Pattern Recognition Letters 14 (1993) 173–179 March 1993

A single scan boundary removal thinning algorithm for 2-D binary object

P.K. Saha, B. Chanda and D. Dutta Majumder

Electronics and Communication Sciences Unit, Indian Statistical Institute, 203 Barrackpore Trunk Road, Calcutta 700 035, India

Received 3 September 1991
Revised 10 September 1992

Abstract

Saha, P.K., B. Chanda and D. Dutta Majumder, A single scan boundary removal thinning algorithm for 2-D binary object, Pattern Recognition Letters 14 (1993) 173–179.

In this paper a new thinning algorithm called Single scan Boundary Removal Thinning Algorithm (SBRTA) is proposed. This algorithm uses a single scan to remove border points from all sides, i.e., left, right, top and bottom. It is found that SBRTA preserves topology and produces rotation-invariant good skeletons. The performance of the proposed algorithm has been compared with those of existing algorithms. The algorithm is computationally efficient.

Keywords. 2-D binary image, single scan 2-D thinning.

1. Introduction

In the context of 2-D binary image processing, thinning is considered by many researchers as an iterative procedure which removes black points (object points) across the boundary until the image is thinned to a one-point thick black arc, called skeleton. A good thinning algorithm must preserve the topology as well as the shape of the original image in the skeleton. Thinning is one of the most popular preprocessing methods in many problems of pattern recognition and analysis like character recognition, finger-print classification etc. This is due to the fact that (i) it preserves essential structural information of an image, (ii) it reduces the space to store topological as well as shape information of an image, (iii) it reduces the complexity of analyzing the image. Two other image transformations which are very near to thinning are medial axis transformation and shrinking. These techniques sometimes fail to preserve topology or shape information. For example, in many cases discrete medial axis transformations [4] do not preserve topology [7], and on the other hand shrinking reduces a digital open arc to one point and looses shape information.

During the past few decades, several thinning algorithms have been proposed [1, 3, 5, 6, 8–12]. Naccache and Singhal [8] made a study of 14 thinning algorithms based on iterative erosion of the boundary. Another approach to thinning is based on local maximum distance from background [2]. But this approach is not very efficient when the width of an image is not large enough (for example, character recognition and finger-print recognition). Furthermore, the former approach can be easily parallelised while the latter cannot be. In this paper we propose a fast, rotation-invariant thin-

Correspondence to: Dr. D. Dutta Majumder, ECSU, Indian Statistical Institute, 203 BT Road, Calcutta 700 035, India.

ning algorithm which needs only one scan to check all boundary points for removal.

Section 2 presents the necessary definitions. In Section 3, the proposed thinning algorithm, called Single scan Boundary Removal Thinning Algorithm (SBRTA) is described. Implementation details are discussed in Section 4. Finally, in the concluding section, the performance of SBRTA is compared with that of SPTA [8] along with experimental results.

2. Definitions and previous works

For convenience, a few definitions which will appear often in this paper are listed below.

Definitions of 2-D binary image, $N(p)$ of a point p, 4- (8-) neighborhood, 4- (8-) connectivity, black and white components are followed as per standard convention. Nomenclature of points in $N(p)$ is shown in Figure 1.

Thinning is an image transformation that reduces a black component to a skeleton by deleting black points preserving shape and topology. Different techniques for thinning exist at present. However, the approach involving erosion of black components across their boundaries is followed in this paper.

As stated earlier, the proposed thinning algorithm is an iterative procedure, and each iteration is termed as a pass which is denoted by a pass number. A black point having at least one white 4-neighbor at the beginning of the pass is considered to be a *boundary point*.

A point is called a *significant-point* if at least two opposite 4-neighbors of the point are white at the beginning of the pass. A black point p is called a *break-point* if removal of the point creates two or more black components in $N(p)$. A boundary point is called a *final-point* if it remains in the output skeleton. A boundary point is called a *flagged-point* if it can be removed, i.e., transformed to white. So, a point is a flagged-point if it is a boundary point, but neither a significant-point, nor a break-point nor its deletion vanishes a two-points' thick protrusion.

n_3	n_2	n_1
n_4	p	n_0
n_5	n_6	n_7

Figure 1. 8-neighbors of the point p.

3. Basic theory of SBRTA

This algorithm is based on iterative erosion of boundary points preserving both shape and topology. The algorithm is fast, invariant to rotation by integral multiple of 90° and produces midline skeletons. Furthermore, the algorithm allows reconstruction of original shape like SPTA.

During each pass SBRTA checks boundary points, and either flags it for removal or marks it as a final-point belonging to the skeleton. It is worthy to note that a black point, once marked as final-point is never flagged in subsequent passes. Thus, in our algorithm each black point on the boundary is checked just once to decide whether it is a final-point or a flagged-point. During a pass different boundary points are classified as follows:

A black point is a *left boundary point* **if** n_4 is white at the beginning of the pass,
else
it is a *top boundary point* **if** n_2 is white at the beginning of the pass,
else
it is a *right boundary point* **if** n_0 is white at the beginning of the pass,
else
it is a *bottom boundary point* **if** n_6 is white at the beginning of the pass.

Thus, it should be noted that if a black point is a right boundary point then n_4 and n_2 must be black at the beginning of the pass, and so on. This information leads us to use different thresholds for setting flags to different boundary points, and is utilized for efficient detection of final-points.

At the beginning of the algorithm the value $-maxint$ is assigned to all the white points and the value zero to all the black points. We use *maxint* to denote a large positive integer. Thresholds for

removing different types of boundary points in the ith pass are set as follows:

i = pass number; (A final-point found during the ith pass is marked with the value of i.)

$c = (i-1) \times 4 - maxint$; (Each point having value less than or equal to c is white at the beginning of the pass.)

$l = c + 1$; (Each point having value equal to l is removed as a left boundary point.)

$t = c + 2$; (Each point having value equal to t is removed as a top boundary point.)

$r = c + 3$; (Each point having value equal to r is removed as a right boundary point.)

$b = c + 4$; (Each point having value equal to b is removed as a bottom boundary point.)

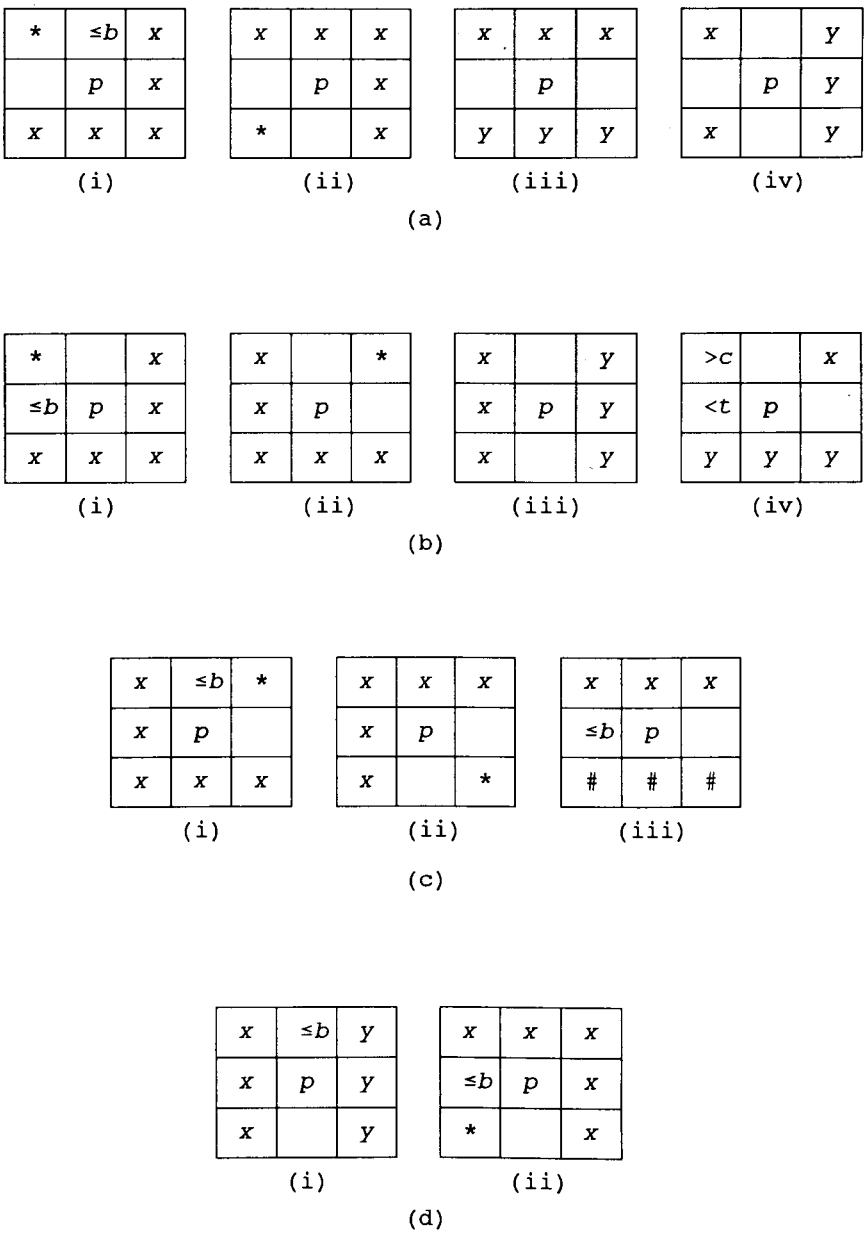

Figure 2. Window configurations for checking the (a) left, (b) top, (c) right, and (d) bottom boundary points, respectively.

Each point having value greater than b is currently black.

Unlike SPTA, SBRTA uses different sets of window configurations for different boundary points as shown in Figure 2. In these windows, (i) points marked with 'x' and 'y' are don't-cares, (ii) blank points are white at the beginning of the pass, (iii) points marked with '$*$' are currently black, and (iv) at least one of the points marked with '#' is currently black. A boundary point is a final-point if it matches a corresponding window configuration given in Figure 2(a)–(d).

3.1. Explanation for left boundary points

In window Figure 2(a)(i)&(ii), if one of the points marked with 'x' is black then p is a break-point, otherwise p is a significant-point. In window Figure 2(a)(iii)&(iv), p is always a significant-point. A left boundary point is a final-point if the boolean expression E_l is false, otherwise it is a flagged-point. E_l is given below:

$$E_l = n_0 > c \cdot (n_2 > c + n_6 > c) \cdot (n_2 > b + n_3 \leqslant b) \cdot (n_6 > b + n_5 \leqslant b).$$

Boolean operators 'AND' or 'OR' are represented by '$\cdot$' and '$+$', respectively.

3.2. Explanation for top boundary points

In window Figure 2(b)(i)&(ii), if one of the points marked with 'x' is black then p is a break-point, otherwise p is a significant-point. In window

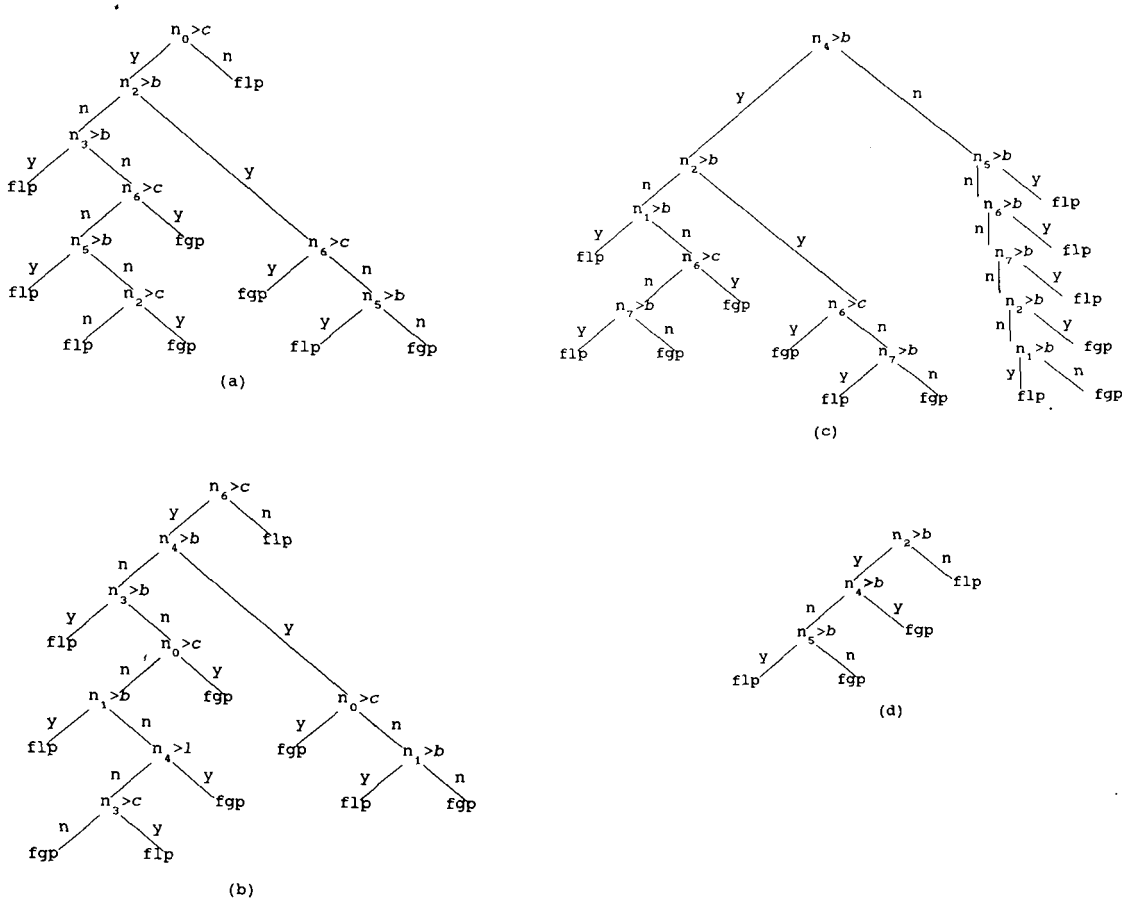

Figure 3. The decision tree structures for checking (a) left, (b) top, (c) right, and (d) bottom final-points or flagged-points. Here, 'fgp' means that p is a flagged-point, and 'flp' means that p is a final-point.

Figure 2(b)(iii), p is always a significant-point. In Figure 2(b)(iv), p is part of a two-points' thick protrusion. A top boundary point is a final-point if the boolean expression E_t is false, otherwise it is a flagged-point. E_t is given below:

$$E_t = n_6 > c \cdot (n_4 \geqslant t + n_0 > c + n_3 \leqslant c) \cdot (n_4 > b + n_3 \leqslant b) \cdot (n_0 > b + n_1 \leqslant b).$$

3.3. Explanation for right boundary points

In window Figure 2(c)(i)&(ii), if one of the points marked with 'x' is black then p is a break-point, otherwise p is part of a two-points' thick protrusion. In window Figure 2(c)(iii), p is always part of a two-points' thick protrusion. A right boundary point is a final-point if the boolean expression E_r is false, otherwise it is a flagged-point. E_r is given below:

$$E_r = (n_4 > b + n_5 \leqslant b \cdot n_6 \leqslant b \cdot n_7 \leqslant b) \cdot (n_2 > b + n_1 \leqslant b) \cdot (n_6 > b + n_7 \leqslant b).$$

3.4. Explanation for bottom boundary points

In window Figure 2(d)(i), n_0 is always black. Thus, if any of the points marked with 'x' is black then p is a break-point, otherwise p is part of a two-points' thick protrusion. In window Figure 2(d)(ii), n_0 is always black, thus p is always a break-point. It should be noted that since p is a bottom boundary point and n_0 is the same as it was at the beginning of the pass, n_0 is thus always black. It simplifies the boolean expression E_b for bottom boundary point as follows:

$$E_b = n_2 > b \cdot (n_4 > b + n_5 \leqslant b).$$

A bottom boundary point is a final-point if E_b is false, otherwise it is a flagged-point.

4. Implementation details

In the previous section, the basic theory of SBRTA has been explained. This section describes the decision trees corresponding to the boolean expressions for different types of boundary points to achieve the result with the minimum number of comparisons on an average. Also, a schematic description of the actual implementation of SBRTA is drawn.

4.1. Checking boolean expressions for boundary points

In Section 3, we have explained window configurations for different types of boundary points and corresponding boolean expressions. A close observation of these boolean expressions points out that the average number of points to be checked to reach a decision depends upon the sequence in which these points are being checked. It also depends on the nature of the image. However, it is found that the sequences described by different decision trees for different boundary points (as shown in Figure 3) are optimal on an average.

4.2. Schematic description of SBRTA

As discussed in the previous section, the value zero is assigned to every black point and the value $-maxint$ is assigned to every white point. In each pass, SBRTA needs only one scan of the entire image space in top-down row-wise fashion. During the scan in the ith pass, suppose SBRTA considers a point p. If it is a white or a marked black point then p is left unchanged and SBRTA skips to the next point. For each unmarked black point it is first decided whether the point is a left, top, right or bottom boundary point. If it is not a boundary point then also SBRTA skips to the next point. Now, for any boundary point with zero value, the corresponding boolean expression is evaluated. Depending on the result one of the following operations is performed on the point.

(i) The value $l/t/r/b$ is assigned to a left/top/right/bottom boundary point if it is a flagged-point.

(ii) Otherwise the point is a final-point and the value i is assigned to it.

Finally, the algorithm stops if no point is flagged during a pass. We have applied the algorithm on the images given in the paper of Naccache and Singhal [8]; the results are shown in Figures 4 and 5. The quality of skeleton is as good as that of [8], while execution time is half. Another result as

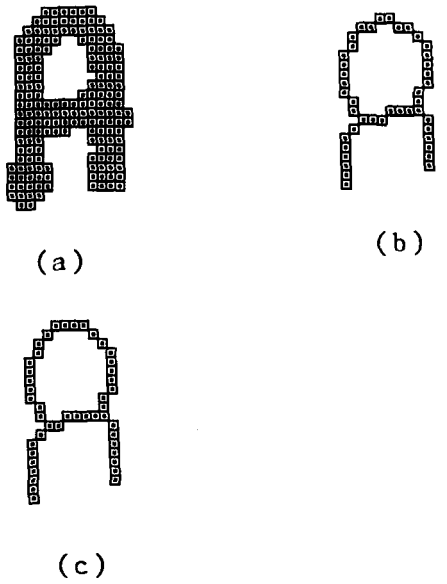

Figure 4. Experimental results of SBRTA and SPTA: (a) original image (same as Figure 7(a) of [8]), (b) skeleton obtained by SBRTA, (c) skeleton obtained by SPTA.

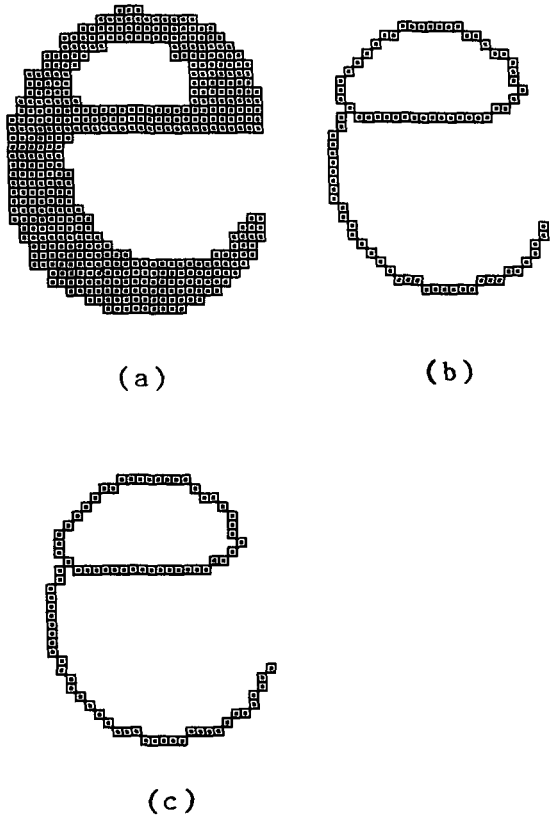

Figure 5. Experimental results of SBRTA and SPTA: (a) original image (same as Figure 8(a) of [8]), (b) skeleton obtained by SBRTA, (c) skeleton obtained by SPTA.

shown in Figure 6 reveals that SBRTA is invariant to rotation by 90° while SPTA is not.

5. Discussion and conclusion

Basic criteria of a good thinning algorithm are (i) it is efficient in computation, (ii) it produces good midline skeletons, (iii) it is rotation invariant (i.e., rotation of image should not change the overall shape of the skeleton), and (iv) removal of any point from the skeleton which has more than one black 8-neighbors changes the topology.

Both SPTA and SBRTA produce good skeleton with the property that removal of any point which has more than one black 8-neighbors changes the topology. This section draws a comparison between SBRTA and SPTA in terms of cost and rotation invariance.

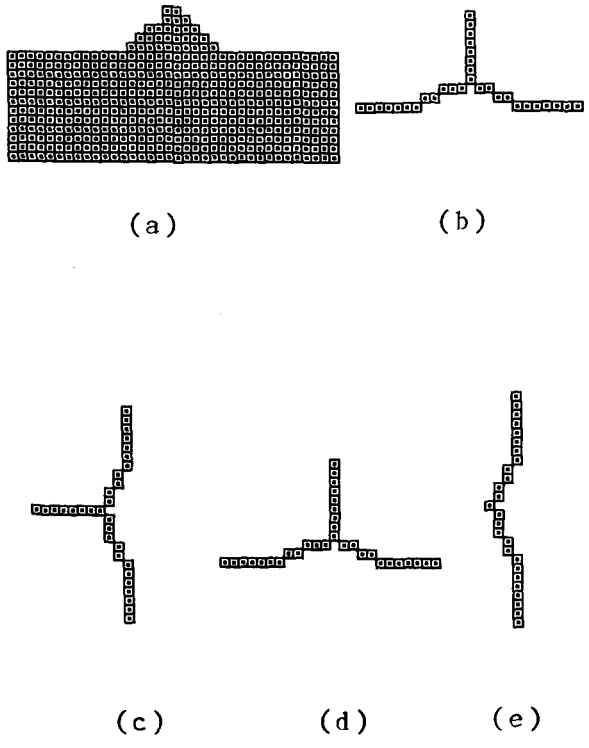

Figure 6. Comparative study between SBRTA and SPTA in terms of rotation invariance: (a) original image, (b) skeleton obtained by SBRTA, (c) skeleton obtained by SBRTA after applying 90° rotation of (a), (d) skeleton obtained by SPTA, (e) skeleton obtained by SPTA after applying 90° rotation of (a).

5.1. Comparison in terms of computational cost

The primary advantage of SBRTA over SPTA is that SBRTA needs only one scan per pass whereas SPTA needs two scans per pass. The number of branches of each decision tree corresponding to each boundary point in SPTA is eighteen. On the other hand, the number of branches of decision trees corresponding to left, top, right and bottom boundary points in SBRTA are sixteen, eighteen, twentyfour and six, respectively. Thus, the average number of branches of decision trees in SBRTA is less than that of SPTA when all types of boundary points are equally likely. Furthermore, a black point once marked as a final point by SBRTA is never checked for its removal, so each point is checked only once for removal by SBRTA. Whereas in SPTA, a black point once it becomes a boundary point is checked in every subsequent pass for removal. In the worst case when a black point is both left (or right) as well as bottom (or top) boundary point, it is checked for removal in both scans of subsequent passes. A comparative study of CPU time between SPTA and SBRTA is performed. For 26 letters (i.e., A to Z), their 90° rotated versions and mirror images, the average CPU time needed by SPTA is 644.8 millisec and that by SBRTA is 331.4 millisec.

5.2. Comparison in terms of rotation invariance

In Figure 6, we have shown an example where SBRTA is rotation invariant and SPTA is not. The logic behind this is discussed here.

SBRTA defines significant-points, two-points' thick protrusions depending on the black-and-white configuration at the beginning of each pass, and the windows are designed accordingly. On the other hand, SPTA has defined end-points, two-points' thick protrusions depending on the current black-and-white configuration. Moreover, the current black-and-white configuration in $N(p)$ depends on the direction of the scan. As a result in case of SPTA, whether or not a tail would be created from a protrusion depends on its direction, and is reflected in Figure 6.

References

[1] Arcelli, C. (1979). A condition for digital point removal. *Signal Process.* 1, 283–285.

[2] Arcelli, C. and S. de Baja (1985). A width-independent fast thinning algorithm. *IEEE Trans. Pattern. Anal. Machine Intell.* 7, 463–474.

[3] Bel-Lan, A. and L. Montoto (1981). A thinning transform for digital images. *Signal Process.* 3, 37–47.

[4] Blum, H. (1967). A transformation for extracting new descriptors of space. In: W. Wathen-Dunn, Ed., *Models for Perception of Speech and Visual Form.* MIT Press, Cambridge, MA, 362–380.

[5] Buen, M. (1973). A flexible method for automatic reading of hand-written numerals. *Philips Tech. Rev.* 31, 130–137.

[6] Hilditch, C.J. (1969). Linear skeleton from square cupboards. In: B. Meltzer and D. Michie, Eds., *Machine Intelligence,* Vol. 4. Edinburgh Univ. Press, Edinburgh, UK, 403–420.

[7] Kong, T.Y. and A. Rosenfeld (1989). Digital topology: introduction and survey. *Comput. Vision Graphics Image Process.* 48, 357–393.

[8] Naccache, N.J. and R. Singhal (1984). SPTA: a proposed algorithm for thinning binary patterns. *IEEE Trans. Syst. Man Cybernet.* 14, 409–418.

[9] Pavlidis, T. (1982). *Algorithms for Graphics and Image Processing.* Springer, Rockville, MD, 195–291.

[10] Rosenfeld, A. (1975). A characterization of parallel thinning algorithms. *Inform. Control.* 29, 286–291.

[11] Stefanelli, R. and A. Rosenfeld (1971). Some parallel thinning algorithms for digital pictures. *J. Assoc. Comput. Mach.* 18, 255–264.

[12] Tamura, H. (1978). A comparison of line-thinning algorithms from a digital geometry viewpoint. *Proc. 4th Int. Joint Conf. on Pattern Recognition,* Kyoto, Japan, 715–719.