Journal of Electronic Imaging 4(3), 298–304 (July 1995).

Feature-preserving thinning algorithm for optical character recognition

Hsin-Chia Fu
Ting-Shan Cheng
Cheng-Chin Chiang
Shing-Ming Roan
National Chiao-Tung University
Department of Computer Science and Information Engineering
Hsinchu, Taiwan 300
E-mail: hcfu@csie.nctu.edu.tw

Abstract. *We propose an improved thinning algorithm. Basically, the algorithm uses a 3×3 window to accommodate an eight-neighbor skeleton in each thinning iteration. This algorithm overcomes many thinning problems, such as Y-shaped distortions, spiky skeletons, and skeleton shortening, and thus preserves precise features of digital character patterns. By using this thinning algorithm, better structural features of a character pattern can be provided to an optical character recognition system, such that accurate recognition results can be achieved.*

1 Introduction

Thinning is usually regarded as the process of deleting boundary pixels until all strokes are one pixel in width without disturbing the original configuration and connection of a character pattern. To date, many thinning algorithms[1,2] have been reported to perform well on patterns with smooth boundaries of even width. However, character patterns in real-world applications are usually not smooth at boundaries and even in width. Problems commonly resulting from existing thinning processes, as shown in Fig. 1, include Y-shaped distortion, spiky skeletons, and shortened skeletons. To achieve high recognition rate, an optical character recognition (OCR) system needs precise skeleton features from the original character patterns to provide better feature-matching attributes. If, for instance, a skeleton displays Y-shaped distortion or spiky skeletons, then an OCR system may end up using the wrong feature attributes and identifying a character incorrectly.

Paper 94-015 received Apr. 19, 1994; revised manuscript received Mar. 10, 1995; accepted for publication Mar. 21, 1995. 1017-9909/95/$6.00.

Most previously proposed thinning algorithms apply local operators to decide whether a pixel should be removed. As shown in Fig. 1(b), the thinning process sometimes destroys the straightness of a horizontal stroke, especially at its junction with a vertical stroke. When a Y-shaped skeleton is used as a structural feature in the recognition process, instead of the proper T-shaped skeleton, a "T" character may erroneously be recognized as a "Y" or "r" character. We suggest that to preserve original stroke features, global attributes should also be considered in the thinning procedure. In this paper, we present a new 3×3 window-based thinning method that considers both local and global attributes in each thinning iteration. Experimental results show that our thinning method can preserve precise skeleton features of the original character patterns.

The rest of the paper is organized as follows. In Sec. 2, we review existing thinning algorithms and discuss their major problems; then we propose a new thinning method, which is designed to overcome these problems. Section 3 presents experimental results of our method and compares our method with other thinning algorithms. Section 4 presents our conclusions and plans for future work.

2 Feature-Preserving Thinning

2.1 *Previous Research and Related Problems*

Zhang and Suen[3] proposed a fast thinning algorithm with two subiterations, one for removing the southeast boundary and the northwest corner pixels, the other for deleting the northwest boundary and the southeast corner pixels. Since the operations performed in this algorithm can be independently performed on each pixel, it is very easy to map this algorithm on a parallel computer for speed improvement.

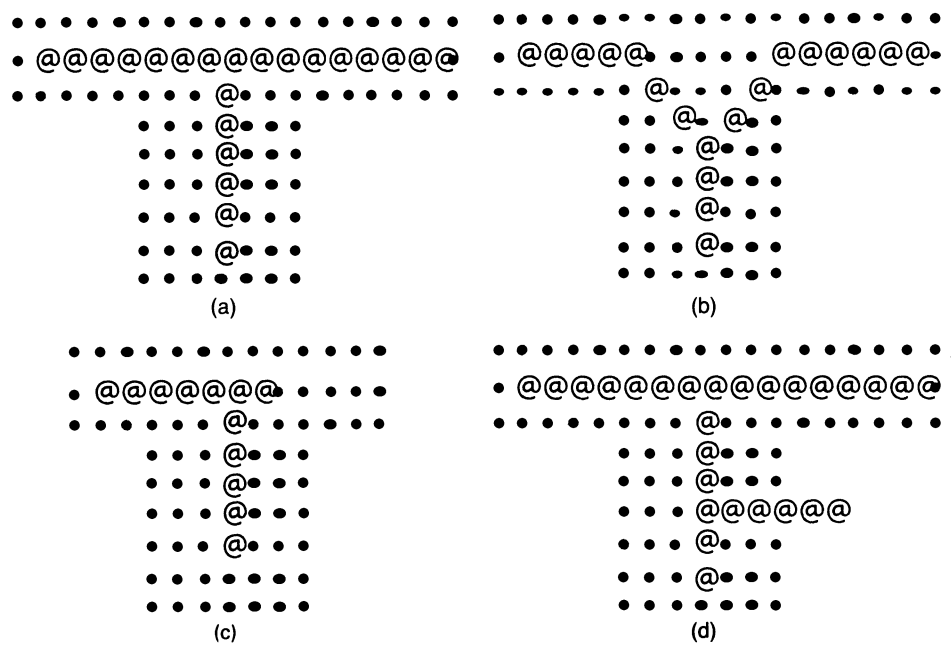


Fig. 1 Thinning results for a "T" character: (a) a desired skeleton, (b) a Y-shaped distortion, (c) shortened skeletons, and (d) spiky skeletons.

Their method generates 4-neighbor connected skeletons from an input pattern. In other words, the pixels in the pattern after the thinning process are allowed to connect to other pixels in only four (i.e., left, right, up, and down) directions. By including some extra checking conditions, Chen and Hsu[4] modified Zhang and Suen's algorithm to generate 8-neighbor connected skeletons. (See Sec. 5 for a brief introduction to Chen and Hsu's algorithm.) Therefore, the pixels in the thinned patterns have better connectivities, because they can be connected in eight (left, right, up, down, left-up, right-up, left-down, and right-down) directions. In both algorithms, whether a pixel is removed or not depends only on the status of its 8-neighbor pixels. A snapshot of pixel patterns during an application of Chen and Hsu's thinning process is shown in Fig. 2; in this case pixels *a, b, c, d, e,* and *f* would be deleted and pixel *x* would be retained in the current thinning iteration. As we can see, the removal of pixels *a, b,* and *c* will cause a Y-shaped distortion, the removal of pixels *d, e,* and *f* will result in a shortened skeleton, and the retention of pixel *x* will create a spiky skeleton. In the next section, we will describe the proposed new thinning method that aims to solve these problems.

2.2 *Feature-Preserved Digital Thinning*

2.2.1 *Prevention of Y-shaped distortion and skeleton shortening*

In Fig. 2, we see that the Y-shaped distortion and skeleton shortening result from the removal of too many pixels (i.e., pixels *a, b, c, d, e,* and *f*) from the two ends of a thick stroke. Before introducing our method, we first define the following terms:

Scan Trace. For each pixel $P(i,j)$ in a character pattern, we define four scan traces in the following orientations: west-east, northwest-southeast, north-south, and northeast-southwest. The four traces extending from $P(i,j)$ are defined as follows (see Fig. 3):

$T_{w\leftrightarrow e}(i,j)$ = a line of pixels in the input pattern passing through $P(i,j)$ horizontally

$T_{nw\leftrightarrow se}(i,j)$ = a line of pixels in the input pattern passing through $P(i,j)$ at an angle of 45 deg

$T_{n\leftrightarrow s}(i,j)$ = a line of pixels in the input pattern passing through $P(i,j)$ vertically

$T_{ne\leftrightarrow sw}(i,j)$ = a line of pixels in the input pattern passing through $P(i,j)$ at an angle of 135 deg.

Scan length. The length $L_d(i,j)$ of a scan trace of $P(i,j)$ in the orientation d is defined as follows: $L_d(i,j)$ is the number of pixels in a scan trace along the orientation d, where d can be one of the four orientations $w\leftrightarrow e$, $nw\leftrightarrow se$, $n\leftrightarrow s$, or $ne\leftrightarrow sw$. As shown in Fig. 4, there are four scan traces passing through $P(3,1)$.

$$T_{w\leftrightarrow e}(3,1)=[P(3,1),P(3,2),P(3,3)],$$

$$T_{nw\leftrightarrow se}(3,1)=[P(3,1),P(4,2),P(5,3),P(6,4)],$$

$$T_{n\leftrightarrow s}(3,1)=[P(0,1),P(1,1),P(2,1),P(3,1),P(4,1),P(5,1)],$$

$$T_{ne\leftrightarrow sw}(3,1)=[P(2,2),P(3,1),P(4,0)] \quad . \tag{1}$$

The lengths of the four scan traces are $L_{w\leftrightarrow e}(3,1)=3$, $L_{nw\leftrightarrow se}(3,1)=4$, $L_{n\leftrightarrow s}(3,1)=6$, and $L_{ne\leftrightarrow sw}(3,1)=3$.

For each pixel $P(i,j)$, we define three logical operators in each scan orientation: directional AND (DAND), directional

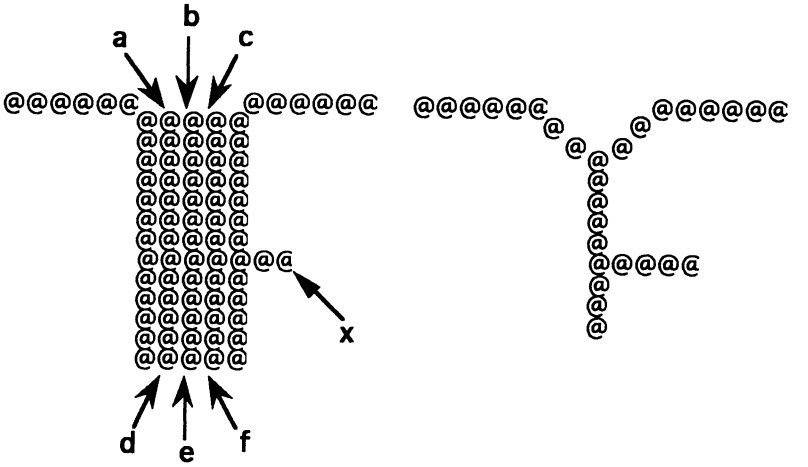


Fig. 2 A snapshot of the thinning process. Previous thinning algorithms result in a Y-shaped distortion, a shortened and spiky skeleton.

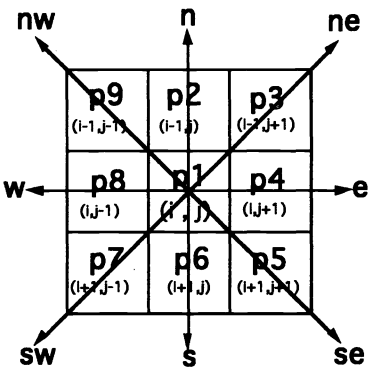


Fig. 3 The four scan traces and eight neighbors of $P(i, j)$.

left OR (DORL), directional right OR (DORR). The directional left (or right) means to the left (or right) side of a scan line. These operators along the four orientations are defined as follows:

Directional AND:

$$\mathrm{DAND}_{w\leftrightarrow e}[P(i,j)] = P(i,\ j-1) \wedge P(i,\ j+1)$$

$$\mathrm{DAND}_{nw\leftrightarrow se}[P(i,j)] = P(i-1,\ j-1) \wedge P(i+1,\ j+1),$$

$$\mathrm{DAND}_{n\leftrightarrow s}[P(i,j)] = P(i-1,\ j) \wedge P(i+1,\ j),$$

$$\mathrm{DAND}_{ne\leftrightarrow sw}[P(i,j)] = P(i-1,\ j+1) \wedge P(i+1,\ j-1).$$

Directional left OR:

$$\mathrm{DORL}_{w\leftrightarrow e}[P(i,j)] =$$

$$P(i+1,\ j-1) \vee P(i+1,\ j) \vee P(i+1,\ j+1),$$

$$\mathrm{DORL}_{nw\leftrightarrow \mathrm{se}}[P(i,j)] =$$

$$P(i,\ j-1) \vee P(i+1,\ j-1) \vee P(i+1,\ j),$$

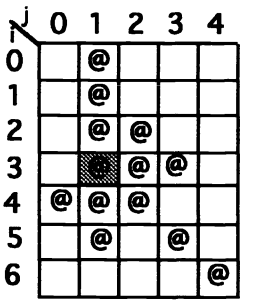


Fig. 4 Scan traces and scan length for the pixel $P(3, 1)$.

$$\mathrm{DORL}_{n\leftrightarrow s}[P(i,j)] =$$

$$P(i-1,\ j-1) \vee P(i,\ j-1) \vee P(i+1,\ j-1),$$

$$\mathrm{DORL}_{ne\leftrightarrow sw}[P(i,j)] =$$

$$P(i-1,\ j) \vee P(i-1,\ j-1) \vee P(i,\ j-1)\ .$$

Directional right OR:

$$\mathrm{DORR}_{w\leftrightarrow e}[P(i,j)] =$$

$$P(i-1,\ j-1) \vee P(i-1,\ j) \vee P(i-1,\ j+1),$$

$$\mathrm{DORR}_{nw\leftrightarrow se}[P(i,j)] =$$

$$P(i-1,\ j) \vee P(i-1,\ j+1) \vee P(i,\ j+1),$$

$$\mathrm{DORR}_{n\leftrightarrow s}[P(i,j)] =$$

$$P(i-1,\ j+1) \vee P(i,\ j+1) \vee P(i+1,\ j+1),$$

$$\mathrm{DORR}_{ne\leftrightarrow sw}[P(i,j)] =$$

$$P(i,\ j+1) \vee P(i+1,\ j+1) \vee P(i+1,\ j)\ .$$

In these definitions, $P(i,j) = 1$ if the pixel at $P(i,j)$ is a foreground pixel in the character pattern, and $P(i,j) = 0$ if the pixel is a blank.

To avoid *Y*-shaped distortion and skeleton shortening problems like those in Fig. 2, we propose the following criteria for checking whether an edge pixel is deletable or not. If a pixel satisfies the following two conditions, then the pixel will not be removed in the current thinning iteration.

Condition 1. To check if the current stroke is long and thin:

Suppose a pixel $P(i,j)$ is at a long-and-thin stroke. Then there exists a scan trace $T_d(i,j)$ such that $L_d(i,j) > \delta * L_{d'}(i,j)$ for all $d' \neq d$, where d' and d are the orientation of the scan trace passing through $P(i,j)$. The term δ is the scan length ratio constant, which is the length-to-width ratio of the scan length. According to our experience, δ is suggested to be 2 or larger.

Condition 2. To check if an edge pixel is on the terminal edges of a long-and-thin stroke:

Suppose the orientation of the long-and-thin stroke is d. If a pixel point $P(i,j)$ satisfies the following conditions, then $P(i,j)$ is at the terminal edges of the stroke.

$$\mathrm{DAND}_{d \leftrightarrow d'}[P(i,j)] = 0 \ , \tag{2a}$$

$$\mathrm{DORL}_{d \leftrightarrow d'}[P(i,j)] = 1, \quad \text{and} \tag{2b}$$

$$\mathrm{DORR}_{d \leftrightarrow d'}[P(i,j)] = 1 \ . \tag{2c}$$

Condition 1 is used to decide whether a pixel is on a long-and-thin stroke or on a short-and-thick stroke. The scan ratio constant δ is a threshold to distinguish a stroke to be one of these two cases. If the length-to-width ratio of a stroke is equal to or larger than δ, then the stroke can be considered a long-and-thin stroke in applying our algorithm.

In Fig. 5(a), each long-and-thin stroke consists of four edges (a,b,c,d); edges a and b are the side edges, and edges c and d are the terminal edges. To avoid skeleton shortening problems and to preserve the original length of the stroke, we need to delete the light-gray pixels (i.e., the side pixels) and leave the dark-gray pixels (i.e., the terminal pixels). Thus, when a pixel point $P(i,j)$ satisfies Condition (2a), $P(i,j)$ is on the terminal edge (i.e., edge c or d in Fig. 5(a)). In fact, Condition (2a) determines whether pixel point $P(i,j)$ and its two neighboring pixels along the longest scan trace are all

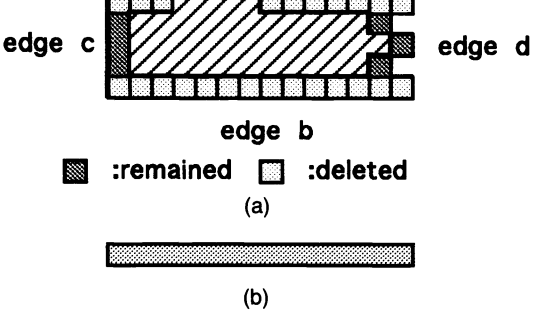


Fig. 5 A thinning example: (a) The original stroke is shown, with the boundary pixels depicted by gray shading; pixels along edges c and d should be preserved. (b) The thinning result.

1's or not. Conditions (2b) and (2c) are used to distinguish pixels at a corner [see Fig. 5(a)] from pixels at the terminal edges. To preserve the long-and-thin features of a stroke during a thinning iteration, pixels at a corner are likely to be removed, but pixels that lie on terminal edges [the dark-gray pixels in Fig. 5(a)] must be preserved. In Fig. 5(a), for example, the dark-gray pixels satisfy Conditions (2b) and (2c), but the corner pixels do not. If a pixel satisfies both Condition 1 and Condition 2, the pixel should be retained. Otherwise, Y-shaped distortions and shortened skeletons may occur if further thinning iterations are performed. As shown in Fig. 5(b), by using our thinning algorithm, the final results present a precise skeleton with the same length as the original stroke.

2.2.2 *Prevention of spiky skeletons*

When we are trying to apply Chen and Hsu's thinning algorithm[4] on some noisy character patterns, we see that spiky skeletons usually originate from two adjacent pixels that are attached perpendicularly to a block of a stroke pattern, as shown in Fig. 6. After several thinning iterations, the two adjacent pixels stand out and become longer and longer. Since Chen and Hsu's algorithm considers pixels x and y to be the terminal pixels, pixel x will not be stripped out in any of the thinning iterations.

We shall modify Chen and Hsu's algorithm to prevent the generation of spiky skeletons. In Zhang and Suen's algorithm, variable $B(P1)$ is defined to be the number of nonzero neighbors of pixel $P1$ (see Fig. 3). In both the Zhang-Suen and Chen-Hsu algorithms, all the terminal pixels [$B(P1) = 1$] are retained. As shown in Fig. 6, some terminal pixels, such as the pixel x, should be deleted, otherwise a spiky skeleton may be created after several thinning iterations. Before we state our spike-free method, we need to define a reference pixel.

Reference Pixel. If a terminal pixel x has only one neighbor pixel, then we shall call this neighbor pixel its *reference pixel.*

For each terminal pixel x, we propose the following procedure for preventing spiky skeletons:

1. Temporarily assume the pixel x does not exist in the input pattern (i.e., let $x = 0$) and then use Chen and Hsu's criteria to check whether its reference pixel y is deletable (in the next iteration).
2. If pixel y is deletable, then delete pixel x; otherwise, retain pixel x.

Note that the checking on pixel y is only to determine whether pixel x will be removed, rather than to determine the removal of pixel y. In our algorithm, for the situation in which three or more connected pixels attach perpendicular to a stroke,

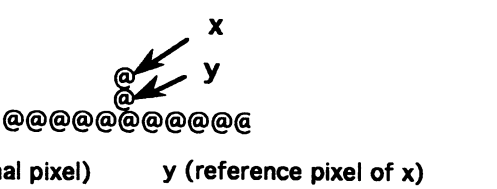


Fig. 6 Pixel y is a reference pixel of x. The terminal pixel x should be deleted.

these connected pixels are considered to be a normal skeleton instead of a noisy skeleton.

2.2.3 *Feature-preserving thinning algorithm*

This algorithm is applied for each thinning iteration, and the iterations terminate when no more pixels can be deleted. In the algorithm, the variable $A(P(1))$ represents the number of pattern state changes ($0 \rightarrow 1$) among the neighboring pixels clockwise around the pixel $P1$ (i.e., pixels $P2$, $P3$, $P4$, $P5$, $P6$, $P7$, $P8$, $P9$, $P2$, as shown in Fig. 3). The variable $B(P(1))$ is the number of nonzero neighbors of pixel $P1$. The algorithm consists of two subiterations.

Input: pattern $P(i,j)$, $0<i, j<M$, where M is the dimension of the pattern,
Output: next subiteration pattern $P'(i,j)$ of $P(i,j)$
begin
  compute $B(P(i,j))$ for all i, j such that $P(i,j)=1$;
  if $B(P(i,j))=1$ (i.e., $P(i,j)$ is a terminal pixel)
    begin
      apply Chen and Hsu's checking conditions to the reference pixel of $P(i,j)$;
      if it satisfies the conditions of Chen and Hsu's algorithm
      then $P'(i,j) \leftarrow 0$ (delete $P(i,j)$)
      else $P'(i,j) \leftarrow 1$;
    end
  else if $B(P(i,j))>1$
    begin
      apply Chen and Hsu's conditions to $P(i,j)$;
      if $P(i,j)$ satisfies the conditions of Chen and Hsu's algorithm
      begin
        compute $d_{w \leftrightarrow e}(P(i,j))$, $d_{nw \leftrightarrow se}(P(i,j))$, $d_{n \leftrightarrow s}(P(i,j))$, $d_{ne \leftrightarrow sw}(P(i,j))$;
        if there exists a scan trace $l \leftrightarrow l'$ such that $d_{l \leftrightarrow l'}(P(i,j)) > \delta * d_{r \leftrightarrow r'}(P(i,j))$ for any of the remaining scan traces $r \leftrightarrow r'$, and
          $\mathrm{DAND}_{l \leftrightarrow l'}[P(i,j)]=0$, and
          $\mathrm{DORL}_{l \leftrightarrow l'}[P(i,j)]=1$, and
          $\mathrm{DORR}_{l \leftrightarrow l'}[P(i,j)]=1$
        then $P'(i,j) \leftarrow 1$
        else $P'(i,j) \leftarrow 0$ (delete $P(i,j)$);
      end
    end
end

3 Experimental Results and Comparisons

In this section, we present the experimental results of the proposed algorithm and compare results from the Zhang-Suen[3] and Chen-Hsu[4] algorithms. In Fig. 7, the left column shows the thinning results for the proposed algorithm, the center column shows the results for the Chen-Hsu algorithm, and the right column shows the results for the Zhang-Suen algorithm. As we can see, there are serious Y-shaped distortion and spiky skeletons in the thinned "T" and "P" characters by using the Chen-Hsu and Zhang-Suen algorithms. For the "H" and "K" characters, the skeleton shortening problem occurs during the thinning results of the Chen-Hsu and Zhang-Suen algorithms. However, our method provides better thinning results in dealing with the problems of Y-shaped distortion, and shortened and spiky skeletons. Thus, more original stroke features are preserved.

4 Conclusions

A new feature-preserving thinning algorithm with better thinning results is proposed in this paper. The proposed algorithm preserves the original strokes and junction features and prevents spiky skeletons better than the Zhang-Suen and Chen-Hsu algorithms. Since our algorithm requires more computation than the previous algorithms, we are now working on a parallel and pipeline architecture of this proposed algorithm for on-line or real-time OCR systems.

5 Appendix: Fast Parallel Thinning Algorithm

The fast parallel thinning algorithm was proposed by Zhang and Suen,[3] and later was modified by Chen and Hsu.[4] Basically, this algorithm contains two subiterations, where each iteration consists of a set of checking conditions. When a pixel $P(1)$ satisfies these conditions, it is then deleted from the original digital pattern. In the following listing, the conditions that were modified by Chen and Hsu are preceded with a symbol *. The variables $A(P1)$ and $B(P1)$ are defined to be the number of transitions from black pixels to white pixels among the eight neighbors of $P1$ in clockwise order and the number of black pixels in the eight neighbors of $P1$, respectively. In the first subiteration,

*(a) $2 \leq B(P1) \leq 7$;

(b) if $A(P1)=1$:

(c) $P2*P4*P6=0$, and

(d) $P4*P6*P8=0$;

*(e) if $A(P1)=2$:

*(f) $P2*P4=1$ and $P6+P7+P8=0$, or

*(g) $P4*P6=1$ and $P2+P8+P9=0$.

In the second subiteration, only (c), (d), (f), and (g) are changed as follows:

(c′) $P2*P4*P8=0$,

(d′) $P2*P6*P8=0$,

*(f) $P2*P8=1$ and $P4+P5+P6=0$,

*(g′) $P6*P8=1$ and $P2+P3+P4=0$.

Acknowledgment

This work was supported in part by the National Science Council under grant NSC 82-0408-E-009-427.

References

1. R. W. Smith, "Computer processing of line images: a survey," *Patt. Recog.* **20**(1), 7–15 (1987).
2. H. E. Lu and P. S. P. Wang, "A comment on 'A fast parallel algorithm for thinning digital patterns,' " *CACM* **29**(3), 239–242 (1986).
3. T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *CACM* **27**(3), 236–239 (1984).
4. Y. S. Chen and W. H. Hsu, "A modified fast parallel algorithm for thinning digital patterns," *Patt. Recog. Lett.* **7**, 99–106 (1988).

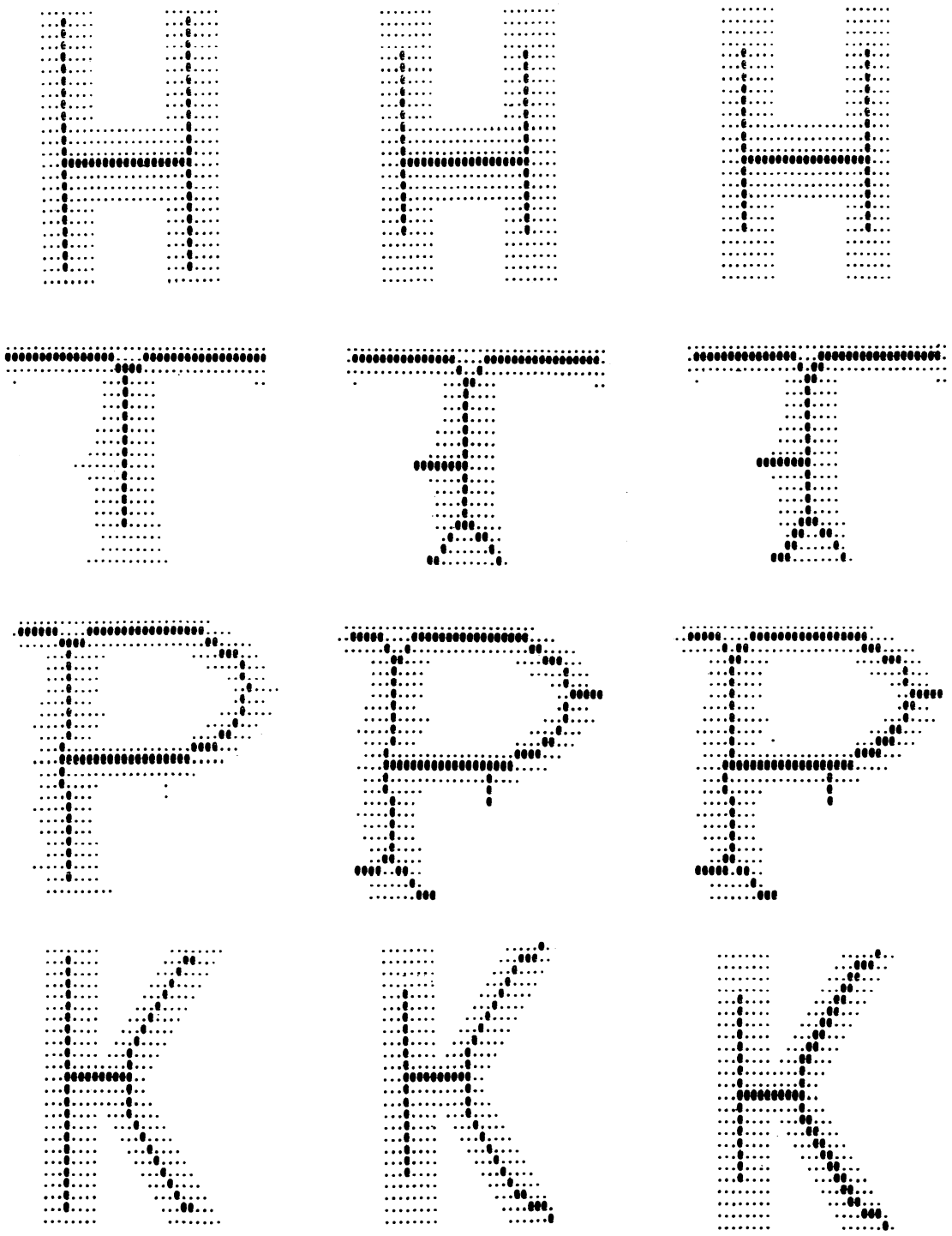
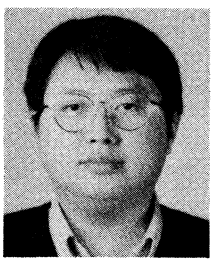


Fig. 7 Thinning results for the characters "H", "T", "P", and "K".

Hsin-Chia Fu received his BS degree from National Chiao-Tung University (NCTU) in electrical and communication engineering, and his MS and PhD degrees from New Mexico State University, both in computer and electrical engineering, in 1972, 1975, and 1981, respectively. He is professor of computer science and information engineering at National Chiao-Tung University, Hsinchu, Taiwan, where he has been since 1983. Prior to his appointment at NCTU, he was a member of the technical staff at Bell Laboratories. From 1987 to 1988, he has served as the director of the Department of Information Management at the Research Development and Evaluation Commission, of the Executive Yuan, Republic of China. During the academic year 1988 to 1989, he was a visiting scholar at Princeton University. From 1989 to 1991, he served as the chairman of the Department of Computer Science and Information Engineering. From September to December of 1994, he was a visiting scholar at Fraunhofer-Institut for Production Systems and Design Technology, Berlin, Germany. He has served as the program cochair and general cochair of the International Symposium

on Artificial Neural Networks in 1993 and 1995, respectively. He has authored and coauthored more than 50 technical papers and two textbooks, *PC/XT BIOS Analysis* (1987) and *Introduction to Neural networks* (1991) by Sun-Kung Book Company, and Third Wave Publishing Company, respectively. His research interests include computer architecture, parallel computing, pattern recognitions, and neural networks. Dr. Fu is a member of SPIE, the IEEE Computer Society, Phi Tau Phi, and the Eta Kappa Nu Electrical Engineering Honor Society.

Ting-Shan Cheng: Biography and photograph not available.



Cheng-Chin Chiang received the BS degree in electrical engineering from the National Cheng-Kung University, Tainan, Taiwan, in 1986. He received his MS degree in computer engineering and PhD degree in computer science and information engineering from the National Chiao-Tung University, Hsinchu, Taiwan, in 1988 and 1993, respectively. In 1993, he performed postdoctoral research in the Department of Computer Science and Information Engineering, the National Chiao-Tung University, for six months. Since October 1993, he joined the Advanced Technology Center (ATC) at the Computer and Communication Research Laboratories, Industrial Technology Research Institute, Chutung, Hsinchu, Taiwan. He is now a researcher at ATC. His current research interests include intelligent document analysis and processing, optical handwritten Chinese character recognition, neural networks, and parallel processing.

Shing-Ming Roan: Biography and photograph not available.