects, one at a time, a sequence of pixels where the next pixel selected is the *climbing neighbor* of the previous pixel. In cases where there are several possible choices for the climbing neighbor of a pixel, e.g., it has several runs of maximum-valued neighbors, select the climbing neighbor according to the following direction priority: W, E, N, S, NW, NE, SW, SE. A selected pixel is discarded if it is redundant, i.e., if its addition connects only pixels which are already 8-connected. The climb ends at the first pixel whose climbing neighbor has already been selected for the skeleton (i.e., the climbing neighbor is a local maximum).

A4. Locate all *saddle* points. This is done by scanning the 3×3, and sometimes 4×4, neighborhood of each pixel.

A5. For each saddle, begin a steepest uphill climb from the *hump climbing neighbor* for each hump of the saddle. As in step A3, each steepest uphill climb selects, one at a time, a sequence of pixels where the next pixel selected is the *climbing neighbor* of the previous pixel. A selected pixel is discarded if it is redundant. The climb

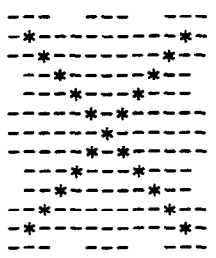

FIG. 12. A thin skeleton that allows exact reconstruction. The object is taken from [6] © 1985 IEEE.

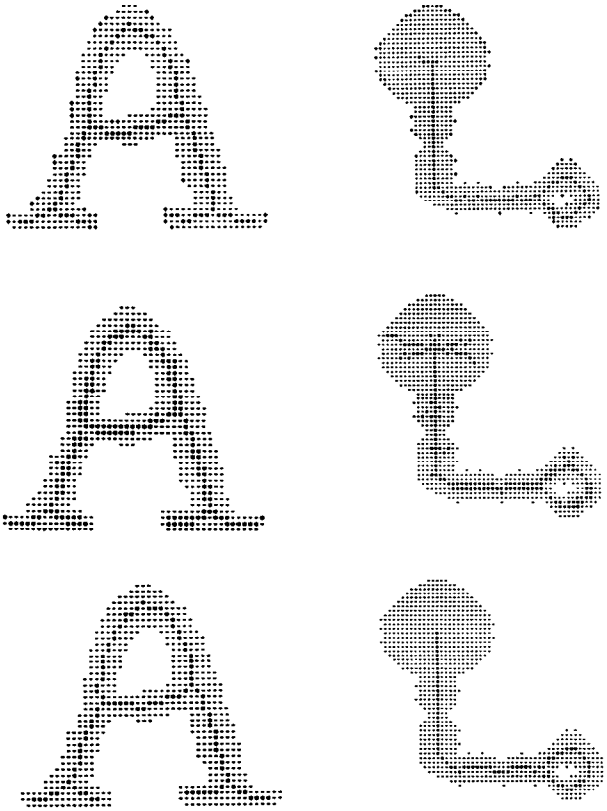

FIG. 13. Examples of letter A (from [4] © 1984 IEEE) and printed circuit board. From top to bottom: thin skeleton allowing the object to be reconstructed to within one 8-neighbor (from approximate reconstruction) plus one 4-neighbor (from the thinness check) along the object border; skeleton for exact reconstruction; centerlines. Errors in reconstruction are shown as +. Note the bottom right portion of the left leg of the A. The allowable reconstruction error is attained in this case.

ends at the first pixel whose climbing neighbor has already been selected for the skeleton.[8] Select the saddle pixel itself unless it is redundant, performing the check after the climbs out. For a 2 × 2 saddle, select the two pixels that are 4-neighbors of the selected hump climbing pixels, subject to the redundancy check.

[8] There is a subtle correctness issue here and in step A3 in not continuing a climb until a local maximum is encountered (as prescribed in Section 2). We can safely end a climb whenever its next step is to a previously selected pixel, p, since any pixels in a climb *through* p would be selected in a climb *starting at* p. This is true since a climb through p, if any, would next step to the climbing neighbor for p, while a climb starting at p would also step to the climbing neighbor (and possibly other hump climbing neighbors, if p is a 1 × 1 saddle). The only subtle case is when p is a member of a 2 × 2 saddle point, since the climbing neighbor for p may not be a hump climbing neighbor for the 2 × 2 saddle. However, it can be shown that it is not possible for a legal climb to climb to a member of a 2 × 2 saddle, so this case cannot occur. We omit the details.

7. CENTERLINES

As an alternative to the skeletons for either exact or approximate reconstruction, we can also generate "centerlines." These are thin subsets (in the same sense as above) of the objects with the same simple connectivity as the objects. They are generated by using the basic algorithm for achieving connectivity (Section 2) with the same techniques for thinning (Section 5), but without the extensions for reconstructability. Thus they use the set of local maxima, not the set N^*. The result is that the generated set has fewer hairs and runners. Centerlines do not allow proper reconstruction of the object, nor are they robust with respect to rotation. However, they provide a small[9] connected, topology-preserving subset of an object that contains all the local maxima, and are naturally immune to border noise. We have found them useful in applications where connectivity is the main issue.

As shown in the examples later, centerlines are often similar to the thin skeleton with one or two pixel errors, and can be generated using a somewhat simpler algorithm.

8. EXAMPLES

Two sets of examples are shown in Fig. 13. The first is of a figure of the letter A that has appeared in numerous skeletonization papers (for example [4]), and the second is one we generated representative of printed circuit board patterns with noisy borders. For both examples, three results are shown: the thin skeleton (allowing approximate reconstruction), the skeleton for exact reconstruction, and the centerlines. The thin skeletons were generated to give a maximum of one 8-neighbor pixel error in reconstruction (i.e., 3 distance units in the 2–3 metric). The thinnes check was enabled, allowing another 4-neighbor pixel error. The reconstruction errors are indicated in the figures as + pixels. These are pixels in the original not in the reconstruction. The opposite case, pixels in the reconstruction not in the original, does not occur in our algorithm.

A more complex example is shown in Figs. 14, 15, and 16. The figure is a test image we generated (we included the original to show the single pixel wide parts of the object). In this set, we also show a skeleton generated to allow larger reconstruction errors. Note: In order to show that the exact reconstruction error bounds are achieved, we performed the reconstruction from the (thinned) N^* points only, and then added the other skele-

[9] Small but not "minimal." The reason is that, in pathological cases, saddle points can be generated that do not connect distinct local maxima, so the connecting paths are not strictly necessary. An example is given in [33].

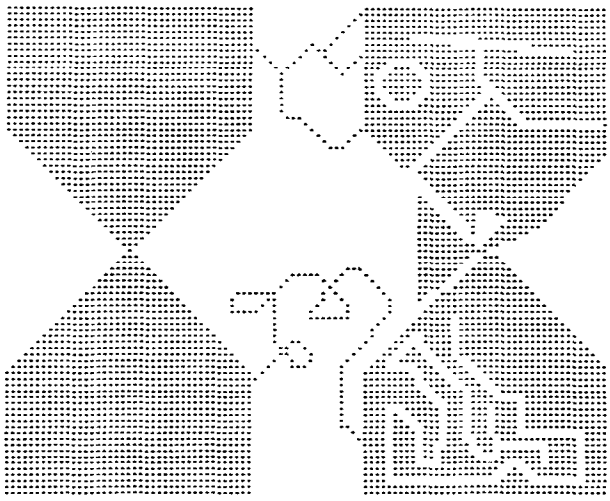

FIG. 14. A test image.

tal points. As mentioned earlier, smaller errors can be obtained if the reconstruction is done by applying the reverse distance transform to *all* skeletal points, including saddles and climbing pixels.

The 2–3 metric was used in these examples.

Another example of the skeleton for exact reconstruction is Fig. 9, shown earlier.

9. DISCUSSION

Robustness to Rotation. Figure 17 shows the skeletons generated for a square rotated 0°, 30°, and 45°. In general, because we use a pseudo-Euclidean distance transform, the skeletons are very robust with respect to rotation. Figure 18 shows an additional example under rotation, the object of Fig. 1.

Holes. Theorem 3 states that any loop in the set S (the set that, with thinning, we now refer to as centerlines) must surround a hole in the object, as long as the loop is a "linked cycle" (see the definition in Section 3). Figure 19 shows an example of apparent "holes" generated by the algorithm when centerlines are specified, resulting from adjacent S pixels that are not linked. This happens when indirect neighbor S pixels are "tangent" on separate arms of S. In fact, it is situations such as this that led us, in deriving the proofs of connectivity, to define linked pixels. We have not observed this type of hole with skeletons.

False single-pixel holes can occur in both skeletons and centerlines if the thinness criterion is applied in the same pass as finding saddles and climbing from saddles. (This is the way we normally run the algorithm.) An example is shown in Fig. 20. Such holes could be removed by an additional pass.

Computational Effort. After the distance transform, the algorithm uses two passes through the image: one to find and climb from N^* points, and one to find and climb from saddles. The thinness check and processing for approximate reconstruction each require an additional pass, so that four passes are needed for all options. The passes for saddles and thinning can be combined, but if so, results differ slightly. The passes involving climbing require nonraster random access to the image pixels to do

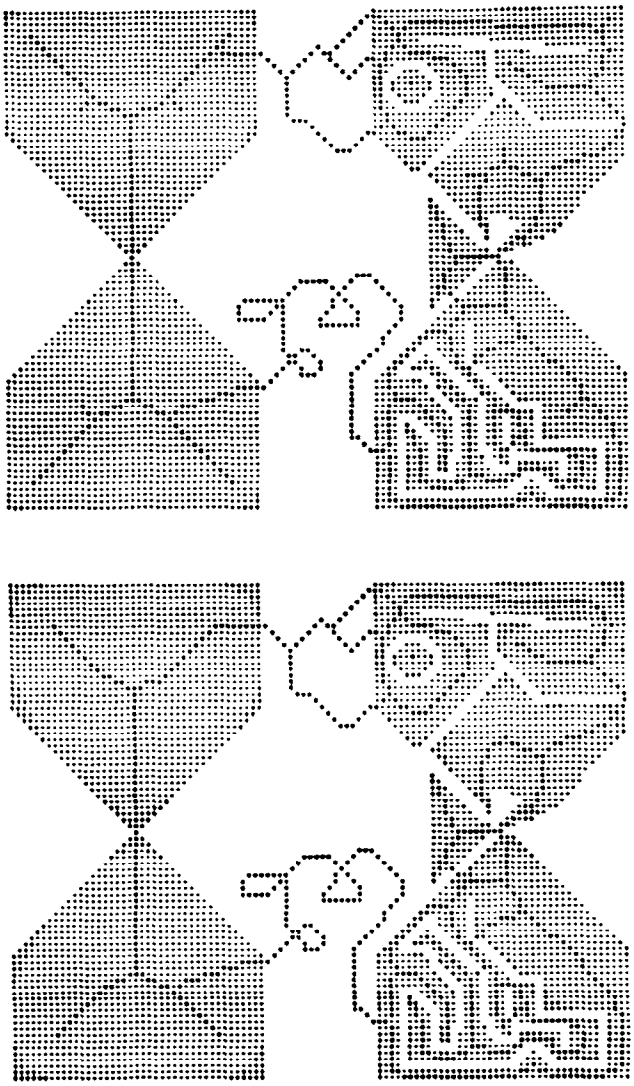

FIG. 15. Top: Skeleton generated to allow reconstruction to within 5 distance units in the 2–3 metric, 3 units (i.e., one 8-neighbor) from approximate reconstruction, plus 2 units (one 4-neighbor) from the thinness check. Bottom: Skeleton generated to allow reconstruction to within 8 distance units, 6 units (e.g., two 8 neighbors) from approximate reconstruction, and 2 units (one 4-neighbor) from the thinness check. Errors in reconstruction are shown as +. Near the upper right corner (fifth object line down, right edge), and at the centers of the Y's in the left half of the object, are examples of extra "thick" pixels not needed for connectivity.

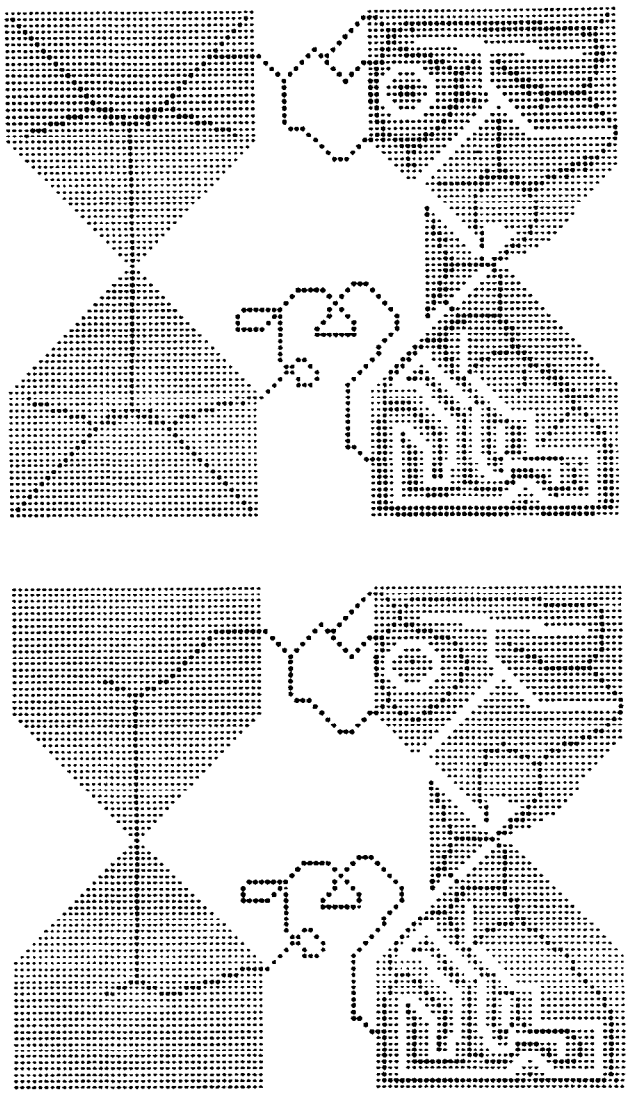

FIG. 16. Top: Skeleton generated to allow exact reconstruction. Bottom: Centerlines.

the climbing. This number of passes is independent of the number of objects or their widths. On a grid of n pixels, the algorithm runs in $O(n)$ time.

The pass to do the processing for approximate reconstruction needs explanation. The procedure (recall Section 4.2) involves climbing from each N^* pixel. As it was described, these climbs are different from the climbs out

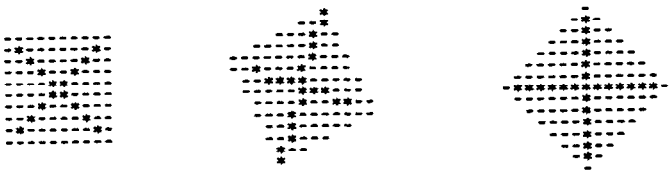

FIG. 17. Skeletons of a square rotated 0°, 30°, and 45°.

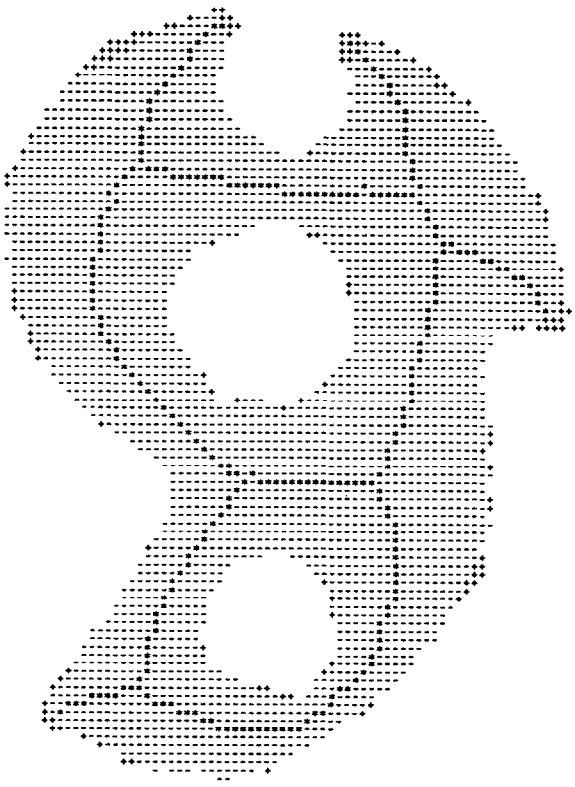

FIG. 18. Skeleton after rotation. Original object is taken from [6] © 1985 IEEE. The 2–3 metric was used, and the skeleton allows the object to be reconstructed to within 8 distance units along the object border, 6 distance units or two 8-neighbors from approximate reconstruction, and 2 distance units or one 4-neighbor from the thinness check. As in earlier figures, errors in reconstruction are shown as +.

of N^* pixels and saddles in that they do not stop when they encounter a pixel processed in a previous climb. Thus it is possible for the set of climbs to visit a pixel many times, and it is not clear that this is an $O(n)$ algorithm. However, by suitable bookkeeping (keeping track of the value of Δ for each point during a climb, and at the end of the climb adding a pointer to each pixel giving its closest "uphill" neighbor with the same value of Δ), this can be avoided and each pixel visited a fixed number of

FIG. 19. An example of apparent "holes" in centerlines when the thinness check is not applied. The pair of pixels labeled a and the pair of pixels labeled b each complete a cycle enclosing a single pixel that appears to be a hole. This type of hole is due to the limitations of the grid, and neither pair are "linked" pixels. The problem goes away as a side effect of the thinning.

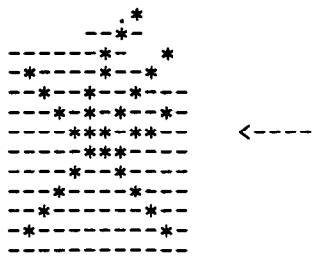

FIG. 20. A case where the thinness check generates holes in the skeleton. (The fifth pixel from the right edge in the line marked with the arrow.)

times. Thus the algorithm can indeed run in $O(n)$ time. In our current implementation, we use the straightforward method as described in Section 4.2.

We have performed limited experiments comparing the execution time of the algorithm with that of other techniques. For a 528 × 512 test image of a printed circuit board from an industrial application, using a contour-based distance transform, our algorithm required 2.5 s. A highly-optimized raster-based four subpass iterative parallel algorithm (as in [10–12]) required 85 s and a sequential implementation of Zhang's two subpass iterative parallel algorithm [13] required 39 s. (Note that these algorithms naturally map better to parallel hardware.) All were written in C and run on an IBM RS/6000, approximately 20 mips. The image contained several thick (approximately 90 pixel diameter) circuit pads.

10. CONCLUSIONS

We have described an efficient algorithm that requires a fixed number of passes though an image to produce a well-centered skeleton with the same simple connectivity as the object, that allows the object to be either exactly or approximately (to within a known error) reconstructed, and that has good robustness with respect to rotation. We were able to prove both its connectivity-preserving and its reconstructability properties. If approximate reconstruction is accepted, the skeleton can also be (almost always) thin. In this case, it is also insensitive to border noise without image prefiltering or skeleton postpruning. By a simple modification, it can also produce centerlines, skeleton-like topology-preserving subsets of the objects even less sensitive to border noise. We have tested it on a variety of real, contrived, and pathological images, and found it to give excellent and visually appealing results. Because of these properties, its robustness to rotation, pleasing visual appearance, and flexiblity, it is well suited for such applications as data compression, image analysis, character recognition, and circuit board inspection.

ACKNOWLEDGMENTS

We acknowledge the help of Jim Hafner of the Discrete Mathematics Group, Almaden Research Center, who provided Theorem 6, as well as helpful discussions related to the reconstruction methods presented in this paper.

REFERENCES

1. B.-K. Jang and R. T. Chin, Analysis of thinning algorithms using mathematical morphology, *IEEE Trans. Pattern Anal. Mach. Intelligence* **PAMI-12,** 1990, 541–551.
2. E. R. Davies and A. P. N. Plummer, Thinning algorithms: A critique and a new methodology, *Pattern Recognition* **14,** 1981, 53–63.
3. T. Pavlidis, An asynchronous thinning algorithm, *Comput. Graphics Image Process.* **20,** 1982, 133–157.
4. G. Bertrand, Skeletons in derived grids, in *7th ICPR, Montreal, July–August 1984,* pp. 326–329.
5. WX. Gong and G. Bertrand, A fast skeletonizing algorithm using derived grids, in *9th ICPR, Rome, November 1988,* pp. 776–778.
6. C. Arcelli and G. Sanniti di Baja, A width independent fast thinning algorithm, *IEEE Trans. Pattern Anal. Mach. Intelligence* **PAMI-7,** 1985, 463–474.
7. C. J. Hilditch, Linear skeletons from square cupboards, in *Machine Intelligence IV,* pp. 403–420, Edinburgh Univ. Press, Edinburgh, 1969.
8. N. J. Naccache and R. Shinghal, SPTA: A proposed algorithm for thinning binary patterns, *IEEE Trans. Systems Man Cybernet.* **SMC-14,** 1984, 409–418.
9. H. Tamura, A comparison of line thinning algorithms from digital geometry viewpoint, in *4th ICPR, Kyoto, November 1978,* pp. 715–719.
10. R. Stefanelli and A. Rosenfeld, Some parallel thinning operations for digital pictures, *J. Assoc. Comput. Mach.* **18,** 1971, 255–264.
11. A. Rosenfeld and L. S. Davis, A note on thinning, *IEEE Trans. Systems Man Cybernet.* **SMC-6,** 1976, 226–228.
12. J. R. Mandeville, Novel method for analysis of printed circuit images, *IBM J. Res. Devel.* **29,** 1985, 73–86.
13. T. Y. Zhang and C. Y. Suen, A fast parallel algorithm for thinning digital patterns, *Commun. Assoc. Comput. Mach.* **27,** 1984, 236–239.
14. C. M. Holt, A. Stewart, M. Clint, and R. H. Perrott, An improved parallel thinning algorithm, *Commun. Assoc. Comput. Mach.* **30,** 1987, 156–160.
15. S. Suzuki and K. Abe. Binary picture thinning by an iterative parallel two-subcycle operation, *Pattern Recognition* **20,** 1987, 297–307.
16. R. W. Hall, Fast parallel thinning algorithms: Parallel speed and connectivity preservation, *Commun. Assoc. Comput. Mach.* **32,** 1989, 124–131.
17. Z. Guo and R. W. Hall, Parallel thinning with two subiteration algorithms, *Commun. Assoc. Comput. Mach.* **32,** 1989, 359–373.
18. R. T. Chin, H.-K. Wan, D. L. Stover, and R. D. Iverson, A one pass thinning algorithm and its parallel implementation, *Comput. Vision Graphics Image Process.* **40,** 1987, 30–40.
19. T. Pavlidis, A thinning algorithm for discrete binary images, *Comput. Graphics Image Process.* **13,** 1980, 142–157.
20. W. Xu and C. Wang, CGT: A fast thinning algorithm, *IEEE Trans. Systems Man Cybernet.* **SMC-17,** 1987, 847–851.
21. P. C. K. Kwok, A thinning algorithm by contour generation, *Commun. Assoc. Comput. Mach.* **31,** 1988, 1314–1324.

22. L. J. van Vliet and B. J. H. Verwer, A contour processing method for fast binary neighborhood operations, *Pattern Recognition Lett.* **7,** 1988, 27–36.

23. Y. Xia, Minimizing the computational complexity of iterative sequential thinning algorithms, in *9th ICPR, Rome, November 1988,* pp. 721–723.

24. S. Suzuki and K. Abe, Sequential thinning of binary pictures using distance transformations, in *8th ICPR, Paris, October 1986,* pp. 289–292.

25. J. Toriwaki and S. Yokoi, Distance transformations and skeletons of digitized pictures with applications, in *Progress in Pattern Recognition* (L. N. Kanal and A. Rosenfeld, Eds.), Vol. 1, pp. 187–264, North-Holland, New York, 1981.

26. P. A. Maragos and R. W. Schafer, Morphological skeleton representation and coding of binary images, *IEEE Acoust. Speech Signal Process.* **ASSP-34,** 1986, 1228–1244.

27. A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Academic Press, New York, 1982.

28. G. Borgefors, Distance transformations on digital images, *Comput. Vision Graphics Image Process.* **34,** 1986, 344–371.

29. U. Montanari, A method for obtaining skeletons using a quasi-euclidean distance, *J. Assoc. Comput. Mach.* **15,** 1968, 600–624.

30. P.-E. Danielson, Euclidean distance mapping, *Comput. Vision Graphics Image Process.* **14,** 1980, 227–248.

31. L. Dorst, Pseudo-euclidean skeletons, in *8th ICPR, Paris, October 1986,* pp. 286–289.

32. W. Niblack, D. Capson, and P. Gibbons, Generating skeletons and centerlines from the medial axis transform, in *10th ICPR, Atlantic City, New Jersey, June 1990,* pp. 881–885.

33. W. Niblack, P. Gibbons, and D. Capson, *Generating Skeletons and Centerlines from the Medial Axis Transform*, Research Report RJ 8589, IBM Almaden Research Center, 650 Harry Road, San Jose, California 95120, January 1992.

34. C. Arcelli and G. Sanniti di Baja, Finding local maxima in a pseudo-euclidean distance transform, *Comput. Vision Graphics Image Process.* **43,** 1988, 361–367.

35. B. J. H. Verwer, Improved metrics in image processing applied to the Hilditch skeleton, in *9th ICPR, Rome, November 1988,* pp. 137–142.