Pattern Recognition Letters 7 (1988) 13–18
North-Holland

January 1988

A preprocessing algorithm for hand-written character recognition

W.H. ABDULLA
Electronics and Computers Research Center, Baghdad, Iraq

A.O.M. SALEH
Electrical Engineering Dept., University of Basrah, Basrah, Iraq

A.H. MORAD
Electrical Engineering Dept., University of Salahaddin, Arbil, Iraq

Received 17 April 1987
Revised 2 October 1987

Abstract: This paper describes an efficient algorithm for extracting a simplified skeletal version of hand-written characters. In this version, curvatures are represented by straight lines connecting tree-, edge- and endpoints in the right sequence. These three basic feature points are detected using an efficient algorithm.

Key words: Thinning, edge detection, character recognition.

1. Introduction

One of the most difficult problems in pattern recognition is the extraction of representative features of a given pattern. The features should, as far as possible, accurately and clearly represent the original pattern. Although there are many approaches for tackling the task of individual pattern interpretation, it is not always easily defined which one of them is the best. The choice is largely dependent on the kind of features we are looking for. Traditionally speaking, extracting and linking edgepoints has been an efficient approach for image segmentation (Pratt, 1978; Hall, 1979).

Corresponding author: W.H. Abdulla, Electronics and Computers Research Center, P.O. Box 2131, Jadriyah, Baghdad, Iraq.

In this paper, the hand-written character is represented by its edge-, tree- and endpoints. These points are extracted from the unitary skeleton of the character. A pre-requirement to implement our technique is that the original character should be represented by a unitary skeleton. A unitary skeleton is a single pixel thickness skeleton, in which each of its pixels is connected to not more than two adjacent pixels unless it represents a treepoint. Methods for extracting the skeleton of a character are well known (Zhang and Suen, 1984; Carlo and Di Baja, 1985). Initially it was decided to follow an algorithm developed by Zhang and Suen, but it was discovered that this algorithm does not always give a unitary skeleton. While it is possible to extract tree-, edge- and endpoints with a non-unitary skeleton, the extraction can be easily accomplished with the unitary skeleton. Thus, a modified algorithm is developed to achieve this task. Once edge-, tree-

and endpoints are specified, straight lines connecting them in the right consequence are then formed. This will give a simplified version of the character.

2. Thinning procedure

This procedure is required to reduce the handwritten character into the unitary skeleton form. The input character is superposed on a 24 × 32 element matrix. Each element is assigned the value '1' if it is covered by part of the character, and the value '0' otherwise. Deciding whether a point is skeletal or not depends on its eight neighbours. Thus, a window of 3 × 3 pixels is used as shown in Figure 1. Firstly, the skeleton of the character is extracted as proposed by Zhang and described in the appendix. Take, for example, the input character shown in Figure 2a: the output skeleton is shown in Figure 2b. Clearly the output skeleton is not unitary as it involves distortion at some points. This distortion will cause difficulties in the detection of edge- and treepoints. To overcome this drawback a procedure is developed which is capable of producing noise-free unitary skeleton. This procedure involves two iterations as follows:

Iteration 1

The skeleton is scanned horizontally by the 3 × 4 pixels window shown in Figure 3. Any two points which are horizontally adjacent to each other and horizontally isolated from other points, are detected as shown in Figure 2b. With P_1 and P_4 representing these two points, apply the following tests to check whether one of them is redundant:

(a) P_1 is deleted if one of the following conditions is true.

(1) $\overline{\mathrm{SP}_1} \wedge P_6 = 1$;
(2) $\overline{\mathrm{SP}_2} \wedge P_2 = 1$;
(3) $[(P_2 \wedge \overline{P_3}) \vee (P_3 \wedge \overline{P_2} \wedge \overline{P_9})]$
$\wedge [(\overline{P_5} \wedge P_6) \vee (P_5 \wedge \overline{P_6} \wedge \overline{P_7})]$;

where

$$\mathrm{SP}_1 = P_3 \vee P_2 \vee P_9, \quad \mathrm{SP}_2 = P_6 \vee P_5 \vee P_7,$$

and $\overline{\ }$, $\vee$, $\wedge$ are complement, logical 'OR' and logical 'AND' respectively.

P_9	P_2	P_3
P_8	P_1	P_4
P_7	P_6	P_5

Figuur 1. The 3 × 3 pixels window for point P_1.

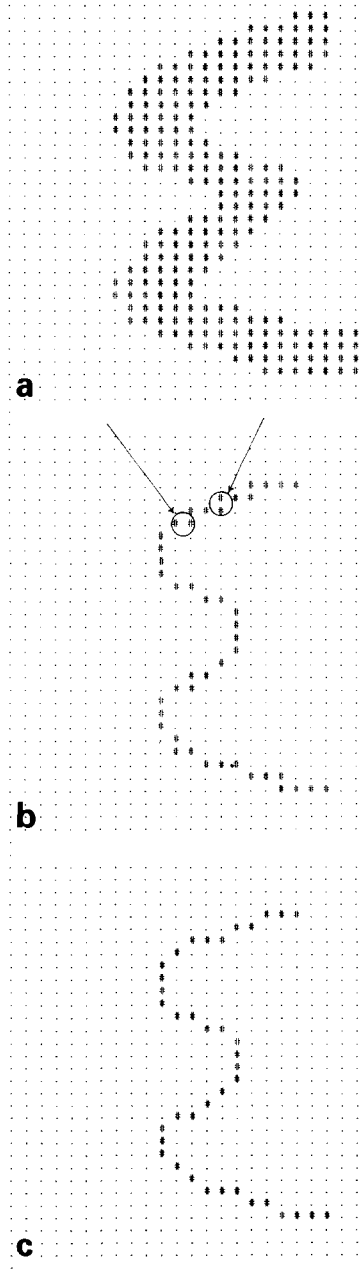

Figure 2. (a) A hand-written character; (b) Its non-unitary skeleton (left arrow: horizontally isolated, right arrow: vertically isolated); (c) Its unitary skeleton.

P_9	P_2	P_3	P_{10}
P_8	P_1	P_4	P_{11}
P_7	P_6	P_5	P_{12}

Figure 3. The 3 × 4 pixels window for Iteration 1.

(b) If P_1 is not redundant then P_4 must be deleted if the following condition is not true.

$$(\overline{P_3} \wedge P_{10}) \vee (\overline{P_5} \wedge P_{12}).$$

Iteration 2

In this iteration the skeleton is scanned vertically by the 4 × 3 pixels window shown in Figure 4. Any two points which are vertically adjacent to each other and vertically isolated from other points are detected, as shown in Figure 2b. With P_1 and P_4 representing these points, apply the same tests of Iteration 1 to locate the redundant points.

The output skeleton resulting from applying the above two iterations to Figure 2b is shown in Figure 2c.

3. Detection process

The detection process involves detecting three features in the skeleton; these are tree-, end- and edgepoints.

3.1. End- and treepoints detection

The tree- and endpoints are detected by applying the previous 3 × 3 pixels window to the unitary skeleton. A point in the skeleton is an endpoint if it satisfies the following condition:

$$B(P_1) = 1,$$

where $B(P_1)$ is the number of neighbour points of P_1 as defined in the Appendix.

On the other hand, a point is considered as a treepoint if it satisfies the following condition:

$$B(P_1) \geqslant 3.$$

Any two or more adjacent treepoints are represented by only the first detected one, or prefarably, by the one in their center of gravity.

P_9	P_2	P_3
P_8	P_1	P_4
P_7	P_6	P_5
P_{12}	P_{11}	P_{10}

Figure 4. The 4 × 3 pixels window for Iteration 2.

3.2. Edge detection

Having detected the end- and treepoints in the skeleton, the detection of edgepoints can be started. Firstly, all the detected tree- and endpoints are taken as edgepoints directly. Then, the process of detecting edgepoints other than those already being specified can be started. While the detection process must start at an endpoint, such an endpoint can be chosen randomly. If this point is considered initially as a reference point, the algorithm proceeds as follows:

Step 1. Calculate the slope (S_1) of the line connecting the reference point and its neighbour. This slope has one of the eight possible values shown in Figure 5.

Step 2. Calculate the slope (S_2) of the next two successive points as shown in Figure 6.

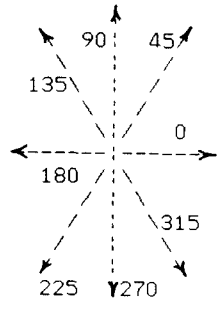

Figure 5. The '8' possible directions.

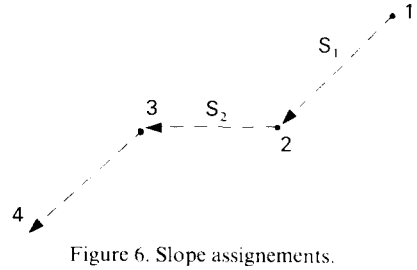

Figure 6. Slope assignements.

Step 3. If S_2 satisfies one of the following two conditions, then the first of the two successive points mentioned in Step 2 is redundant. If there are more points to process, go to Step 2, otherwise stop.

Condition 1. For $S_1 = 0,90,180$ or 270:
$S_1 - 90 \leqslant S_2 \leqslant S_1 + 90$.
Condition 2. For $S_1 = 45,135,225$ or 315:
$S_1 - 45 \leqslant S_2 \leqslant S1 + 45$.

Step 4. If S_2 does not satisfy any of the above two conditions, then the first of the two successive points mentioned in Step 2 is an edgepoint.

Note that if the skeleton has no end- or tree-points, as in circular shapes, then the above algorithm can be started at any point in the skeleton.

4. Examples

The performance of the algorithm has been proved by applying it to all Arabic and English alphanumeric characters. The results of five samples are shown in Figure 7. It is also useful to mention that this algorithm has been used successfully in the preprocessing of Arabic characters (Morad et al., 1987) and enabling the application of line detection techniques for the recognition process (Saleh et al., 1987).

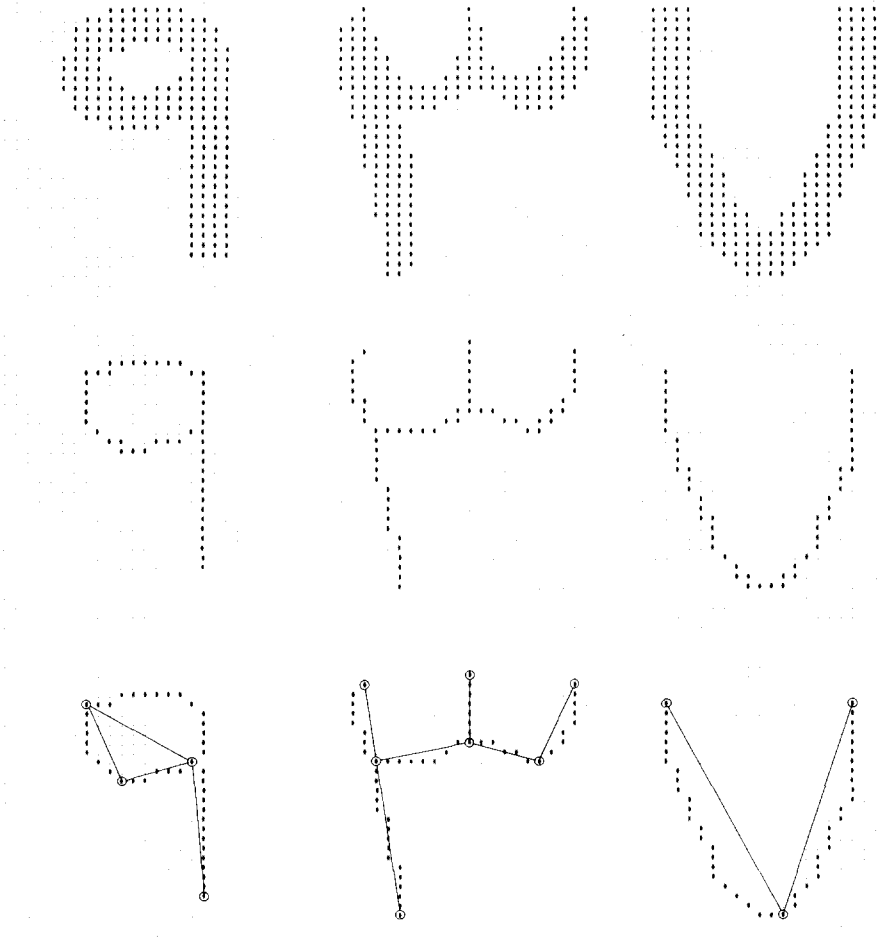

Figure 7a. Preprocessing of three Arabic characters.

The time spent, in our algorithm, to obtain the unitary skeleton compared to the time of the whole thinning procedure of five samples is shown in Table 1.

Table 1
Preprocessing time of five characters

Character	Thinning processing time (seconds)	Extra time for unitary skeleton (seconds)
R	13.4	1.39
S	9.52	1.33
٩	8.14	1.07
٣	12.8	1.66
٧	10.5	1.58

5. Conclusion

Extracting the skeleton of hand-written characters is both necessary and essential for an efficient

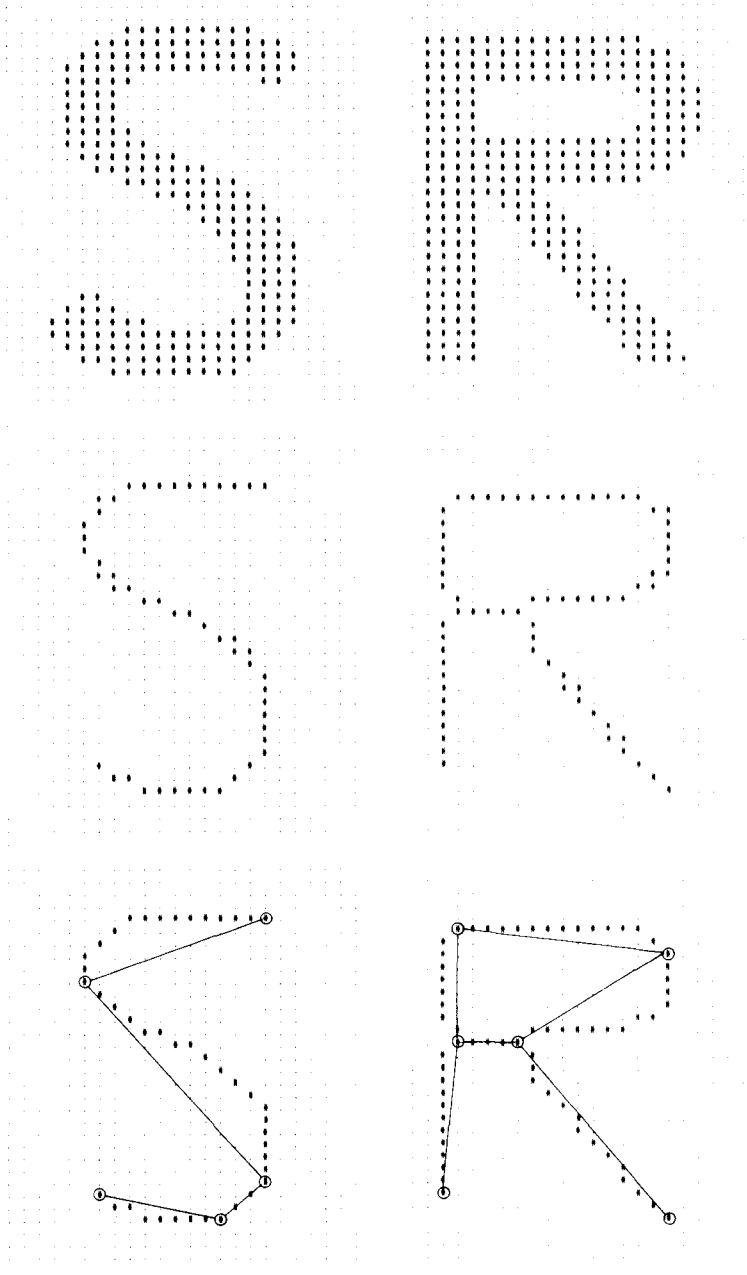

Figure 7b. Preprocessing of the characters R and S.

character recognition process. Investigations show that obtaining a simplified skeletal version, though it will add an extra effort in the preprocessing stage, will ease numerous difficulties in the recognition process (Morad et al., 1987; Badi and Shimura, 1979). For example, recognition of characters which are only composed of straight lines can be carried out easily and accurately. Unfortunately, most hand-written characters do not satisfy this condition. The developed algorithm facilitates the formation of a simplified version of the character such that it is composed of straight lines only. The algorithm achieves this by extracting three basic features of the character, which are tree-, edge- and endpoints. These points are then joined in the consequence by straight lines resulting a simplified skeletal version of the character.

Strictly speaking, it has been found that the extra time spent is almost negligible, considering the gain in time and recognition rate obtained at the recognition process.

The extra time spent in our algorithm to obtain the unitary skeleton was not more than 2 seconds, when the algorithm is achieved on a HP-9000 series 200 computer and using BASIC language.

Acknowledgement

The authors would like to thank Prof. M.N. Al-Tikriti, D.G. of the Electronics & Computers Research Center (ECRC) in Baghdad and G.T. Ibrahim, Head of the Computers Dept. for support and facilities provided by the center.

Appendix

This algorithm involves two subiterations as follows:

Iteration I

The point (P_1) is deleted from the matrix 'A' if it satisfies the following conditions:

(a) $2 \leqslant B(P_1) \leqslant 6$, (b) $A(P_1) = 1$,
(c) $P_4 \wedge P_6 = 0$, (d) $P_2 \wedge P_8 = 0$,

where

$$A(P_1) = (\overline{P_2} \wedge P_3) + (\overline{P_3} \wedge P_4) + (\overline{P_4} \wedge P_5) + (\overline{P_5} \wedge P_6) + (\overline{P_6} \wedge P_7) + (\overline{P_7} \wedge P_8) + (\overline{P_8} \wedge P_9) + (\overline{P_9} \wedge P_2)$$

and

$$B(P_1) = \sum_{i=2}^{9} P_i.$$

Iteration II

In this iteration the point is deleted if it satisfies the following conditions:

(a) $2 \leqslant B(P_1) \leqslant 6$, (b) $A(P_1) = 1$,
(c) $P_2 \wedge P_8 = 0$, (d) $P_4 \wedge P_6 = 0$.

References

Badi, K. and M. Shimura (1979). Machine recognition of Arabic cursive scripts. In: *Pattern Recognition In Practice*. North-Holland, Amsterdam.

Carlo, C. and G.S. di Baja (1985). A width-indepent fast thinning algorithm, *IEEE Tran.*, Vol. PAMI-7. No. 4, pp 463–474.

Hall, E.L. (1979). *Computer Image Processing And Recognition*, Academic Press, New York.

Morad, A.H., A.O.M. Saleh and W.H. Abdulla (1987). Arabic characters recognition using a curve-to-line transformation technique, To be submitted for publication.

Pratt, W.K. (1978). *Digital Image Processing*, Wiley, New York.

Saleh, A.O.M., A.H. Morad and W.H. Abdulla (1987). New technique for lines and curves detection, Submitted for publication in Pattern Recognition Journal.

Zhang, T.Y. and C.U. Suen (1984). A fast parallel algorithm for thinning digital patterns, *Comm. of the ACM*, Vol. 27, No. 3, pp 236–239.