An Alternate Smoothing and Stripping Algorithm for Thinning Digital Binary Patterns

Yat Keung Chu

A Major Technical Report

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montréal, Québec, Canada

March 1985

ABSTRACT

An Alternate Smoothing and Stripping Algorithm for Thinning Digital Binary Patterns

Yat Keung Chu

This report presents an algorithm for thinning digital binary patterns (skeletonization). The algorithm consists of two phases. The first phase thins the input pattern to unit thickness by alternately smoothing of the pattern and stripping of the contour points. The second phase adjusts the resulting skeleton from phase one so that a more medial skeleton can be obtained. The experimental performance of the algorithm is first compared with some recent results from two other research groups and then with nine other skeletonization algorithms that have been proposed by other researchers. Results show that our algorithm produces skeletons of best quality. The skeletons produced are smooth and with least distortion.

ACKNOWLEDGEMENTS

First, I would like to thank my supervisor, Dr. C.Y. Suen, for his help, suggestions, criticisms, and encouragements.

I also wish to thank Michael Yu and Helen Ma who supplied test patterns and programs.

Finally, I wish to thank the people in the Computer Centre of Concordia University for their support of the project.

TABLE OF CONTENTS

1. INTRODUCTION

Thinning is a process which transforms a binary image into a line skeleton of unit thickness. The idea was first developed about twenty five years ago [1]. Its applications are numerous [2-13], e.g. inspection of printed circuit boards [2]; counting of asbestos fibres on air filters [3]; analysis of chromosome shapes [4]; examination of soil cracking patterns [5]; classification of fingerprints [6]; recognition of optical characters [8,9,13], etc. The major functions of thinning in image processing and pattern recognition are:

(a) to reduce data storage

(b) to reduce transmission requirments

(c) to reduce or simplify the amount of data to be processed.

Over the last twenty years a large number of thinning algorithms have been developed [1,4,8,14,15]. There are three major problems experienced by these algorithms:

(a) maintaining connectedness

(b) retaining end-points

(c) ensuring points are removed symmetrically.

Most existing algorithms produce a skeleton by successive stripping of the edge points of the binary pattern. Different algorithms generate different skeletons. Due to the above mentioned problems, different degrees of distortion have also been produced. The goal of our work is to develop a thinning algorithm which minimizes distortion.

2. DEFINITIONS

(a) Digital binary picture:

A matrix where each pixel element has either value 1 (black) or 0 (white). Figure 1 shows a digital binary picture.

(b) Digital binary pattern:

The figure formed by all the black pixels in the digital binary picture.

(c) 3x3 window:

A, B, C, D, E, F, G, H and X together form a 3x3 window of the pixel X.

A	B	C
D	X	E
F	G	H

(d) 4-neighbour:

The neighbours B, D, E, and G are each called a 4-neighbour of the pixel X.

	B	
D	X	E
	G	

2. DEFINITIONS (Cont'd)

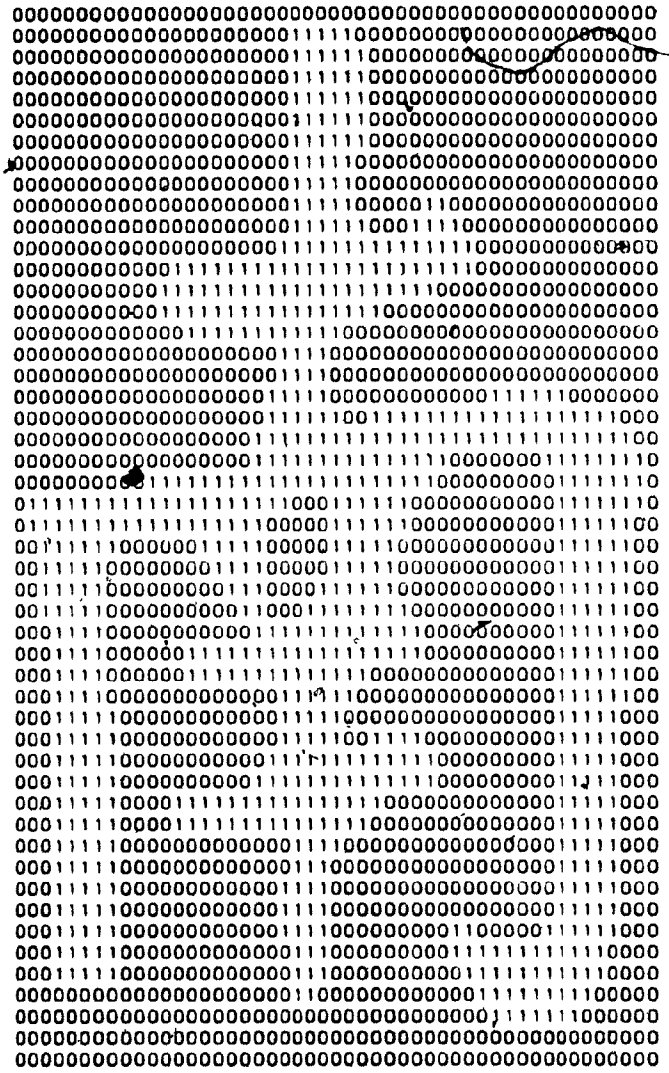


FIGURE 1. DIGITAL BINARY PATTERN

2. DEFINITIONS (Cont'd)

(e) 8-neighbour:

The neighbours A, B, C, D, E, F, G, and H are each called an 8-neighbour of the pixel X.

A	B	C
D	X	E
F	G	H

(f) Cross-point:

A pixel X is a cross-point if itself has a value of 1 and its 8-neighbours have exactly the same values as shown in the 3x3 window below.

0	1	0
1	X	1
0	1	0

(g) End-point:

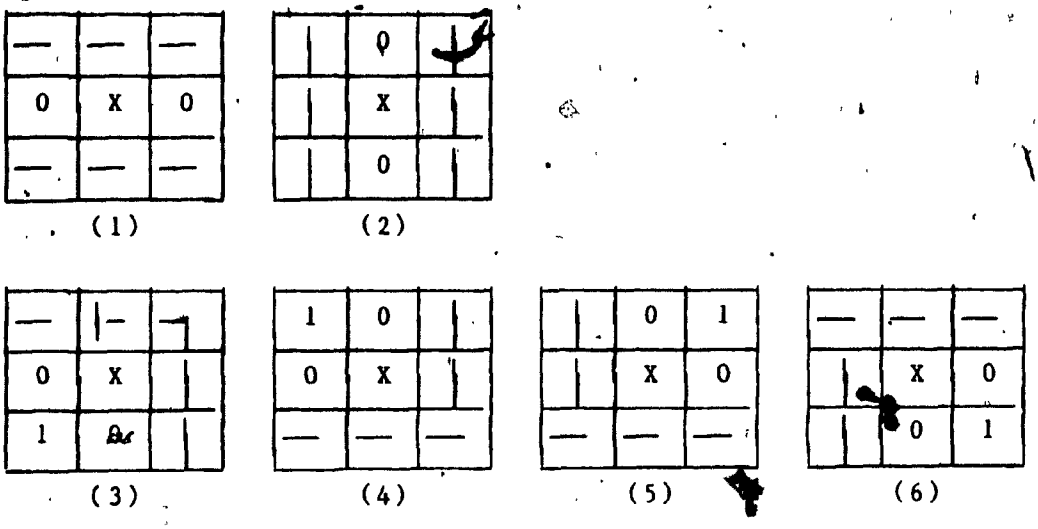
A pixel X is an end-point if itself has a value of 1 and has one and only one black 8-neighbour. For example:

0	0	0
0	X	0
0	0	1

(h) Break-point:

A pixel X is a break-point if itself has a value of 1 and its 3x3 window matches any of the following six window patterns.

2. DEFINITIONS (Cont'd)



Note: A dash line through a number of positions indicates that one or more of these positions contain black pixels.

3. THE THINNING ALGORITHM

The proposed thinning algorithm is accomplished in two main phases:

(1) Thin

(2) Adjust

Following gives the detailed description of each phase.

3. THE THINNING ALGORITHM

3.1 THIN

This phase thins the original binary pattern down to a skeleton figure of unit thickness. The Thin phase is accomplished by four processes. They are executed for several iterations until the pattern cannot be thinned any more. The four processes in order are:

(1) Fill

(2) Delete

(3) Label

(4) Strip

Following gives a detailed description of each of these four processes.

3. THE THINNING ALGORITHM

3.1.1 FILL

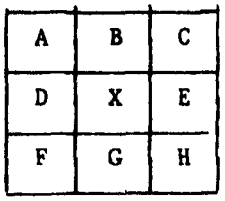
The purpose of this process is to smooth the binary pattern. It fills in holes in the pattern by changing the pixel X from 0 (white) to 1 (black) if it has more than 2 black 4-neighbours. For example:

3. THE THINNING ALGORITHM

3.1.2 DELETE

This process also smooths the binary pattern. It eliminates isolated black pixels and small bumps of the pattern. A pixel X is changed from 1 (black) to (white) 0 if the Boolean function (A+B+D) (E+G+H) + (B+C+E) (D+F+G) is true and X itself is not a break-point or an end-point.

For example:



3 x 3 WINDOW

3. THE THINNING ALGORITHM

3.1.3 LABEL

This process labels all contour points. There are two types of contour points. The first type is the 'corner-point' and the other 'trace-point'.

A black pixel X is a corner-point if it has seven black 8-neighbours of which four of them are 4-neighbours. For example:

0	1	1
1	x	1
1	1	1

A black pixel X is a trace-point if it has three or less black 4-nehghbours. For example:

0	0	1
1	X	1
1	1	1

Contour points are labelled from left to right, top to bottom. Figure 2 shows the contour points of a binary pattern.

Contour points are stripped in the Strip process if they satisfy certain conditions which will be explained in Section 3.1.4.

3.1.3 LABEL (Cont'd)

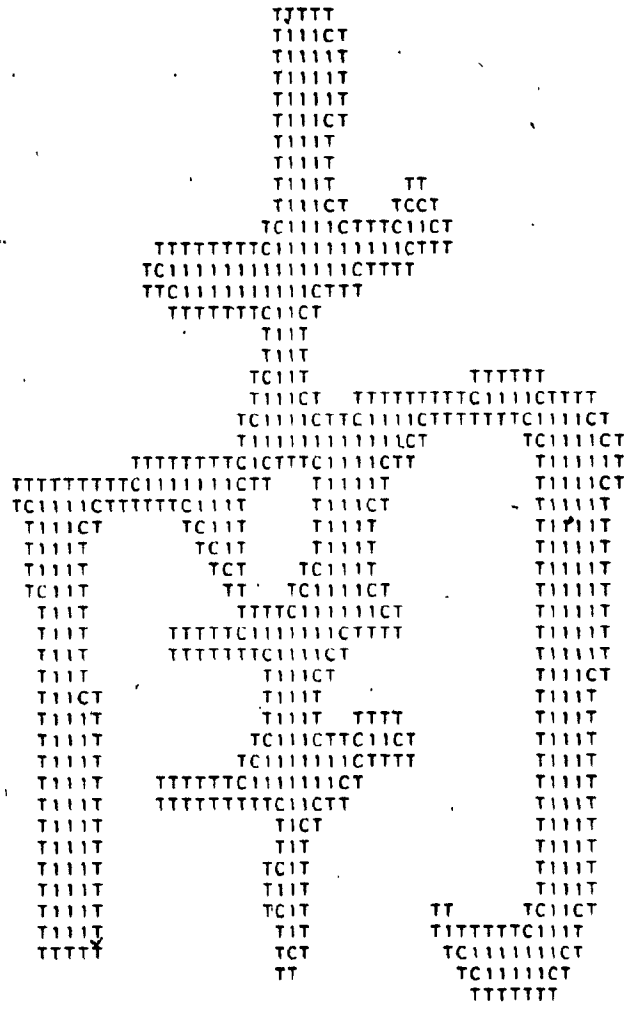


FIGURE 2. CONTOUR POINTS OF A BINARY PATTERN

Trace-points are identified by the letter "T" and corner-points the letter "C".

3. THE THINNING ALGORITHM

3.1.4 STRIP

This process removes contour points from the pattern in two stages. Trace-points are stripped in the first stage, and corner-points in the second stage.

In the first stage, trace-points are traced one by one and deleted right after if they are not end-points or break-points. To trace a contour, a 3x3 window is moved from left to right, top to bottom of the binary picture to search for the first trace-point X. The eight neighbours of X are then examined in a predefined 'Start Scanning Sequence' to look for the next trace-point. The 'Start Scanning Sequence' depends on the type of trace. There are four types of trace: (1) outer contour clockwise, (2) outer contour counter-clockwise, (3) inner contour clockwise, and (4) inner contour counter-clockwise traces. In the odd iterations of the Strip phase, clockwise is assumed for an outer contour trace, and counter-clockwise for an inner contour trace. On the other hand; in even iterations of the Strip phase, counter-clockwise is assumed for an outer contour trace and clockwise for an inner contour trace.

The purpose of alternate inner and outer contour tracing and stripping is to balance the stripping of points from both the inside and outside contours of those binary patterns which have closed loop(s) like the letter 'B'. The function of clockwise and counter-clockwise traces is to avoid stripping points in one specific order. These alternate operations help to strip points symmetrically from the pattern so that the resulting skeleton will not look biased.

3. THE THINNING ALGORITHM

3.1.4 STRIP (Cont'd)

The first contour to be traced is always an outer contour. However, the type of subsequent contours (inner or outer), if any, cannot be known when the first trace-point X is encountered. In the algorithm, an outer contour trace is first assumed. Table 1 lists the 'Start Scanning Sequence' for each type of trace. Once the next trace-point X has been located, its eight neighbours are then scanned in another sequence called 'Next Scanning Sequence'. The position of the next trace-point X relative to the previous 3x3 window determines the 'Next Scanning Sequence' (see Table 2).

For instance, if the position of the next trace-point X is in position H of the previous window and it is an outer contour clockwise trace, the scanning of its eight neighbours will be in the sequence of BCEHGFDA. If any black pixel which is not labelled with 'T' is encountered in the scanning process then the inner contour trace sequence will be used instead because the contour now traced is an inner contour. The steps of scanning and stripping are repeated until the contour is traced back to its starting position. The whole operation repeats until no more trace-points remain.

3. THE THINNING ALGORITHM

3.1.4 STRIP (Cont'd)

Type of Trace	Start Scanning Sequence
outer contour, clockwise	E H G F D A B C
outer contour, anti-clockwise	D F G H E C B A
inner contour, clockwise	G F D A B C E H
inner contour, anti-clockwise	G H E C B A D F

TABLE 1. START SCANNING SEQUENCE

3. THE THINNING ALGORITHM

3.1.4 STRIP (Cont'd)

Position Of X In Previous Window	Outer Contour		Inner Contour	
	Clockwise	Anti-Clockwise	Clockwise	Anti-Clockwise
A	G F D A B C E H	E C B A D F G H		
B	F D A B C E H G	H E C B A D F G		
C	D A B C E H G F	G H E C B A D F	Same as outer contour anti-clockwise	Same as outer contour clockwise
D	H G F D A B C E	C B A D F G H E		
E	A B C E H G F D	F G H E C B A D		
F	E H G F D A B C	B A D F G H E C		
G	C E H G F D A B	A D F G H E C B		
H	B C E H G F D A	D F G H E C B A		

TABLE 2. NEXT SCANNING SEQUENCE

3. THE THINNING ALGORITHM

3.1.4 STRIP (Cont'd)

After all the trace-points have been processed, corner-points are then inspected one by one. There are four types of corner-points as shown in Figure 3. Type 1 has A being white and the rest of the 8-neighbours black. Type 2 has C being white and the rest of the 8-neighbours black and so on. A corner-point will be stripped if it is not a break-point or an end-point and it is less than or equal to 120^{o}. A corner-point of type 1 is said to be less than or equal to 120^{o} if in its 5x5 window (Figure 4) both I and J are black or three of I,J,K,L are black. The angles for corner-points of types 2, 3 and 4 are evaluated in the similar way.

Usually, more pixels are deleted from the outer corners and bends than the inner ones. To balance the number of pixels deleted at corners and bends of the pattern, the corner points are checked for deletion. Figure 5 shows an example of this operation. Two more pixels will be deleted from the inner part of the corner K while corner-points on the outer part of the corner K will still remain after deletion of all 120^{o} corner-points. The reason why we have chosen 120^{o} instead of 90^{o} (both I and J in Figure 4 must be black) can easily be seen from the example given in Figure 5. No corner-points will be deleted if 90^{o} were chosen.

After all the corner-points are processed, the resulting picture is checked to see if there is any more point to be stripped in the next iteration. If the resulting picture contains any point which is not a break-point, an end-point, or a cross-point then the operations described in Sections 3.1.1 through 3.1.4 are repeated; otherwise, the Thin phase terminates.

3. THINNING ALGORITHM

3.1.4 STRIP (Cont'd)

Figure 6 illustrates the binary pattern of Figure 1 passing through the Fill, Delete, Label, and Strip processes of the Thin phase at different iterations. Figure 7 shows the binary pattern upon completion of the Thin phase.

3. THE THINNING ALGORITHM

3.1.4 STRIP (Cont'd)

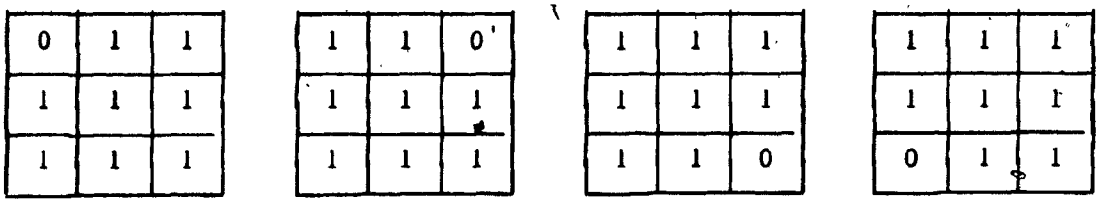
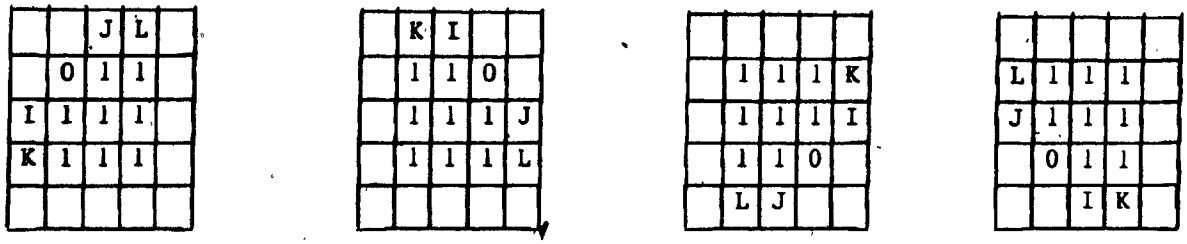


FIGURE 9. THE FOUR TYPES OF CORNER-POINTS

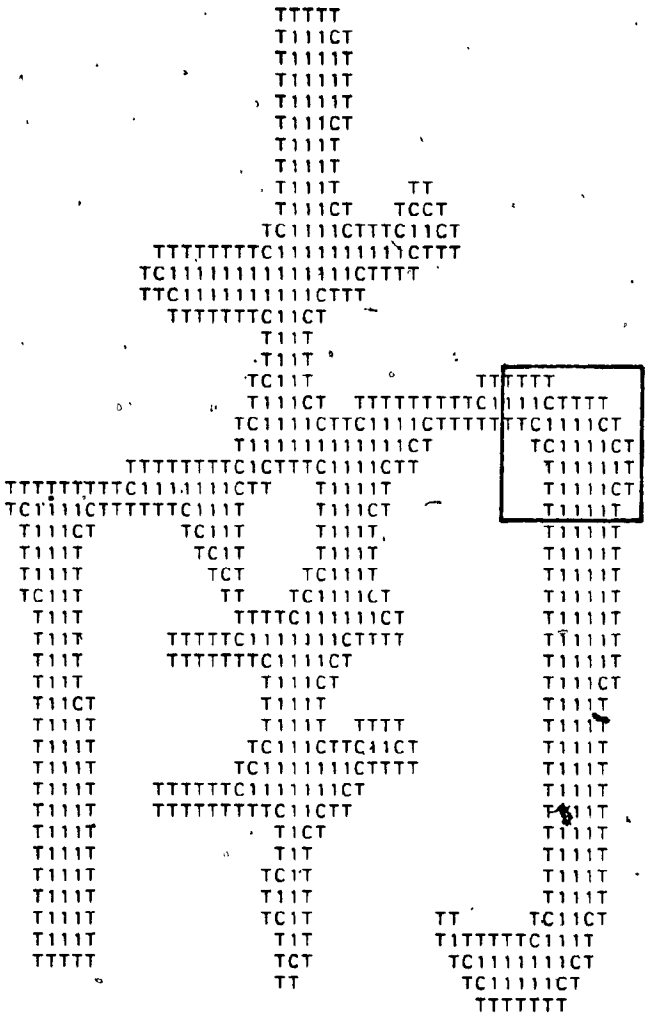
3. THE THINNING ALGORITHM

3.1.4 STRIP (Cont'd)



Type 1 Type 2 Type 3 Type 4

FIGURE 4. 5x5 WINDOWS USED TO EVALUATE ANGLES FOR THE 4 TYPES OF CORNER-POINTS



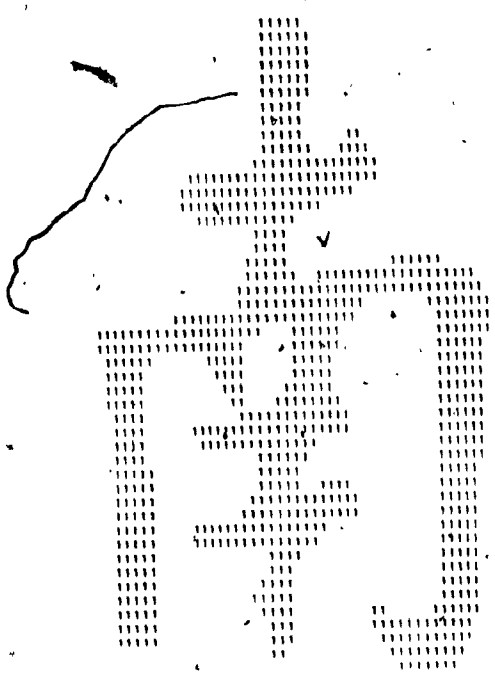
Corner

K

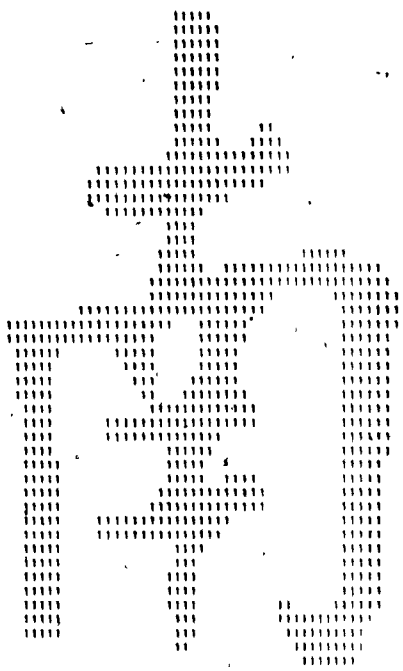
	NO. OF PIXELS DELETED	
	INNER CORNER	OUTER CORNER
DELETION OF CORNER-POINTS NOT IMPLEMENTED	6	13
DELETION OF 120° CORNER-POINTS IMPLEMENTED	8	13

FIGURE 5. EXAMPLE OF DELETING 120° CORNER-POINTS

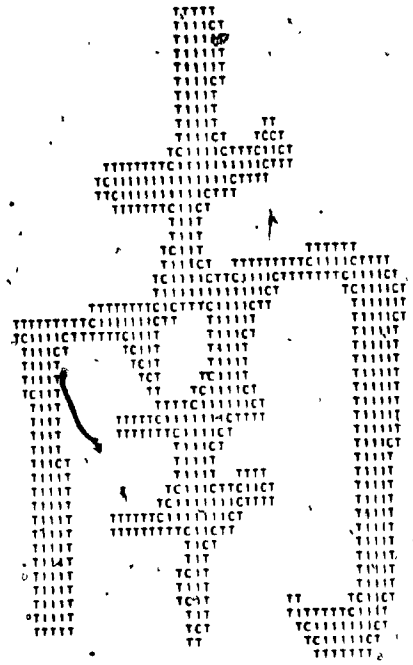
FILL



DELETE



LABEL



STRIP

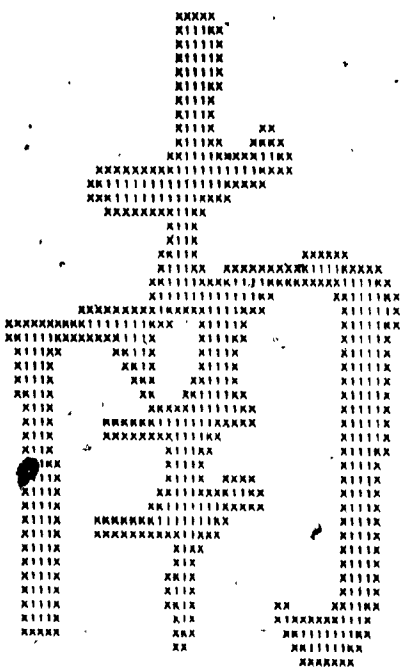
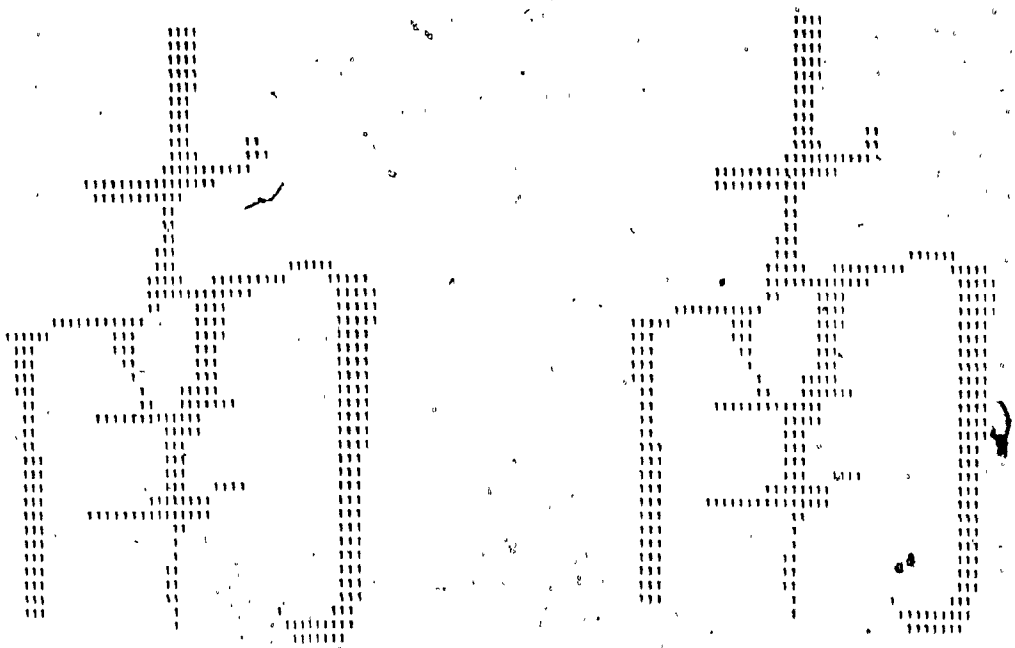


FIGURE 6. AN INPUT PATTERN PASSING THROUGH THE FILL, DELETE, LABEL, AND STRIP PROCESSES OF THE "THIN" PHASE

(X's are the points to be deleted, K's to be kept)

FILL

DELETE



LABEL

STRIP

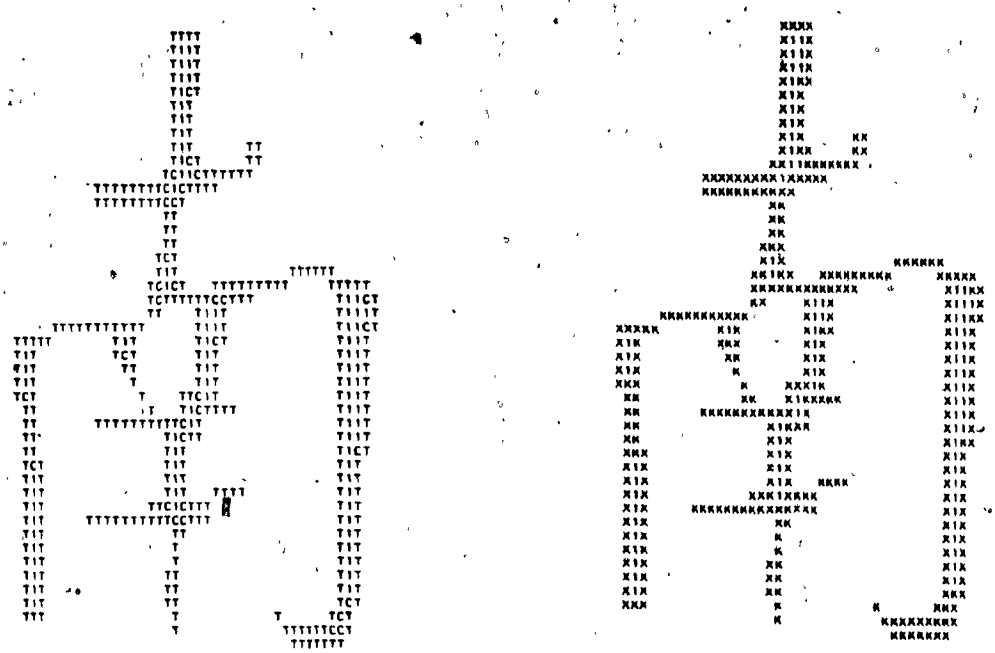
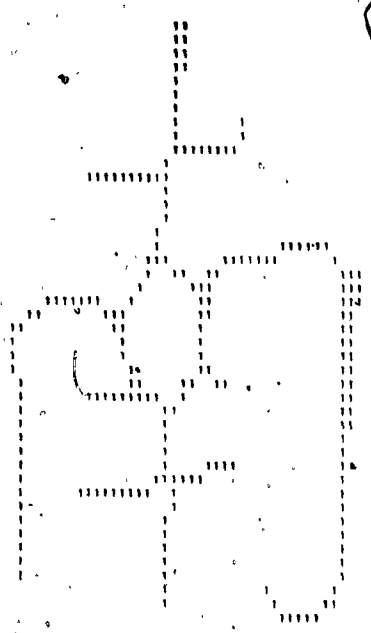
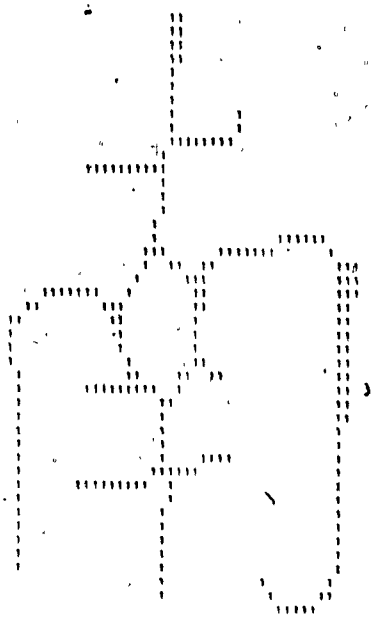


FIGURE 6. AN INPUT PATTERN PASSING THROUGH THE FILL, DELETE, LABEL, AND STRIP PROCESSES OF THE "THIN" PHASE (Cont'd)

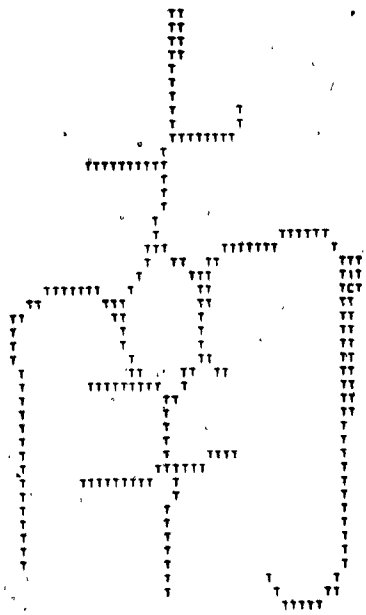
FILL



DELETE



LABEL



STRIP

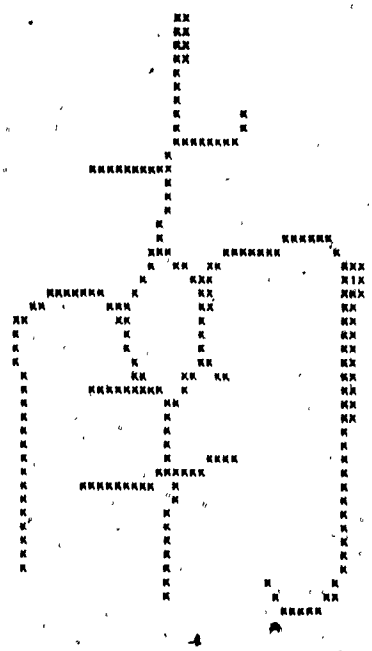


FIGURE 6. AN INPUT PATTERN PASSING THROUGH THE FILL, DELETE, LABEL, AND STRIP PROCESSES OF THE "THIN" PHASE (Cont'd)

3. THE THINNING ALGORITHM

3.1.4 STRIP (Cont'd)

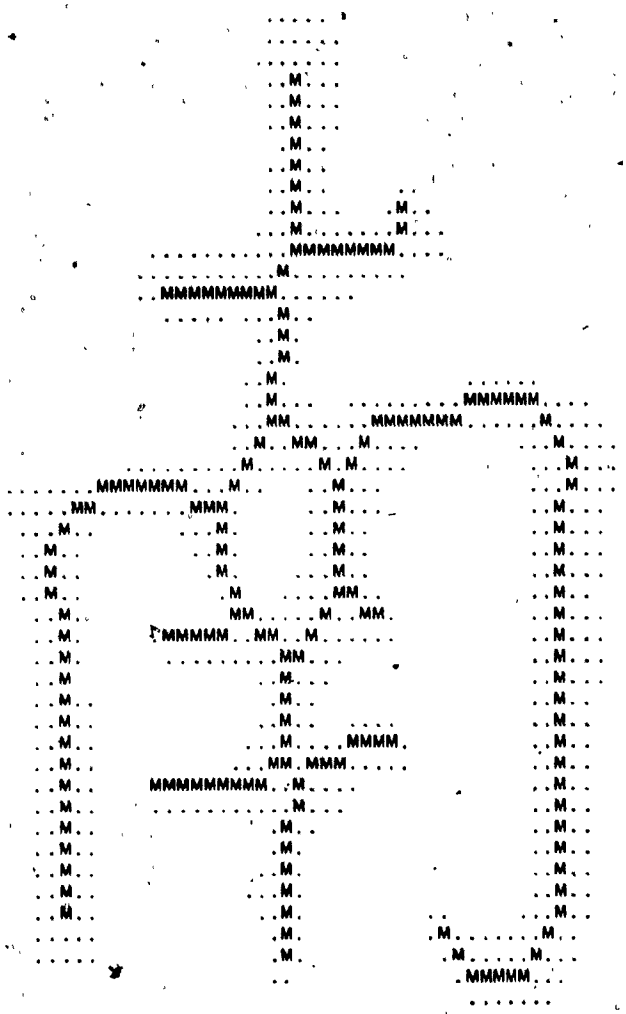


FIGURE 7. THE THINNED BINARY PATTERN

3. THE THINNING ALGORITHM

3.2 ADJUST

The skeleton obtained from phase one is already of good quality. However, there may be some parts of the skeleton that do not sit on the ideal medial line of the pattern. This Adjust phase tries to push the skeleton to the medial line.

There are two major reasons that cause the skeleton to be away from the medial line: (1) most of the contour points deleted during the Delete process lie mostly on the outer-contour, (2) if the input pattern has uneven line thickness, like the letter 'e' shown in Figure 8, some point between the thin line and the thick line becomes a break-point during the thinning process. This causes the skeleton to lean inwards as illustrated in Figure 9.

To deal with the above problem, we have implemented part of an adjustment scheme. A distance function is applied to the input pattern to calculate a value for each black pixel. The location of a point on the skeleton is adjusted if it satisfies the conditions described in Section 3.2.2.

3.2 ADJUST (Cont'd)

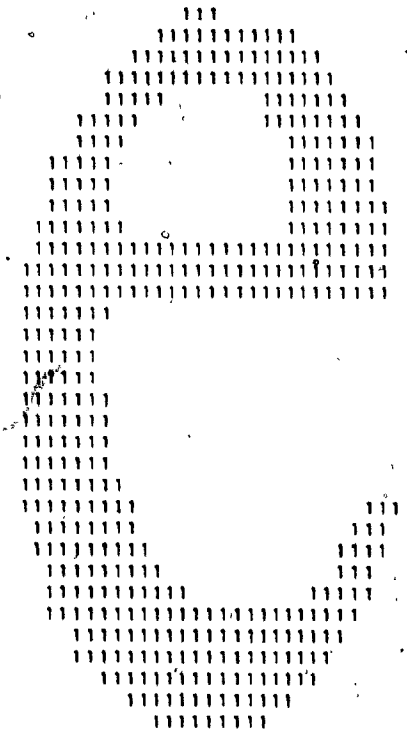


FIGURE 8. INPUT PATTERN WITH UNEVEN LINE THICKNESS

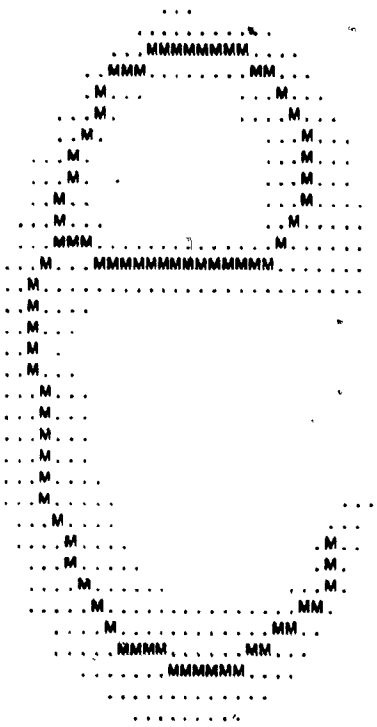


FIGURE 9. SKELETON RESULTED FROM UNEVEN LINE THICKNESS BEFORE THE "ADJUST" PHASE

3. THE THINNING ALGORITHM

3.2 ADJUST (Cont'd)

The Adjust phase is done in two steps:

(1) Propagate

(2) Move

Following is a detailed description of each step.

3. THE THINNING ALGORITHM

3.2.1 PROPAGATE

The purpose of this step is to find out how close to the medial line a pixel is. It assigns an integer value of the distance function to each black pixel in the binary picture. A pixel X with a distance function value higher than its neighbours means it is closer to the medial line. A black pixel is originally assigned a value of 1. Its windows (3x3, 5x5, 7x7, 9x9 etc.) are then examined one by one, from small to large, to see if they are enlargeable. A window is enlargeable if all the pixels in that window are black. The distance function value is incremented by 1 whenever a window is found enlargeable.

Figure 10 shows the distance function values of a binary pattern.

This Propagate step is used in [26] to mark the local maxima (defined as a pixel X whose distance function value is greater than or equal to all its 8-neighbours) of the distance function as points which must not be removed in the Slim phase of its algorithm. The Thin phase then produces a skeleton of unit thickness. We found by experiment and from the examples given in [26] that this algorithm often introduces noise spurs to the resulting skeletons. So we modified the use of the distance function values and apply them to adjust our skeletons after the Thin phase is complete.

3.2.1 PROPAGATE (Cont'd)

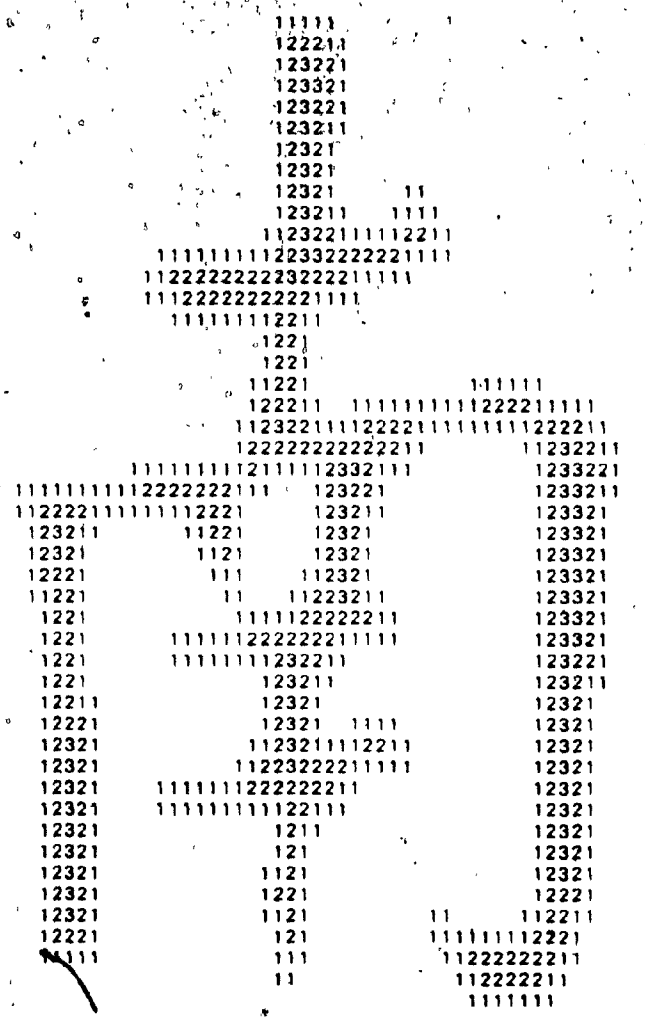


FIGURE 10. DISTANCE FUNCTION VALUES OF A BINARY PATTERN

3. THE THINNING ALGORITHM

3.2.2 MOVE

The skeleton obtained from the Thin phase is now denoted by 1's in the binary picture. The Move step looks at every point on the skeleton. In our case, we tried the left and right neighbours only to see how well this operation works. If the D and E neighbors of a skeleton point X are of value 0 (D and E could be part of the input binary pattern of value 1 previously) and if the distance function value of either D [f(D)] or E [f(E)] is greater than that of X [f(X)], then the 3x3 window of X is checked against all eleven windows shown in Figure 11. These 11 windows cover the most common cases found in binary patterns. If a match occurs, the value of the window is modified according to the directives in the corresponding entry and a local thinning on the resulting 3x3 window is performed. This step continues until no more points can be adjusted. Figure 12 shows the skeleton of binary pattern before and after the Adjust phase. The circles show the parts of the skeleton that have been adjusted. In most cases, some points on the skeleton resulted from the Thin phase are pushed to or closer to the medial line. However, we found that the Adjust phase is not necessary if the limbs of the input pattern are not thick in which case usually no pixels are adjusted.

3.2.2 MOVE (Cont'd)

A	B	C
D	X	E
F	G	H

3 x 3 WINDOW

WINDOW	DIRECTIVE IF f(D) > f(X)	DIRECTIVE IF f(E) > f(X)
1 ? ? 0 1 0 1 0 0	IF C = 1 DO NOTHING ELSE D = 1 X = 0	DO NOTHING
1 ? ? 0 1 0 ? 1 0	IF C = 1 DO NOTHING ELSE D = 1 X = 0	B = 1 E = 1 X = 0
1 ? ? 0 1 0 ? ? 1	IF C = 1 DO NOTHING ELSE D = 1 G = 1 X = 0	IF F = 1 DO NOTHING ELSE E = 1 B = 1 X = 0

FIGURE 11. WINDOWS AND DIRECTIVES IF MATCH OCCURS
(? means don't care)

3.2.2 MOVE (Cont'd)

WINDOW	DIRECTIVE IF f(D) > f(X)	DIRECTIVE IF f(E) > f(X)
0 1 ? 0 1 0 1 0 0	D = 1 X = 0	E = 1 G = 1 X = 0
0 1 ? 0 1 0 ? 1 ?	D = 1 X = 0	E = 1 X = 0
0 1 ? 0 1 0 0 0 1	D = 1 G = 1 X = 0	E = 1 X = 0
0 1 ? 0 1 0 0 0 0	D = 1 X = 0	E = 1 X = 0

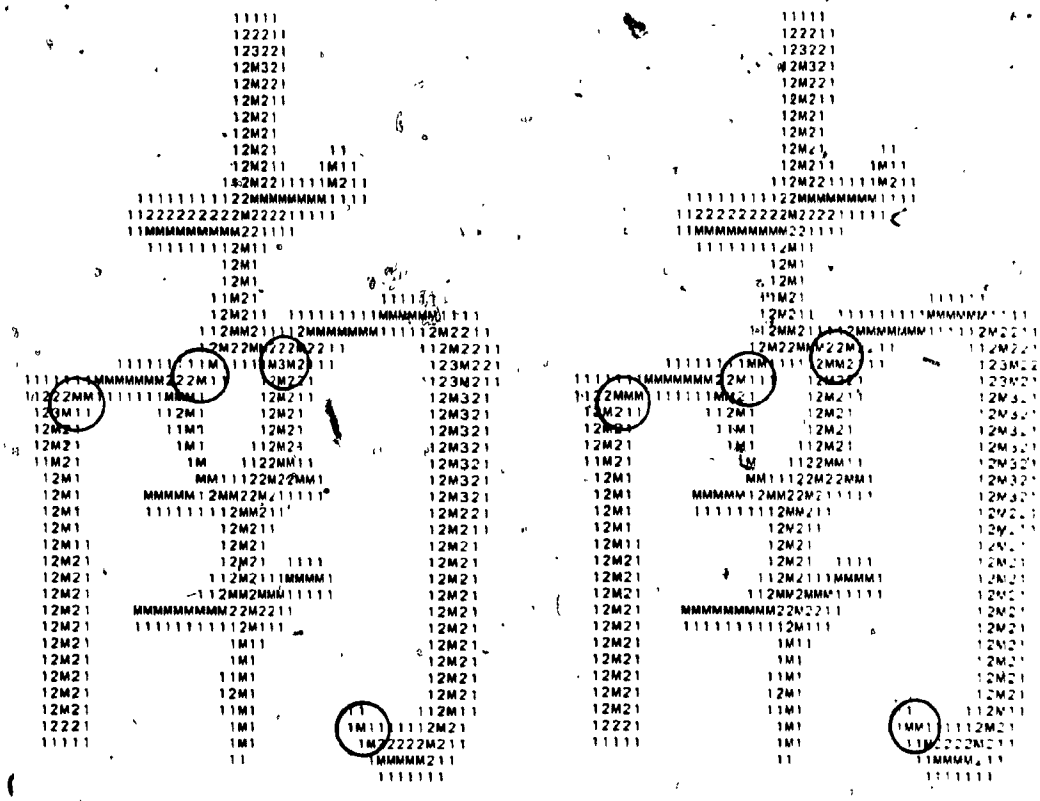
FIGURE 11. WINDOWS AND DIRECTIVES IF MATCH OCCURS (Cont'd)

3.2.2 MOVE (Cont'd)

WINDOW	DIRECTIVE IF f(D) > f(X)	DIRECTIVE IF f(E) > f(X)
0 0 1 0 1 0 1 ? ?	IF H = 1 DO NOTHING ELSE D = 1 B = 1 X = 0	IF A = 1 DO NOTHING ELSE E = 1 G = 1 X = 0
0 0 1 0 1 0 0 1 ?	B = 1 D = 1 X = 0	E = 1 X = 0
0 0 1 0 1 0 0 0 1	DO NOTHING	E = 1 X = 0
0 0 0 0 1 0 0 1 0	D = 1 X = 0	E = 1 X = 0

FIGURE 11. WINDOWS AND DIRECTIVES IF MATCH OCCURS (Cont'd)

3.2.2 MOVE (Cont'd)



BEFORE AFTER

FIGURE 12. SKELETON BEFORE AND AFTER THE "ADJUST" PHASE

4. EXPERIMENTAL RESULTS

The results of our algorithm are first compared with two recent results obtained by other research groups [23, 24, 25] and then with nine other algorithms. A complete set of skeletons produced by our algorithm is provided in the Appendix.

4. EXPERIMENTAL RESULTS

4.1 COMPARISON WITH RECENT RESULTS FROM OTHER RESEARCH GROUPS

The left half of Figure 13 shows the input and output patterns found in references [23], [24] and [25]. The skeletons produced in [23] are labelled ZF-1 through ZF-4 while those of [24] and [25] are identified by NS-1 and NS-2. Since the authors of [23], [24] and [25] did not put the skeletons produced by their algorithms on top of the original input patterns, it is difficult to see how good their outputs are. A careful look shows that certain parts of ZF-4 and NS-1 are significantly distorted. On the right half of the figure we show the skeletons obtained from our algorithm overlaid on top of the input patterns. The skeletons are marked from C-1 to C-6. As can be seen from the figure, C-2 is better than ZF-2 because the noise at the bottom left corner of 'E' has been eliminated; also C-4 is better than ZF-4 because our resulting '人' part of the Chinese ideograph appears to be less distorted than the one shown in ZF-4. Both NS-1 and C-5 suffer from the same weakness of deleting more points on outer corner than inner corner, but our partly implemented adjustment scheme produces a smoother skeleton than NS-1. Also the skeleton shown in C-6 resembles the original pattern more than that of NS-2. In general, the skeletons generated by our algorithm look smoother and closer to the medial line than the others.

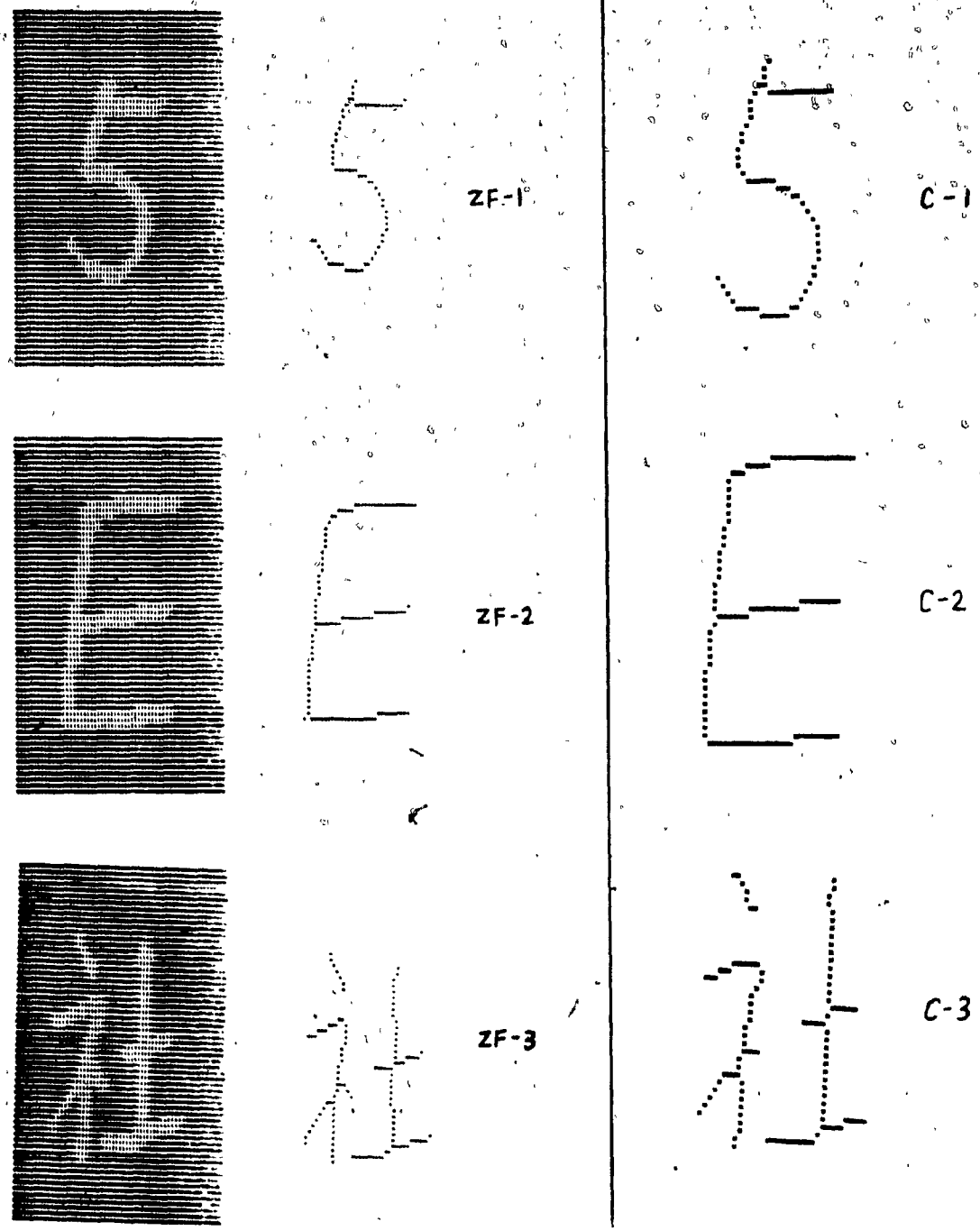


FIGURE 13. SKELETONS PRODUCED BY [23], [24], [25] AND ALTERNATE SMOOTHING AND THINNING

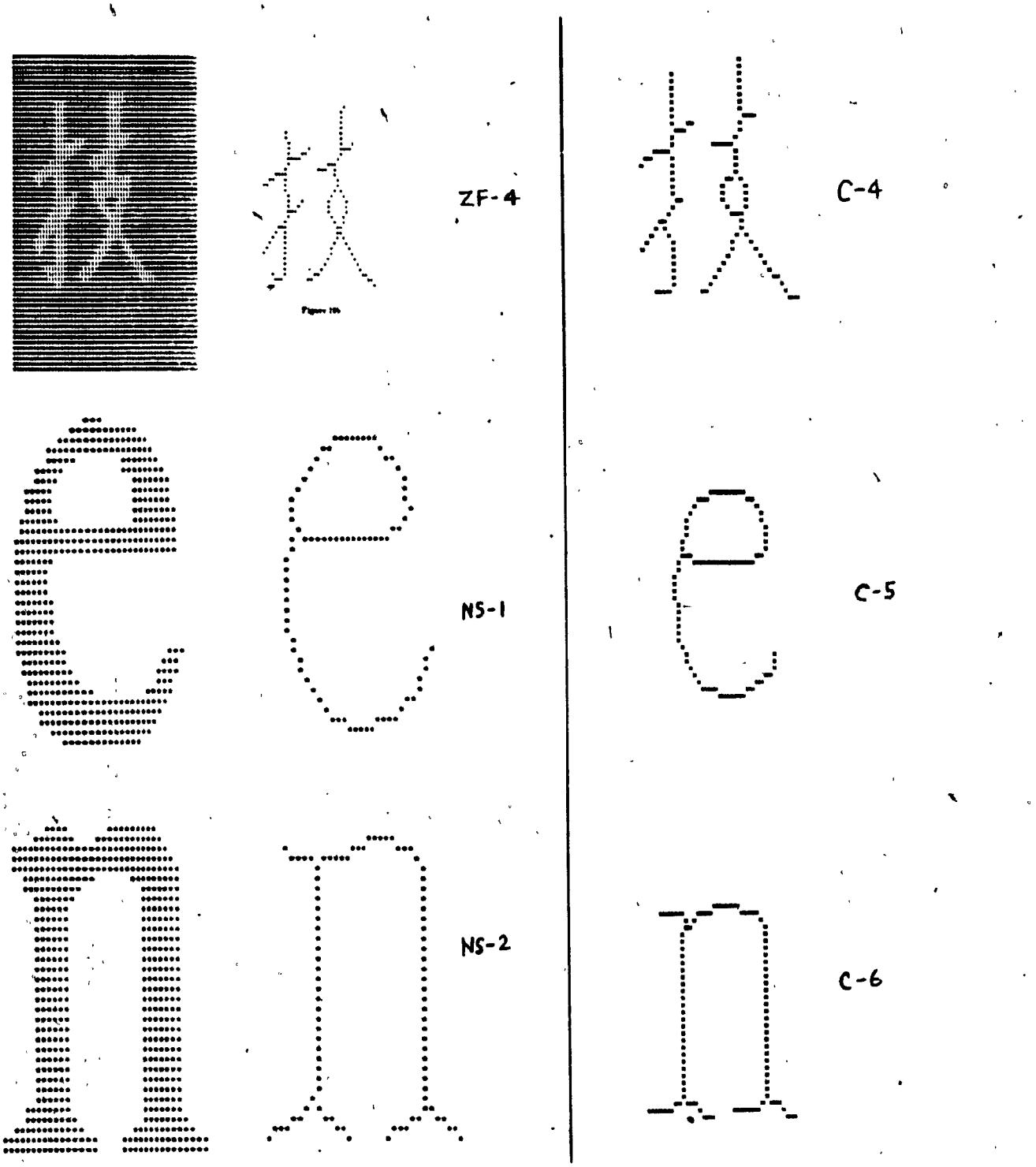


FIGURE 13. SKELETONS PRODUCED BY [23], [24], [25] AND ALTERNATE SMOOTHING AND THINNING (Cont'd)

4. EXPERIMENTAL RESULTS

4.2 COMPARISON WITH NINE OTHER ALGORITHMS

The results of our algorithm are compared with nine other algorithms presented in Ma's report [22]. These algorithms were chosen because the majority of them have been quoted widely in the literature. They are authored by:

1. E.S. Deutsch [16]
2. C.J. Hilditch [4]
3. R. Stefanelli and A. Rosenfeld [17]
4. D. Rutovitz [14]
5. H. Tamura [18]
6. S. Tsuruoka [19]
7. T.Y. Zhang and C.Y. Suen [20]
8. C. Arcelli [21]
9. H. Ma [22]

As Ma's method is not readily available, the following gives a summary of her algorithm.

TERMINOLOGY:

Contour tracing -- find the leftmost black point(p). A scanning spot moves point by point, from bottom to top of the leftmost column and successively repeats the procedure on the column immediately to the right of the column previously scanned until a black point(p) is found.

Two preprocessing steps were done before the thinning process. One of them eliminates all isolated points which are black elements with no neighbour. The other step fills a gap or eliminates a notch on an edge.

4. EXPERIMENTAL RESULTS

4.2 COMPARISON WITH NINE OTHER ALGORITHMS (Cont'd)

DESCRIPTION OF ALGORITHM:

The thinning operation traces the contour of the image. It scans the image horizontally by passing a 3 x 3 window over each contour point and performs the tests described below.

[I] P will be retained if it satisfies any of the following conditions:-

1. BREAK POINT TEST

A point of interest is considered as a break point if there exists one or more black points in the positions covered with symbol "X" in a 3 by 3 window and match one of the following patterns:

(A)

X	X	X
0	P	0
X	X	X

(B)

X	X	X
0	P	X
1	0	X

(C)

1	0	X
P	P	X
X	X	X

(D)

X	0	X
X	P	X
X	0	X

(E)

X	0	1
X	P	0
X	X	X

(F)

X	X	X
X	P	0
X	0	1

2. LOOP POINT TEST

A point of interest is considered as a loop point if there exists a 3 by 3 window that matches one of the following patterns with symbol "X" representing a don't care point.

(a)

X	1	X
1	P	1
X	1	X

(b)

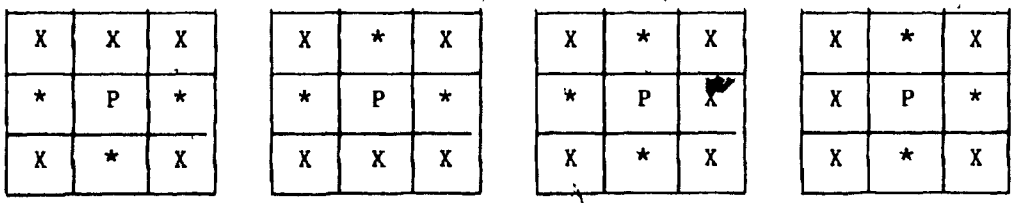
1	X	1
X	P	X
1	X	1

4. EXPERIMENTAL RESULTS

4.2 COMPARISON WITH NINE OTHER ALGORITHMS (Cont'd)

3. CORNER POINT TEST

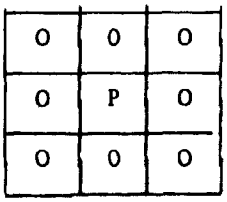
A point of interest is considered as a corner point if there exist's a 3 by 3 window that matches one of the following patterns with symbol "X" representing a don't care point and symbol "*" representing a contour point.



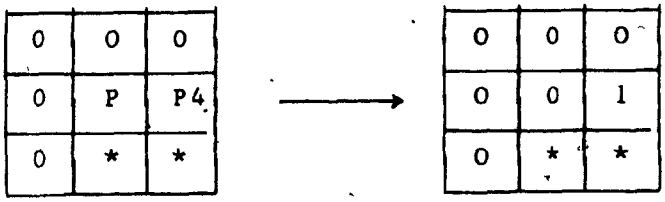
[II] P will be removed if it satisfies at least one of the following conditions:-

1. ISOLATED POINT

A point of interest is considered as an isolated point if all 8 neighbours in a 3 by 3 window are zeros.



2. EXCESSIVE EROSION TEST



P will be removed and point P4 will be changed to a black point.

4. EXPERIMENTAL RESULTS

4.2 COMPARISON WITH NINE OTHER ALGORITHMS (Cont'd)

3. END POINT TEST

If the sum of the eight neighbours is 1, call this point B, and the 8 neighbours of point B will be examined. If the sum of all neighbour points of point B is equal to 2 then it will not be removed.

4. TESTS IN I.1 to I.3 ARE NOT SATISFIED

This process of contour tracing and subsequent thinning are repeated until the pattern is nowhere more than one unit wide.

Each algorithm was tested on a data set of 183 binary pictures. The data set contains different types of patterns such as English alphabets, numerals, Chinese ideographs, line-like particles and chromosome etc. as shown in the Appendix. To give an intuitive feeling on the types of skeletons produced by the different algorithms, some illustrative examples are presented in Figure 14. All skeletons are printed on top of the input patterns for ease of comparison. Table 3 provides a summary of the computation requirement of these algorithms.

Tsuruoka's algorithm proved to use the least CPU time while Stefanelli and Rosenfeld's used the most. Arcelli's method required the least memory space while ours required the most. However, the skeletons obtained from Tsuruoka's and Arcelli's algorithm always contain noise spurs. The skeletons generated by Stefanelli and Rosenfeld's algorithm do not contain much noise but some of the skeletons are not connected. Zhang and Suen's algorithm and Ma's also produced noise spurs. Tamura's method and Deutsch's do not have noise spurs but they yielded excessive erosions. The remaining

4.2 COMPARISON WITH NINE OTHER ALGORITHMS (Cont'd)

algorithms produced good skeletons but they are usually not as close to the medial line as ours. Moreover, Rutovitz's method is slower than our algorithm.

4. EXPERIMENTAL RESULTS (Cont'd)

Notations for Figure 14:

CA - C. Arcelli

YC - Y.K. Chu

ED - E.S. Deutsch

CH - C.J. Hilditch

MY - H. Ma

SR - R. Stefanelli and A. Rosenfeld

DR - D. Rutovitz

HT - H. Tamura

ST - S. Tsuruoka

ZS - T.Y. Zhang and C.Y. Suen

4. EXPERIMENTAL RESULTS (Cont'd)

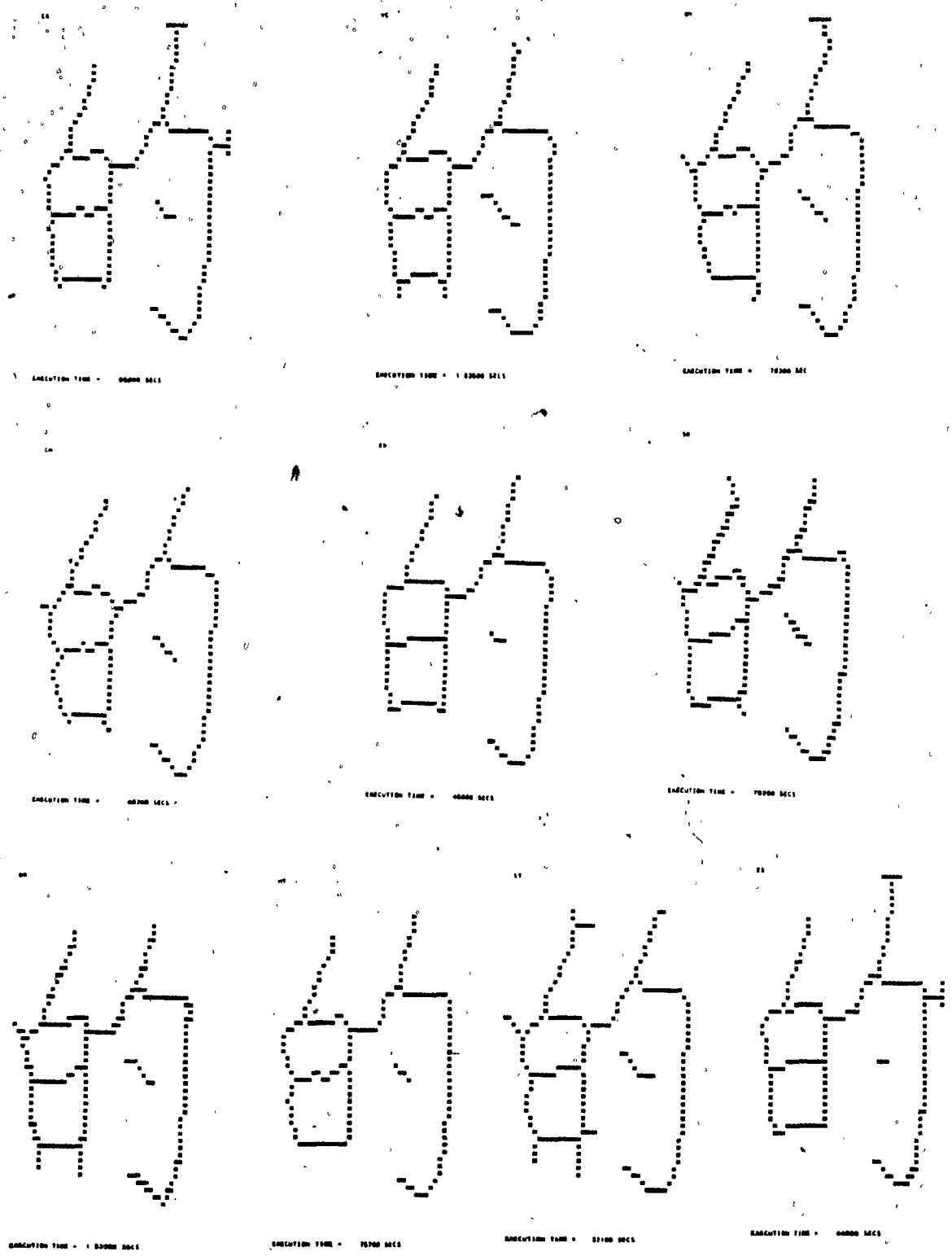


FIGURE 14. SKELETONS PRODUCED BY DIFFERENT ALGORITHMS

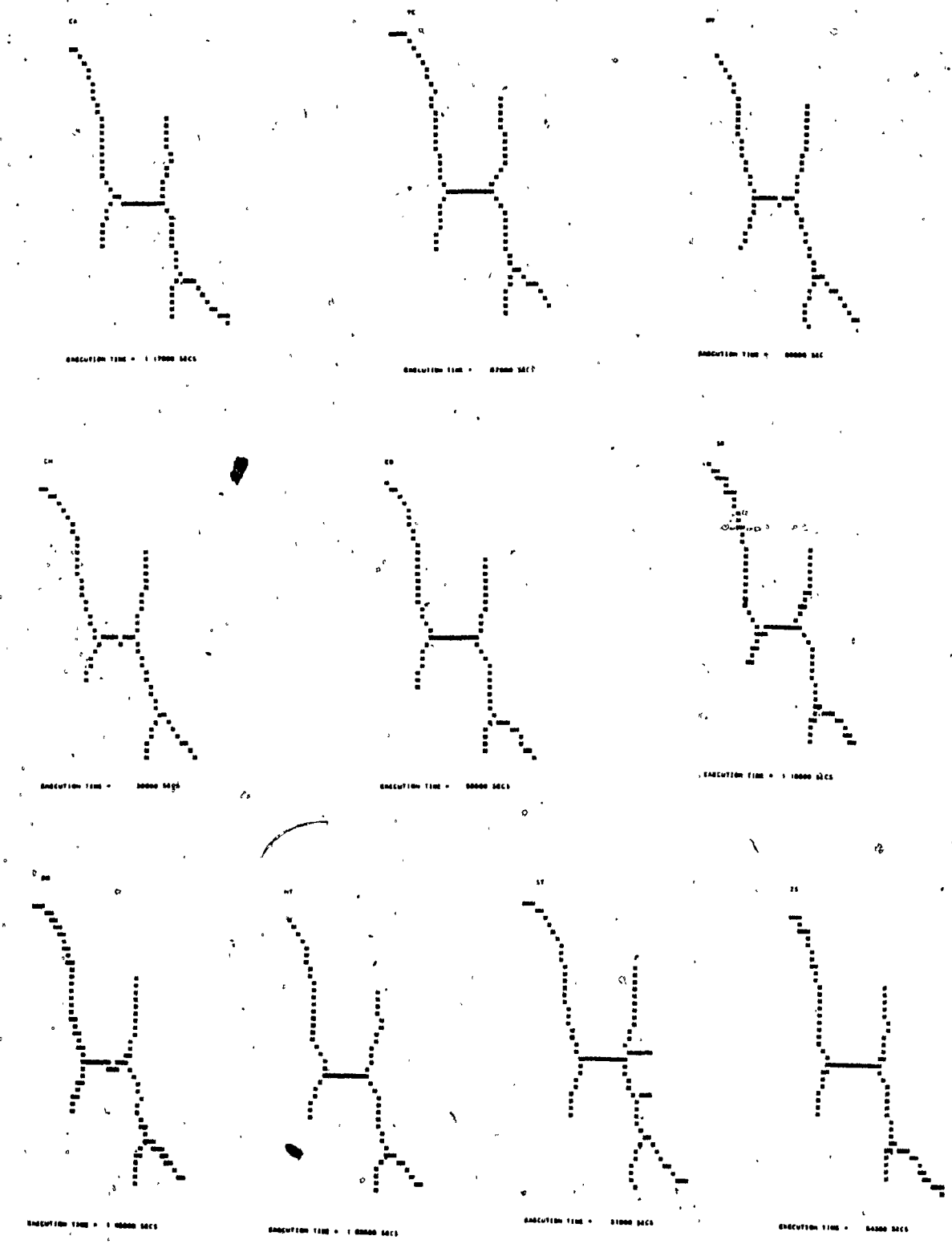


FIGURE 14. SKELETONS PRODUCED BY DIFFERENT ALGORITHMS (Cont'd)

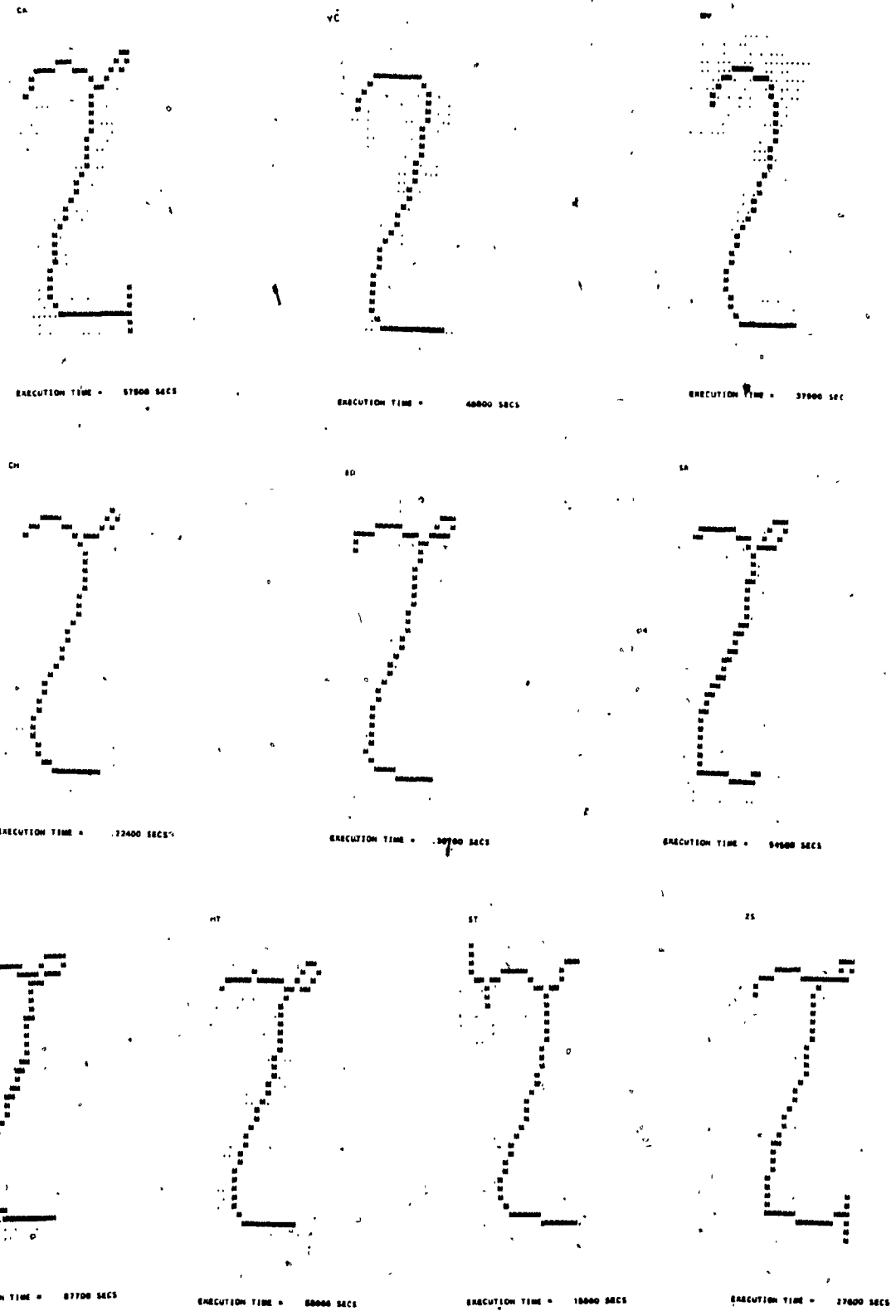


FIGURE 14. SKELETONS PRODUCED BY DIFFERENT ALGORITHMS (Cont'd)

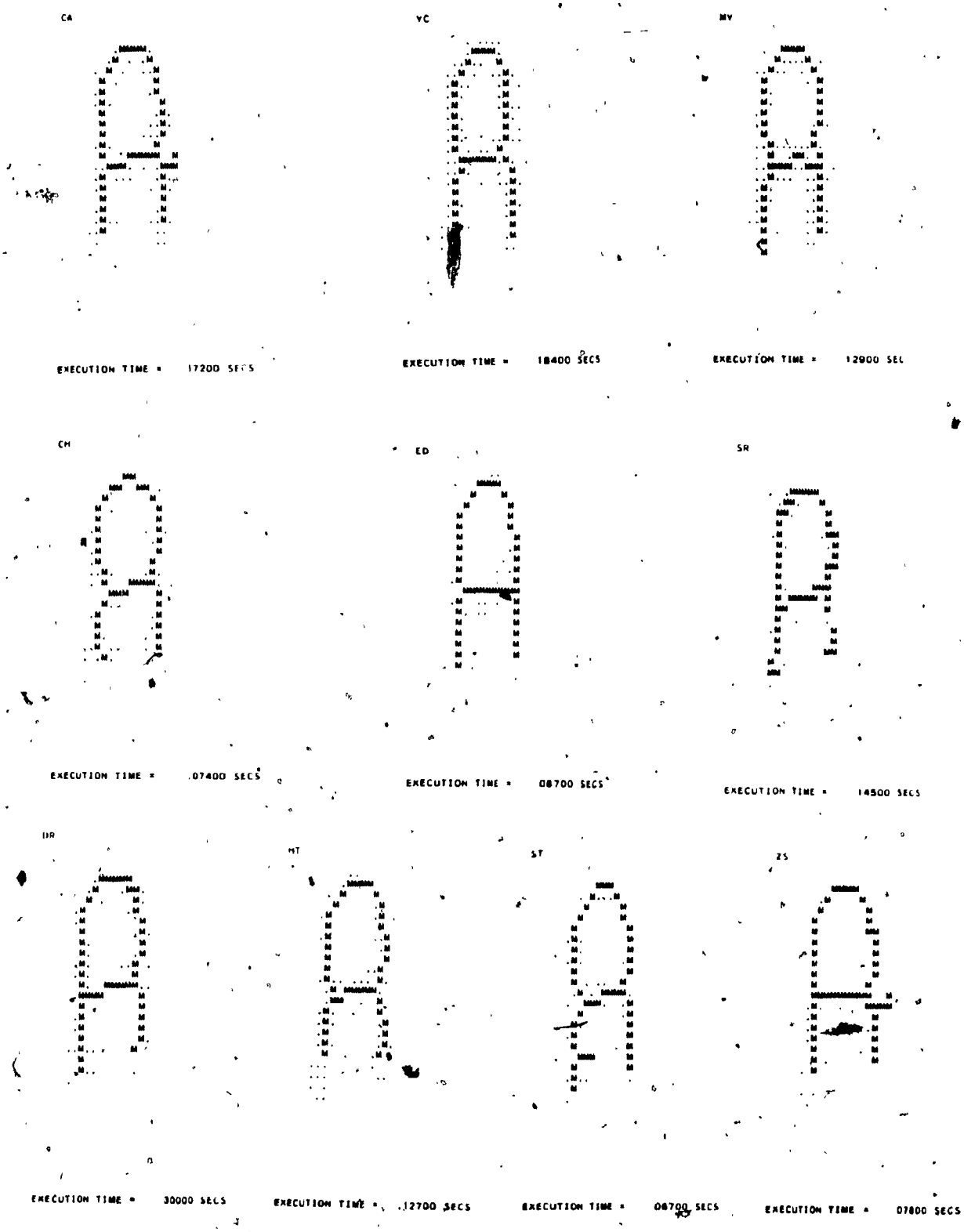


FIGURE 14. SKELETONS PRODUCED BY DIFFERENT ALGORITHMS (Cont'd)

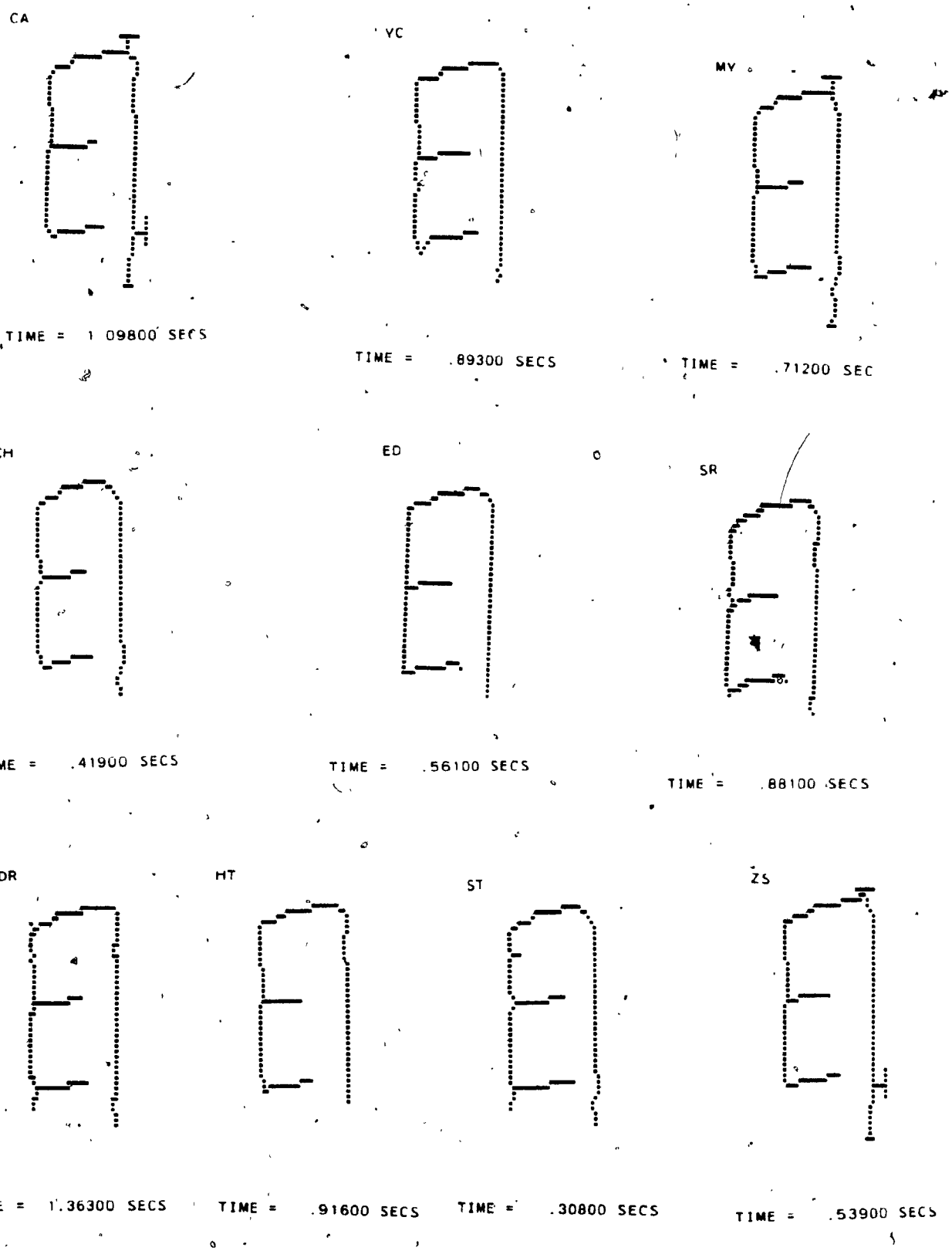


FIGURE 14. SKELETONS PRODUCED BY DIFFERENT ALGORITHMS (Cont'd)

4. EXPERIMENTAL RESULTS (Cont'd)

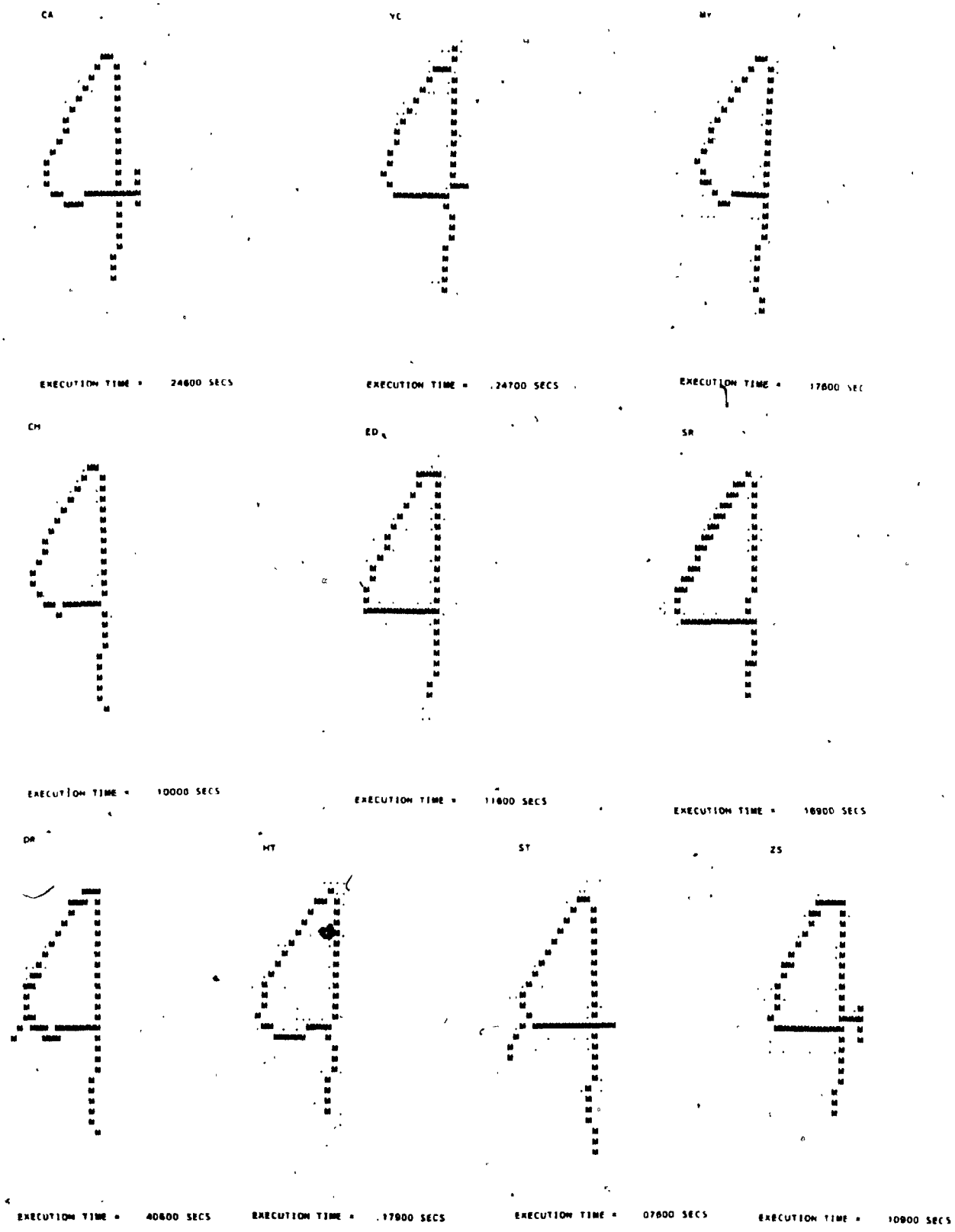


FIGURE 14. SKELETONS PRODUCED BY DIFFERENT ALGORITHMS (Cont'd)

TABLE 3

COMPUTATION REQUIREMENT

ALGORITHM	CPU TIME (SEC.) *	MEMORY (WORDS) **
Arcelli	84.209	26,048
Chu	84.060	50,564
Deutsch	38.734	26,240
Hilditch	32.246	26,496
Ma	67.261	26,752
Stefanelli and Rosenfeld	340.286	29,696
Rutovitz	123.851	28,096
Tamura	58.046	26,688
Tsuruoka	30.711	26,624
Zhang and Suen	38.220	26,176

* CDC CYBER/835 FORTRAN, the CPU time refers to the total amount of time in seconds required by the computer to process all 183 patterns.

** Memory requirement includes the two 2-dimensional arrays of 2 x (110 x 80) = 17600 words to store the original pattern and the thinned pattern, the program itself, and the buffers for the operations.

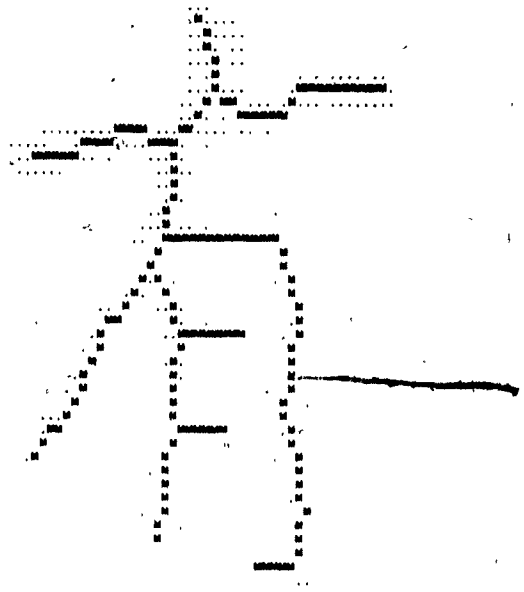
5. CONCLUDING REMARKS

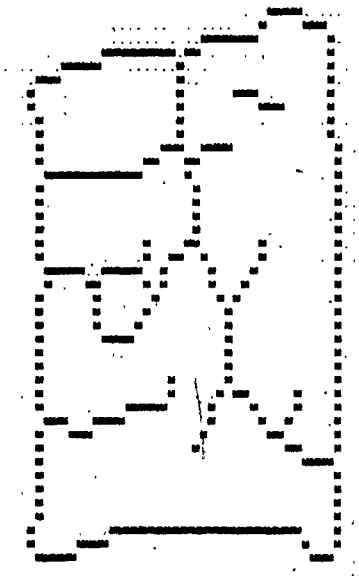
An algorithm which incorporates alternate smoothing, stripping and pixel adjustment for thinning digital binary patterns has been developed and implemented. A large set of binary patterns has been used to test the algorithm. The results are compared with numerous other algorithms. In general, our algorithm produces better and smoother skeletons. The algorithm shows good resistance to noise and produces skeletons which bear close resemblance to the original patterns. We note that the implementation of a partial ADJUST operation improves the skeletons, we expect still better skeletons can be achieved once the Adjustment program is fully implemented in other directions. Although our algorithm is not the slowest, it does use a lot of CPU time. This is mainly due to the complexity of the program and the smoothing process applied at each iteration. Our algorithm also uses a considerable amount of memory. The major reasons are twofold: a) an extra matrix is required to store the distance function values and b) the program is more complex than the others. However, with the capacity of hardware improving and the price dropping everyday, we think we have proposed a practical and viable algorithm for thinning digital binary patterns which can be implemented on microcomputers such as the IBM PC AT/XT, and others.

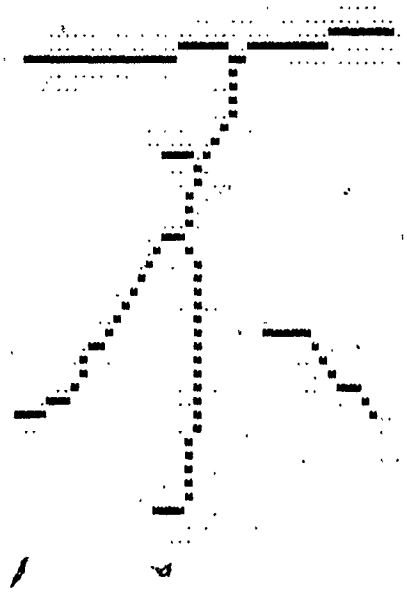
APPENDIX

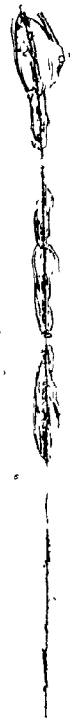
的一

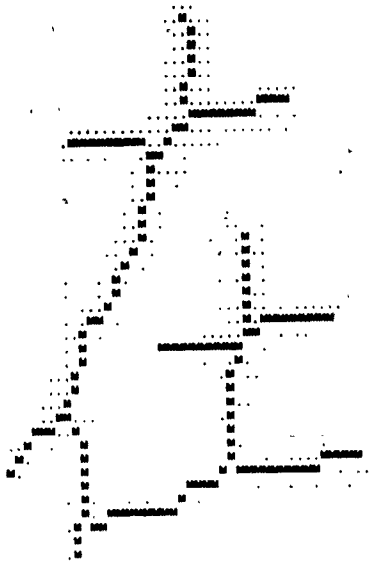
是我

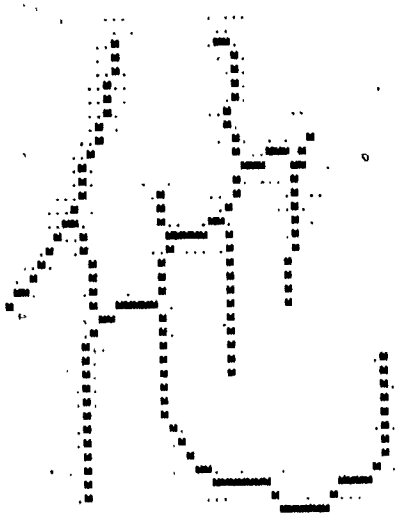


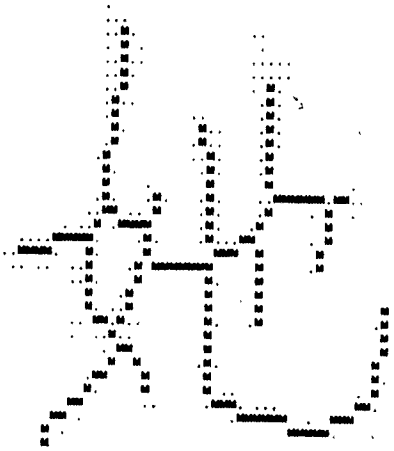


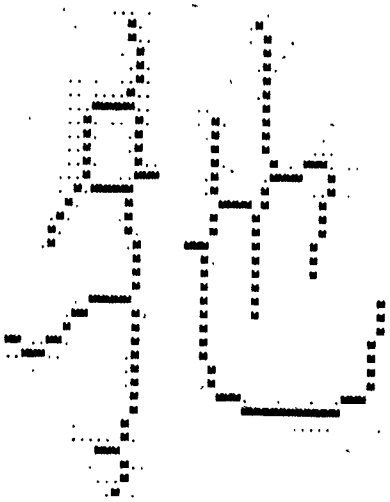


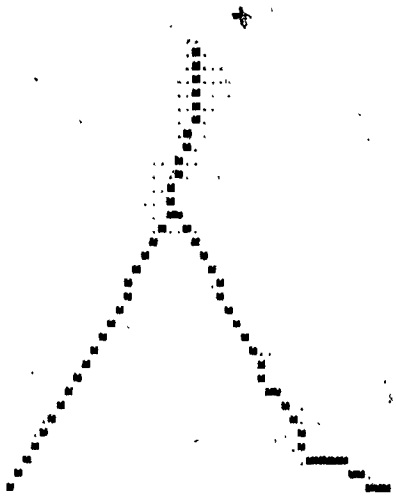


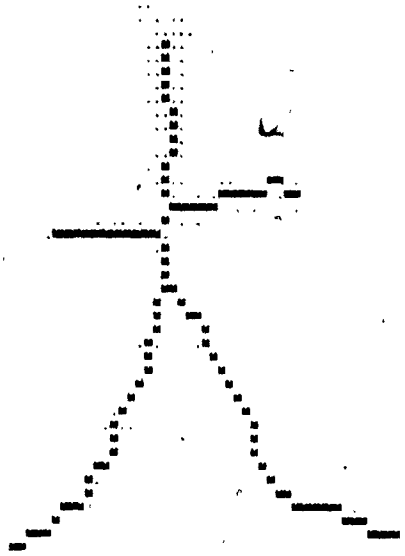


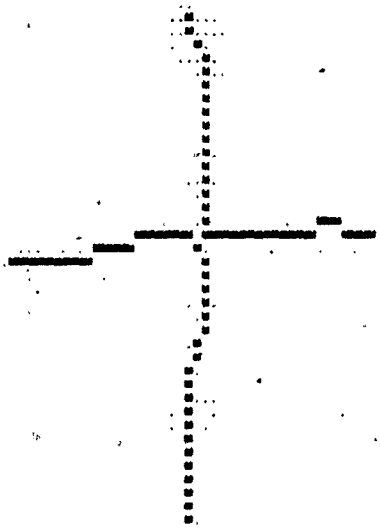


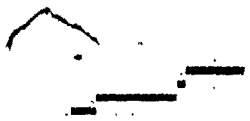


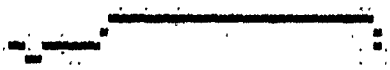


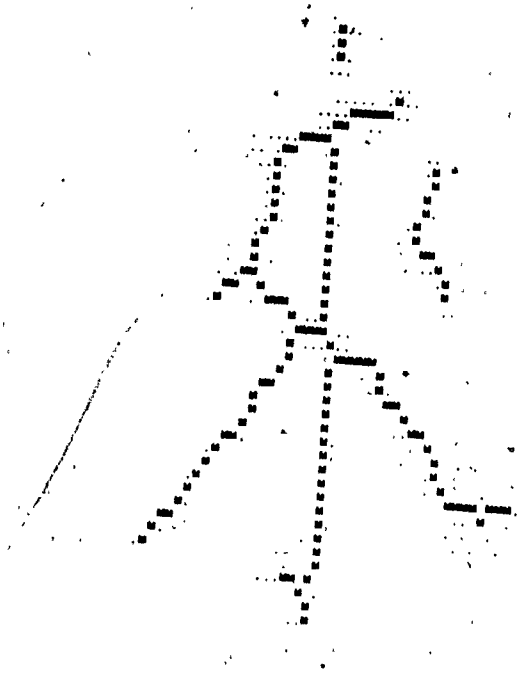


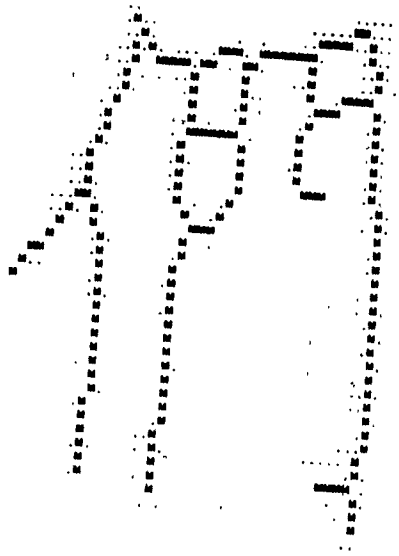


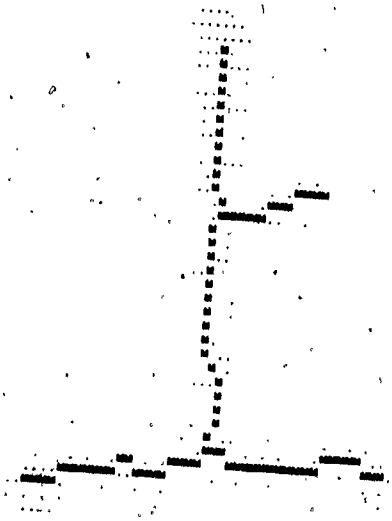


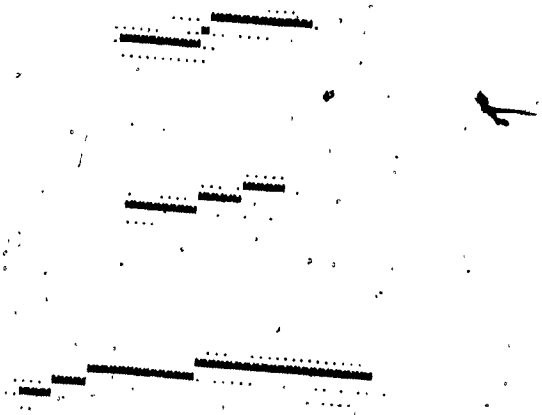


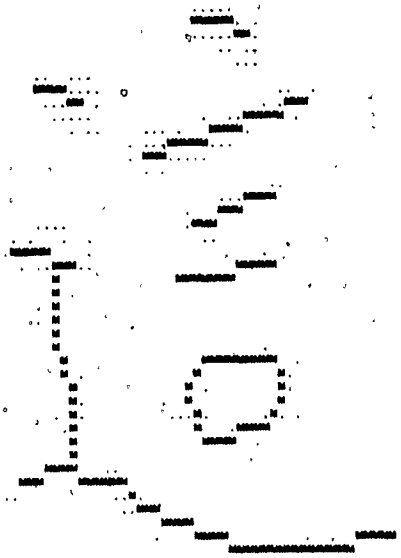


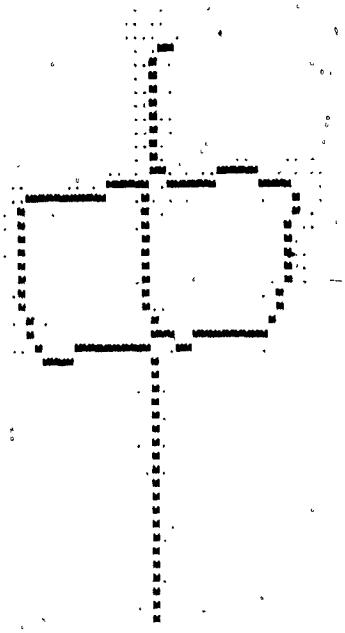


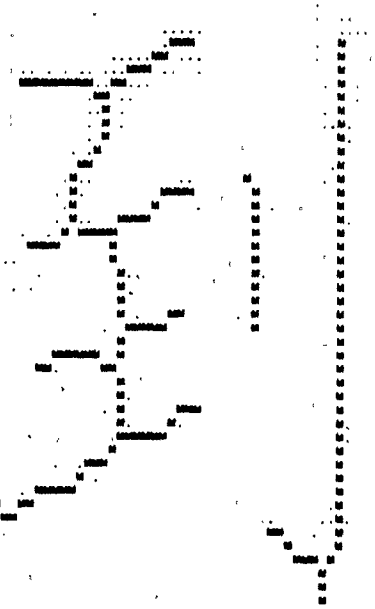


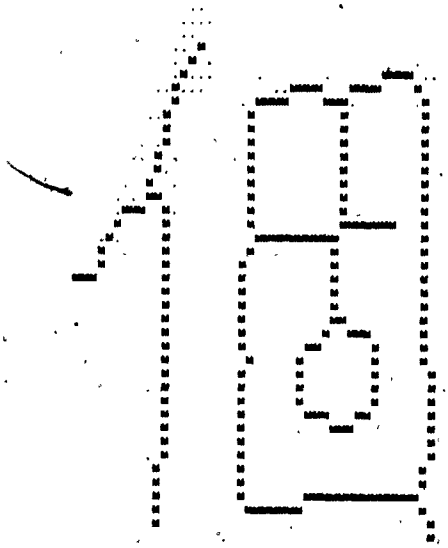


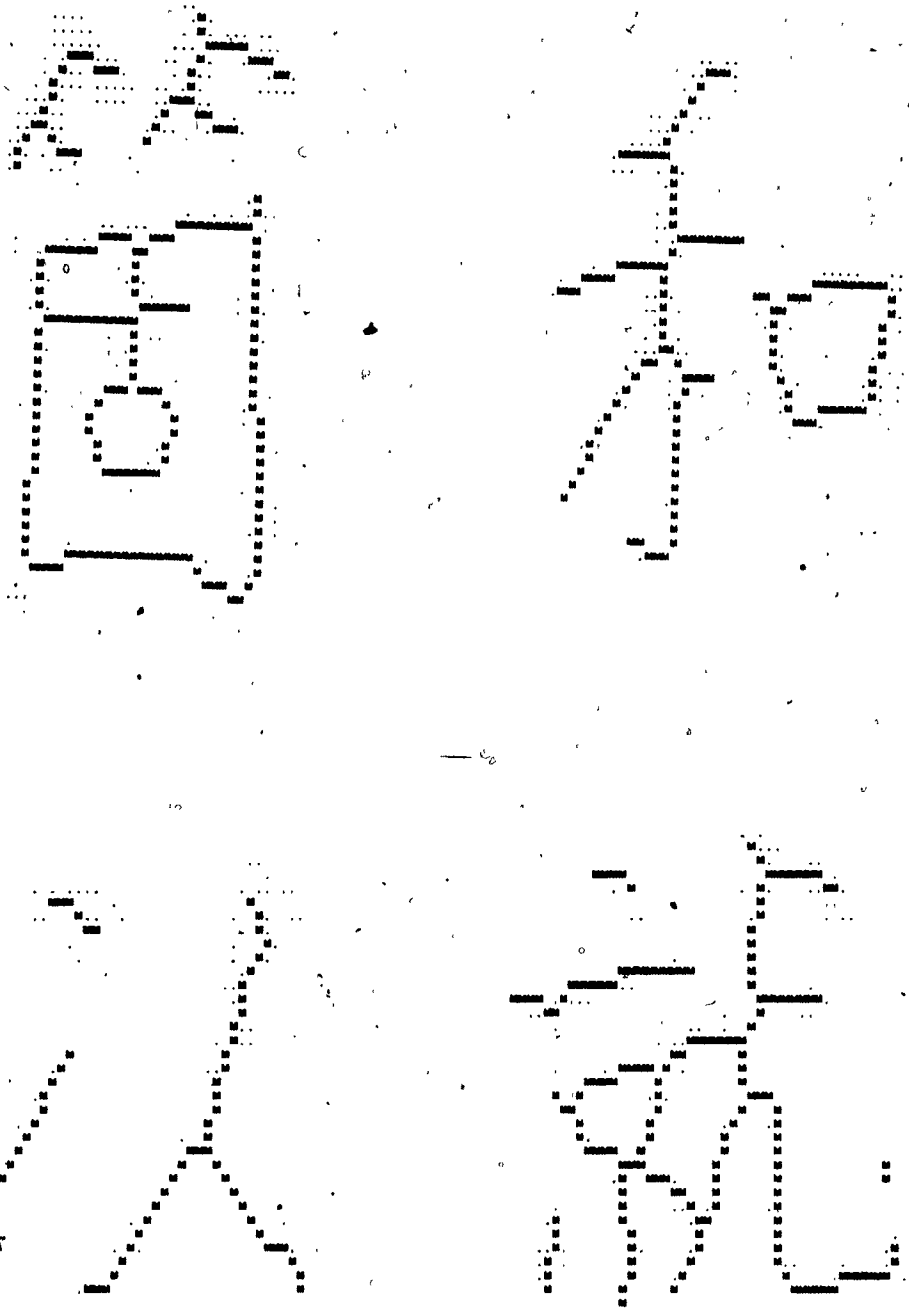


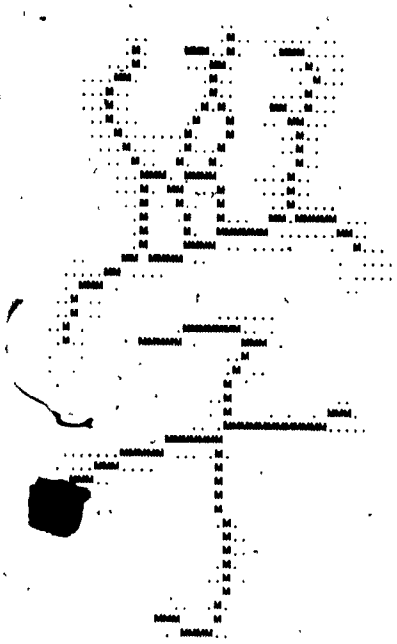


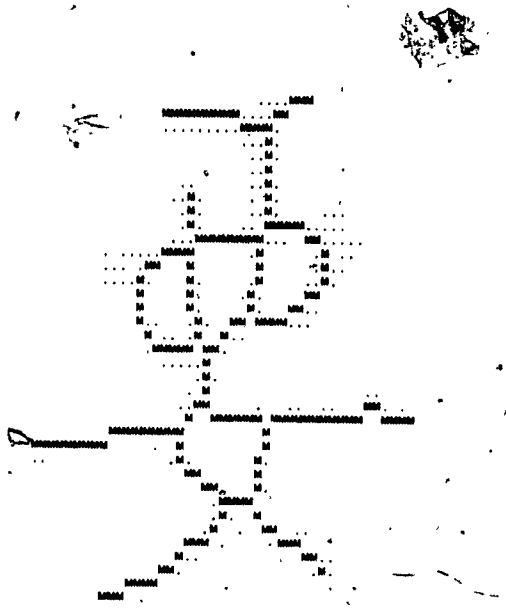


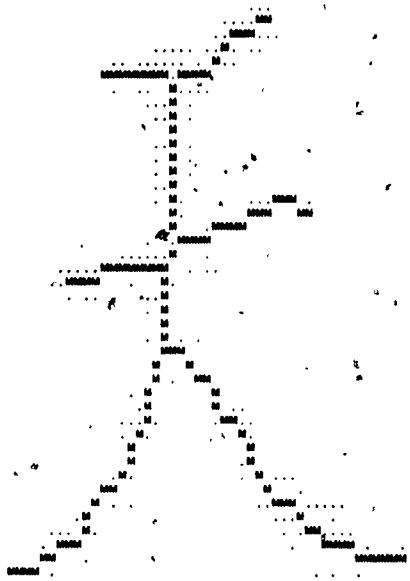


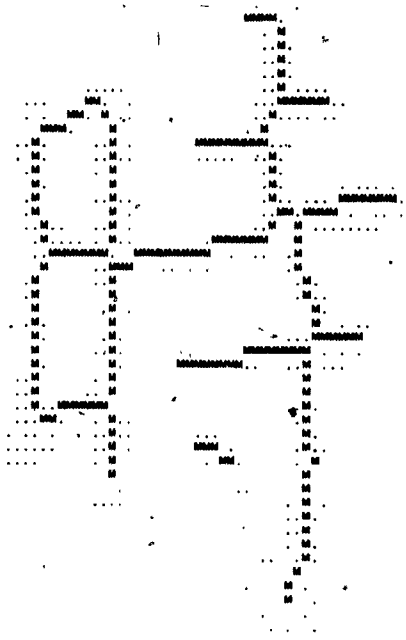


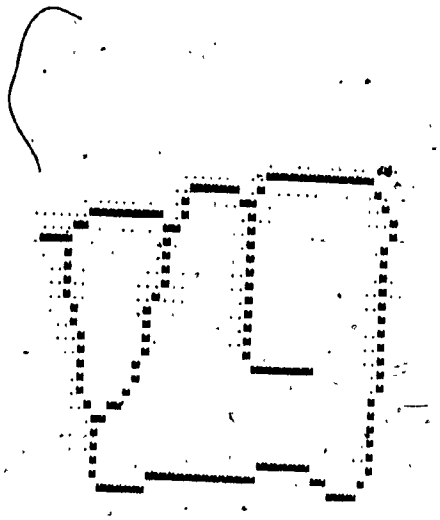


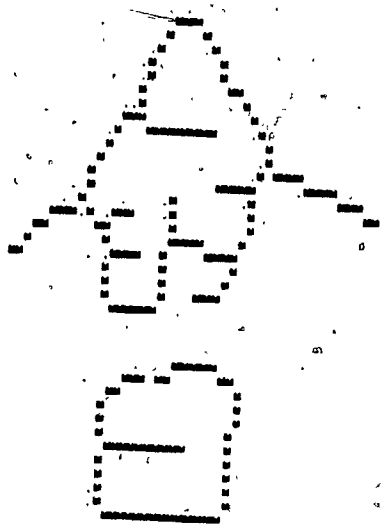


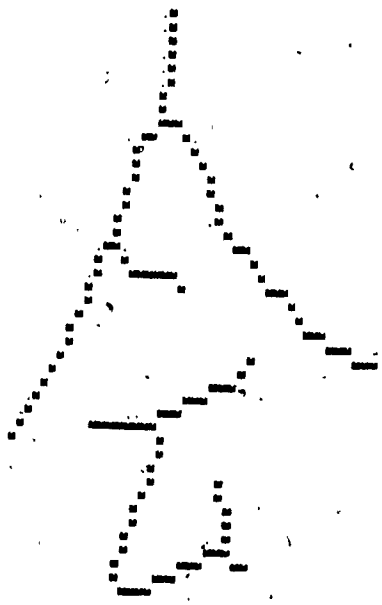


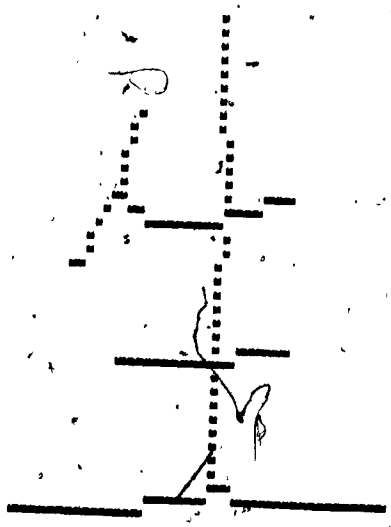


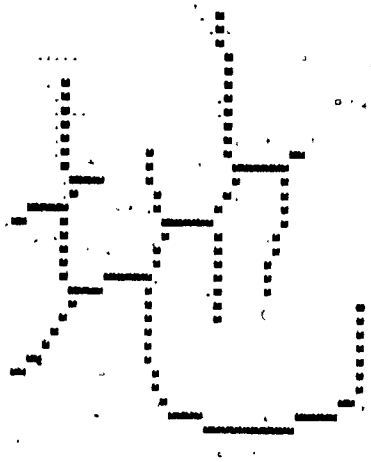


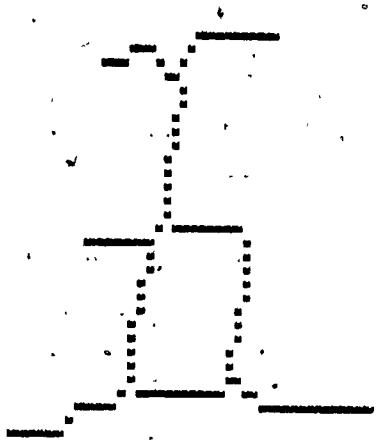


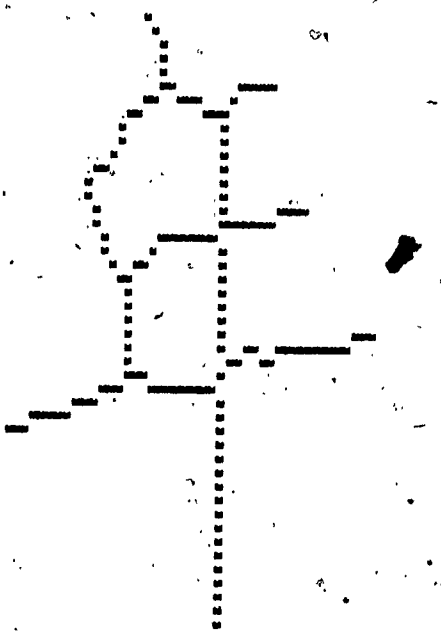


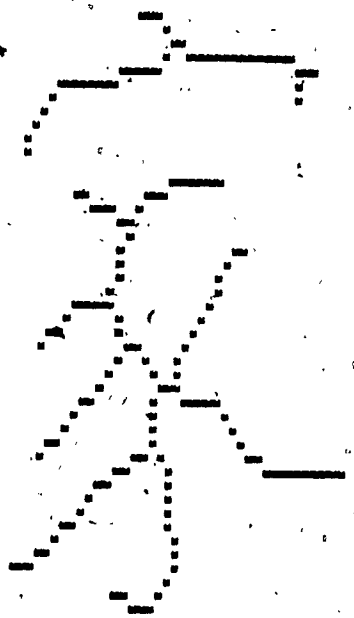




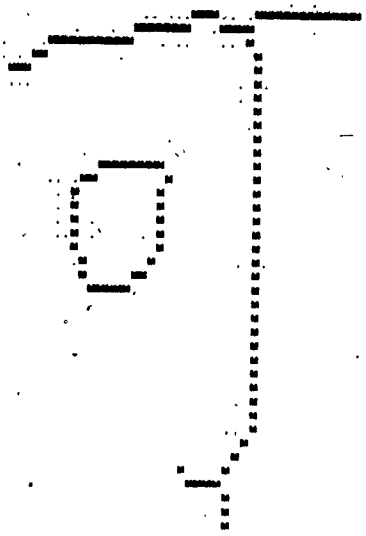


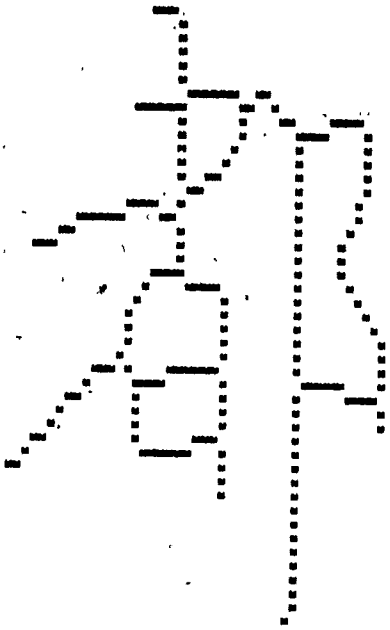


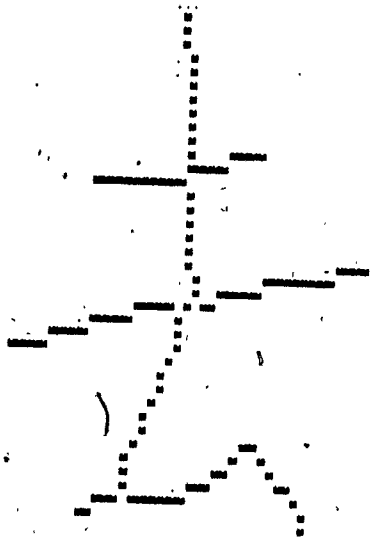


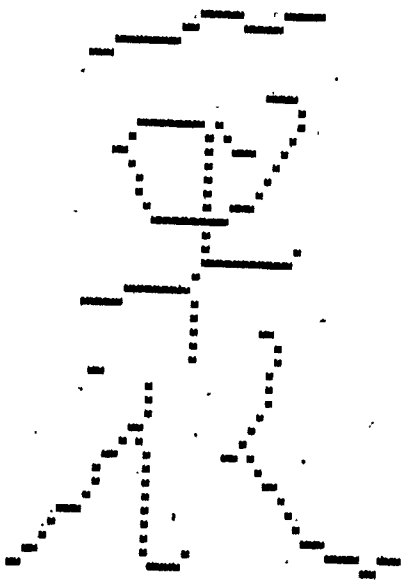


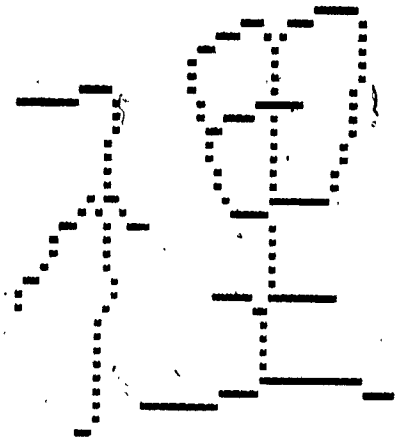
朋小
說第

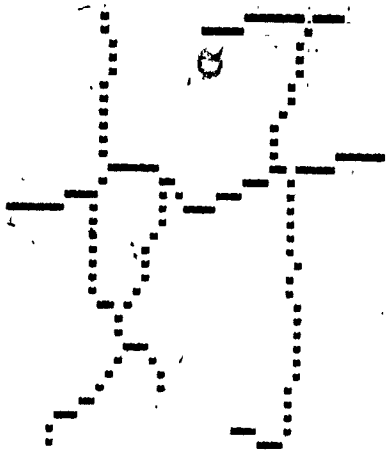


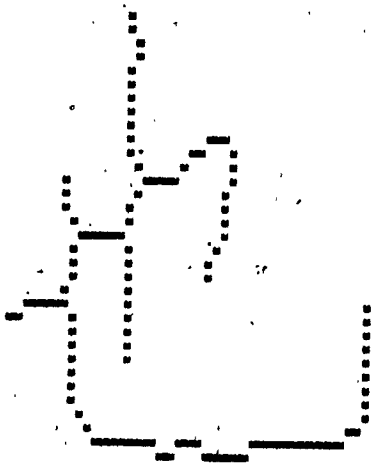


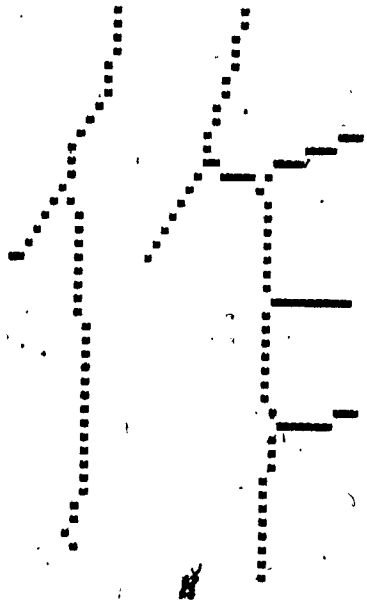


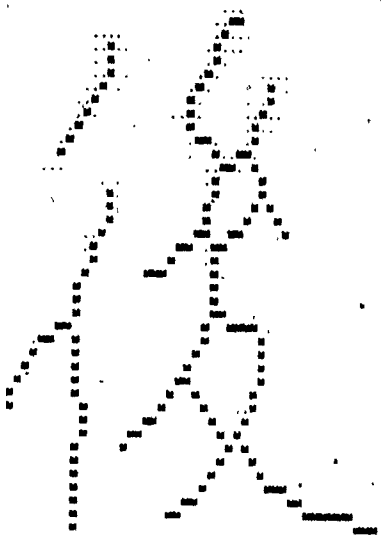


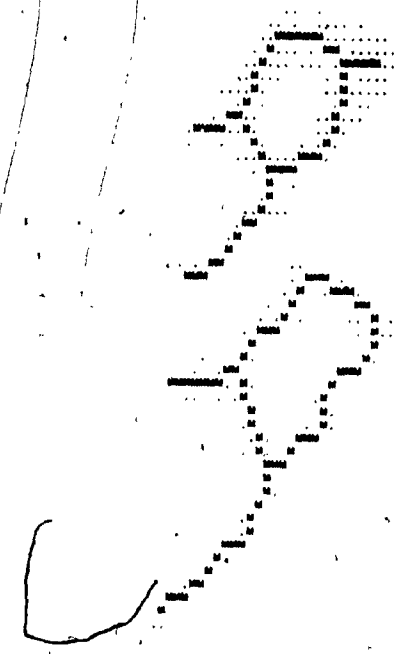


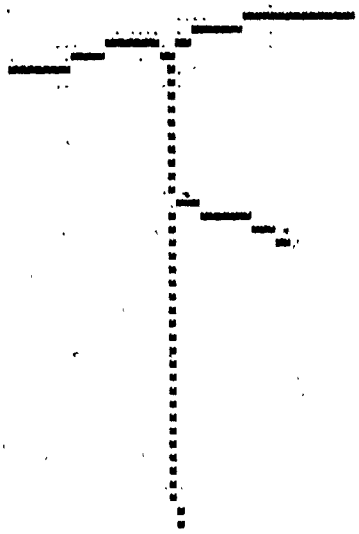


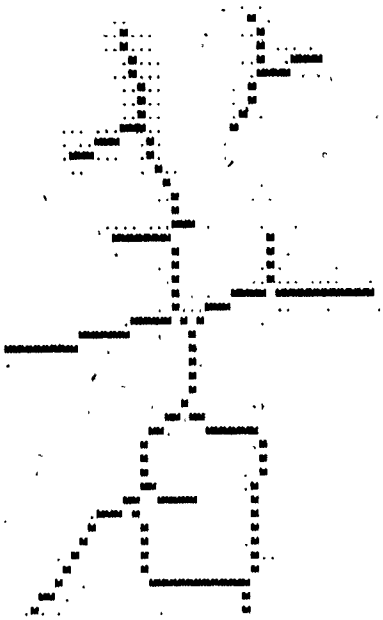


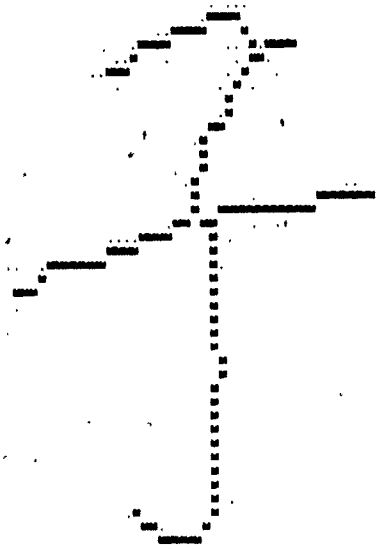


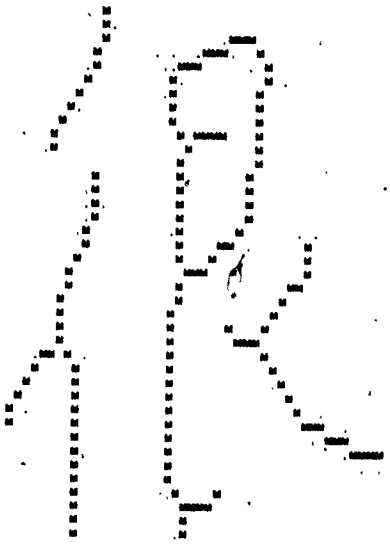


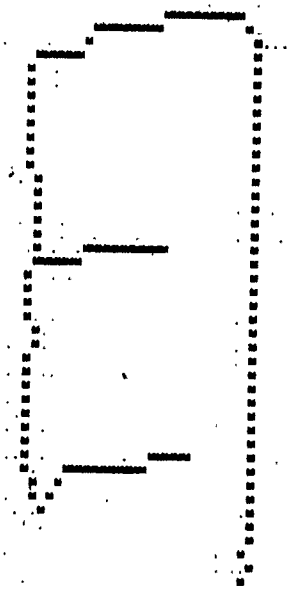


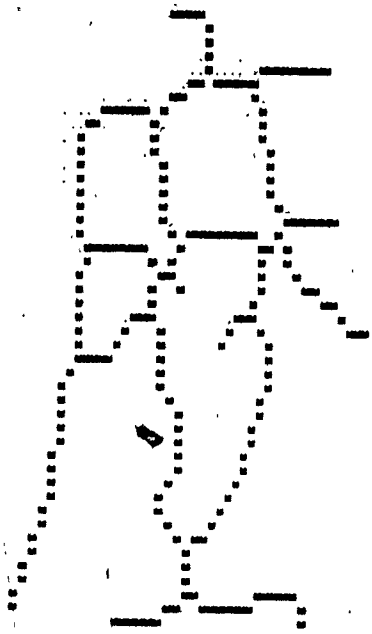


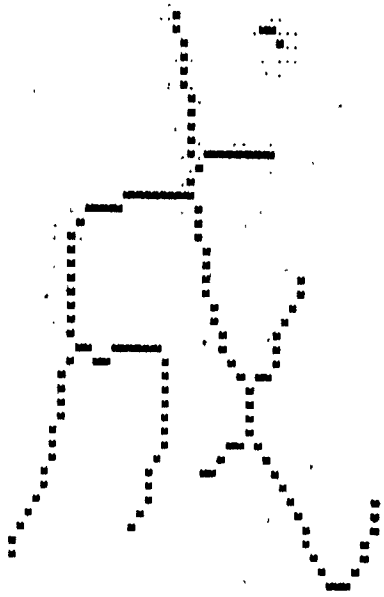


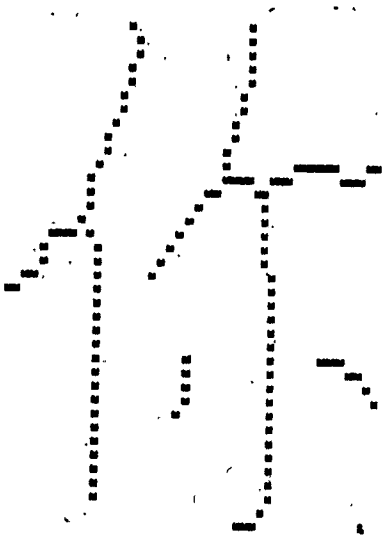


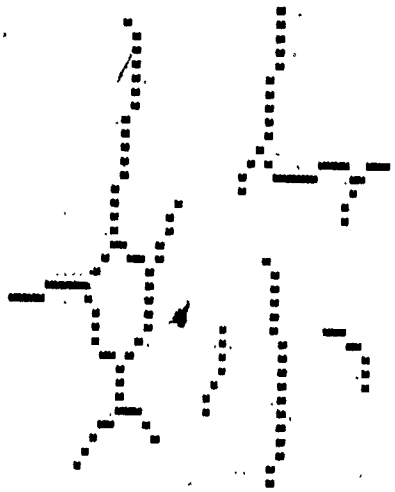


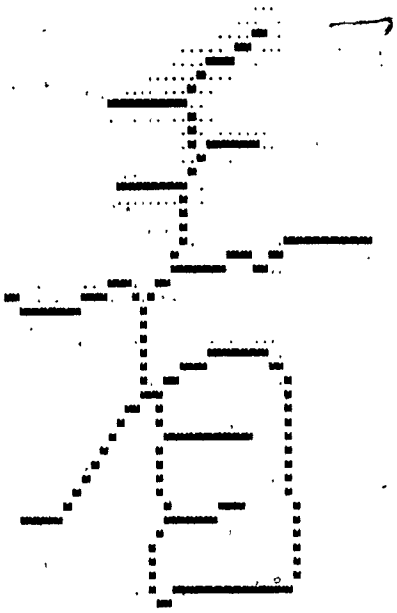


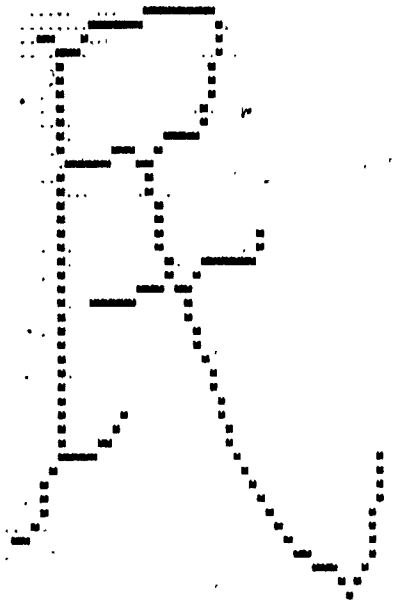


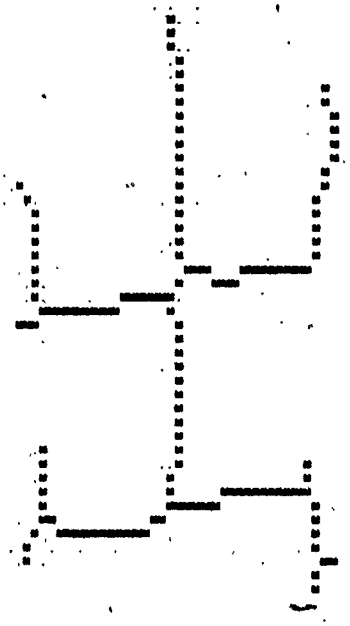


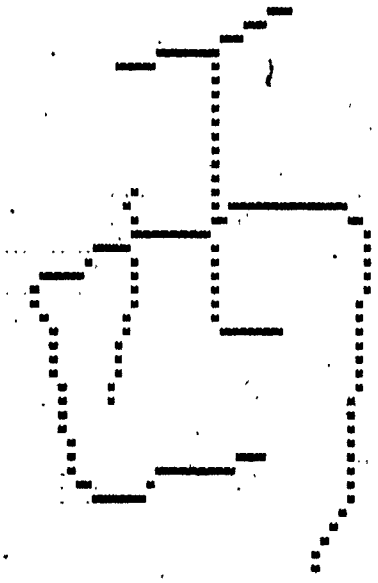


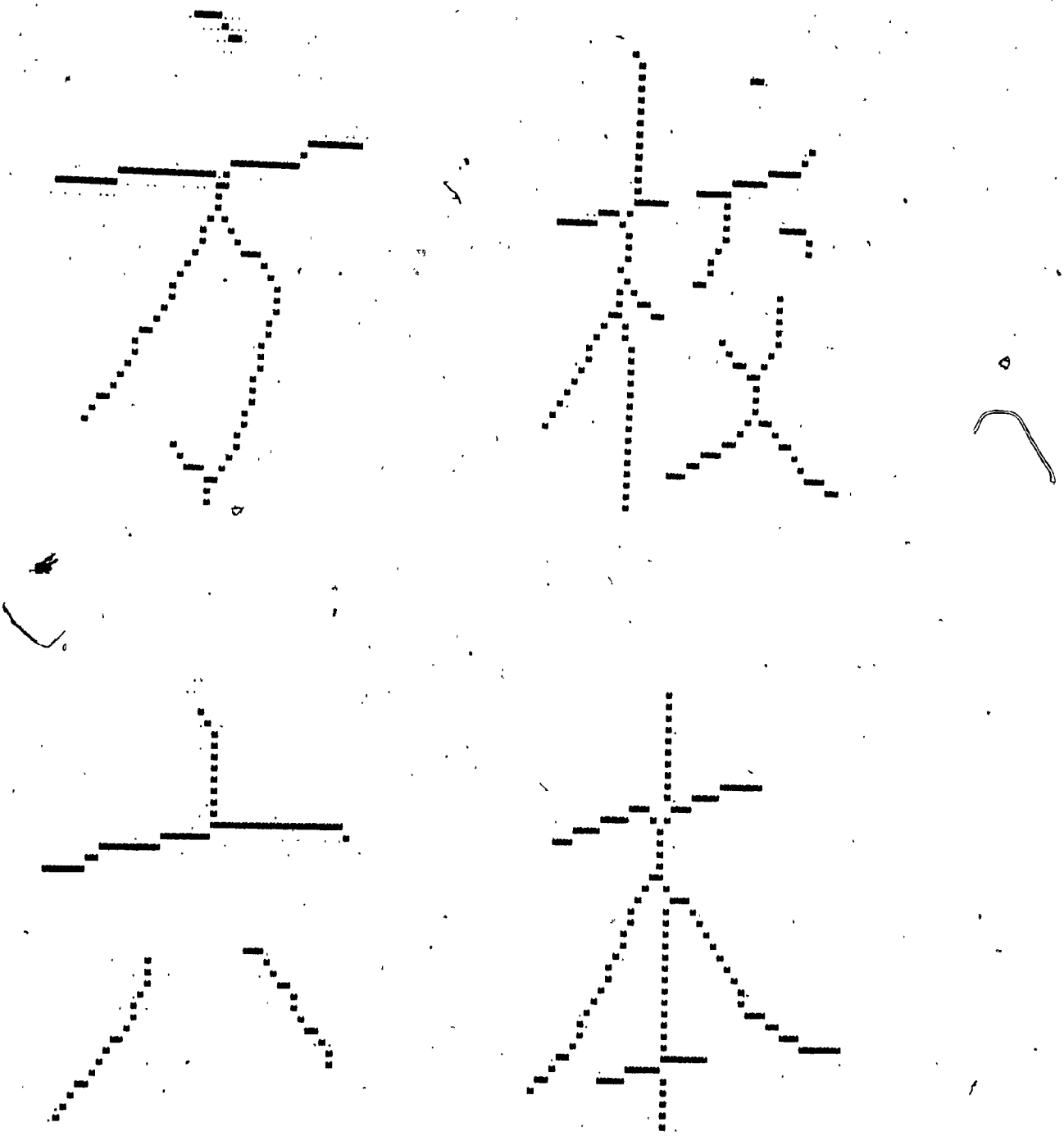


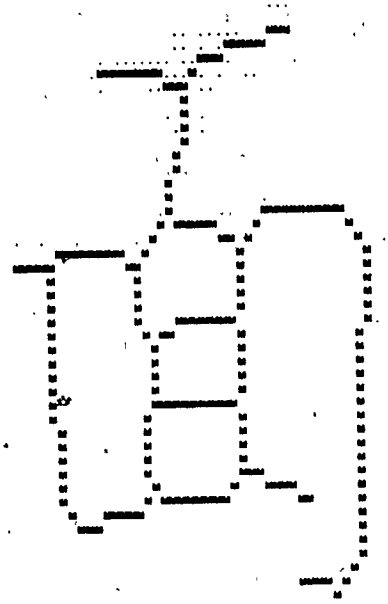


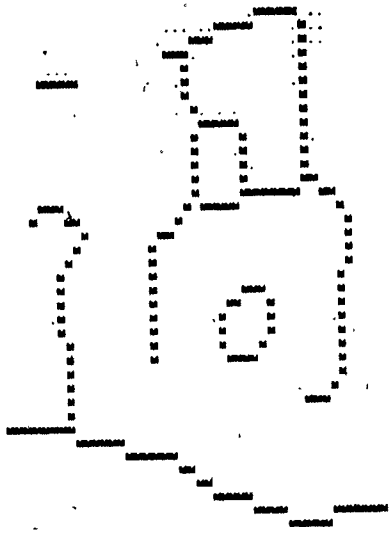


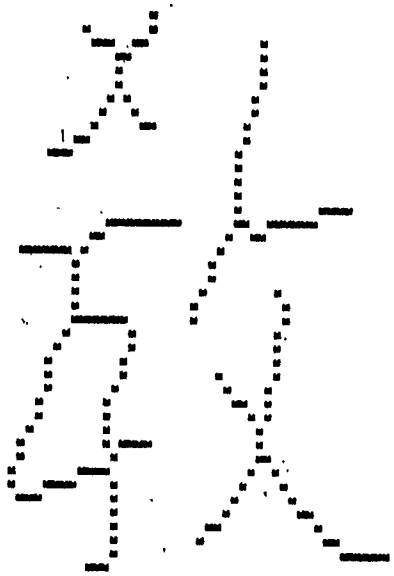


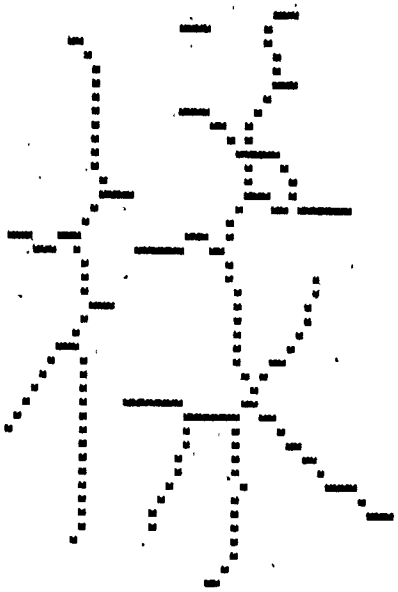






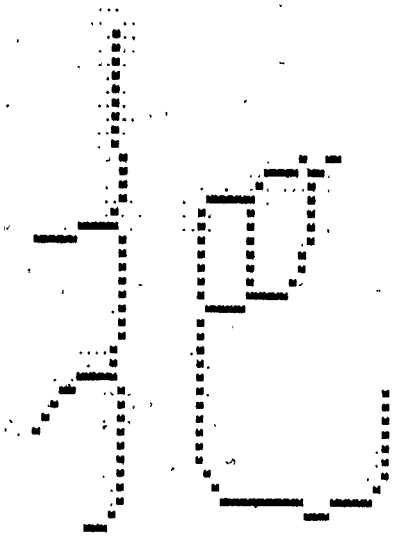


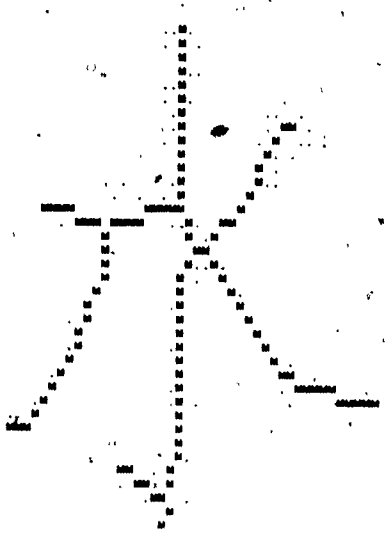


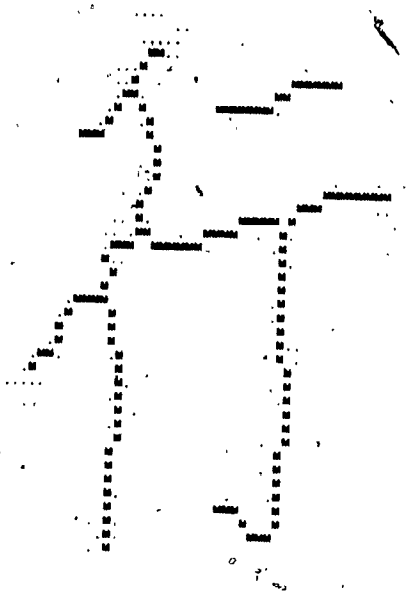


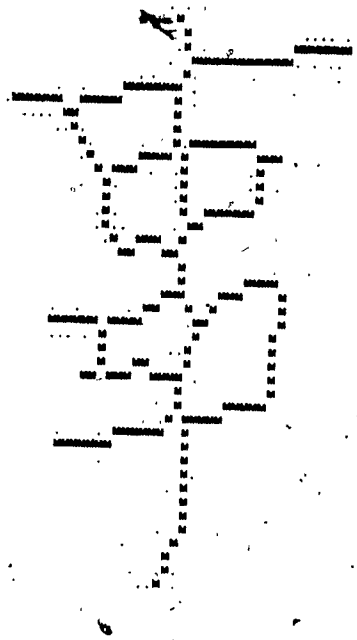
能　那

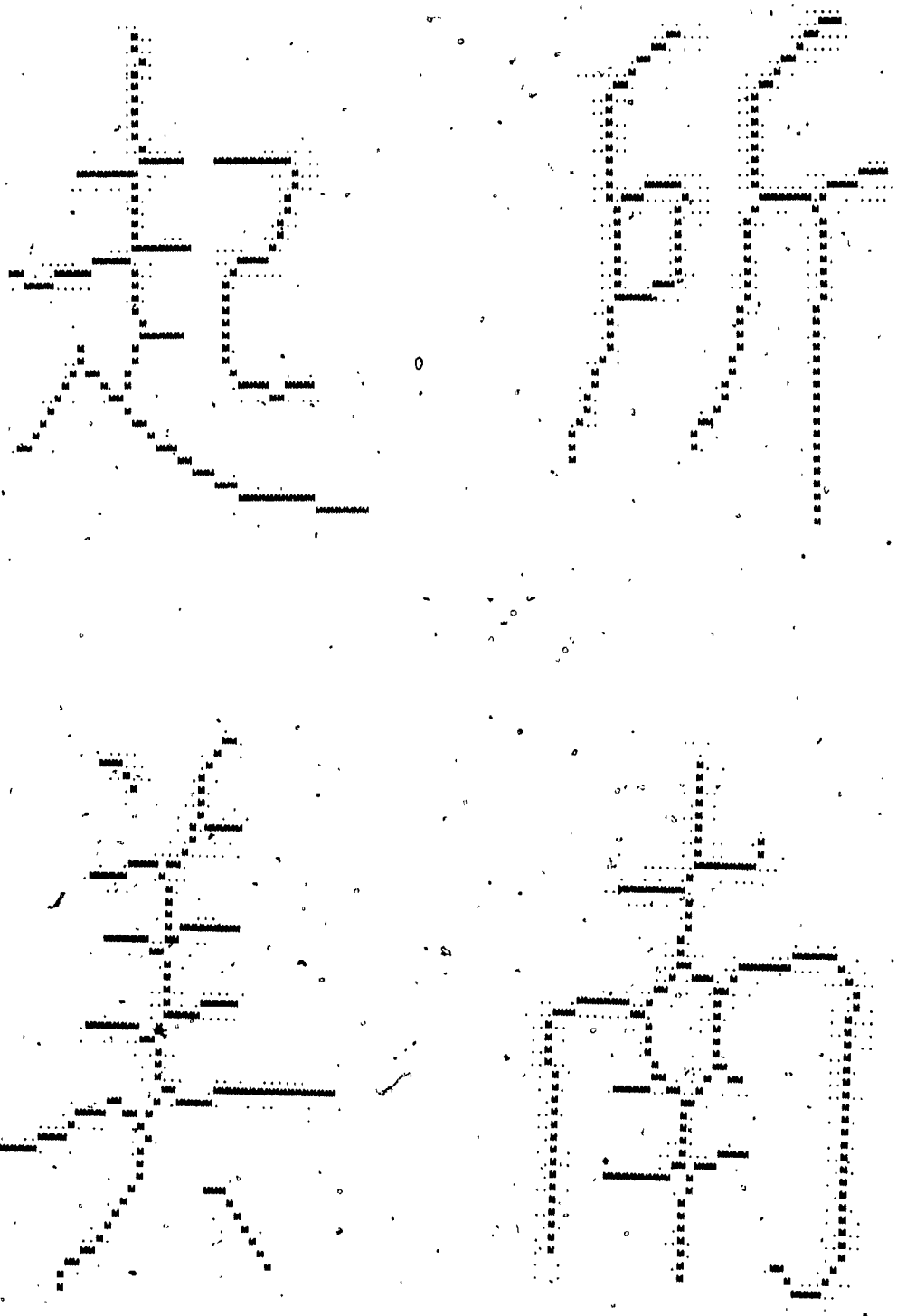
北　發

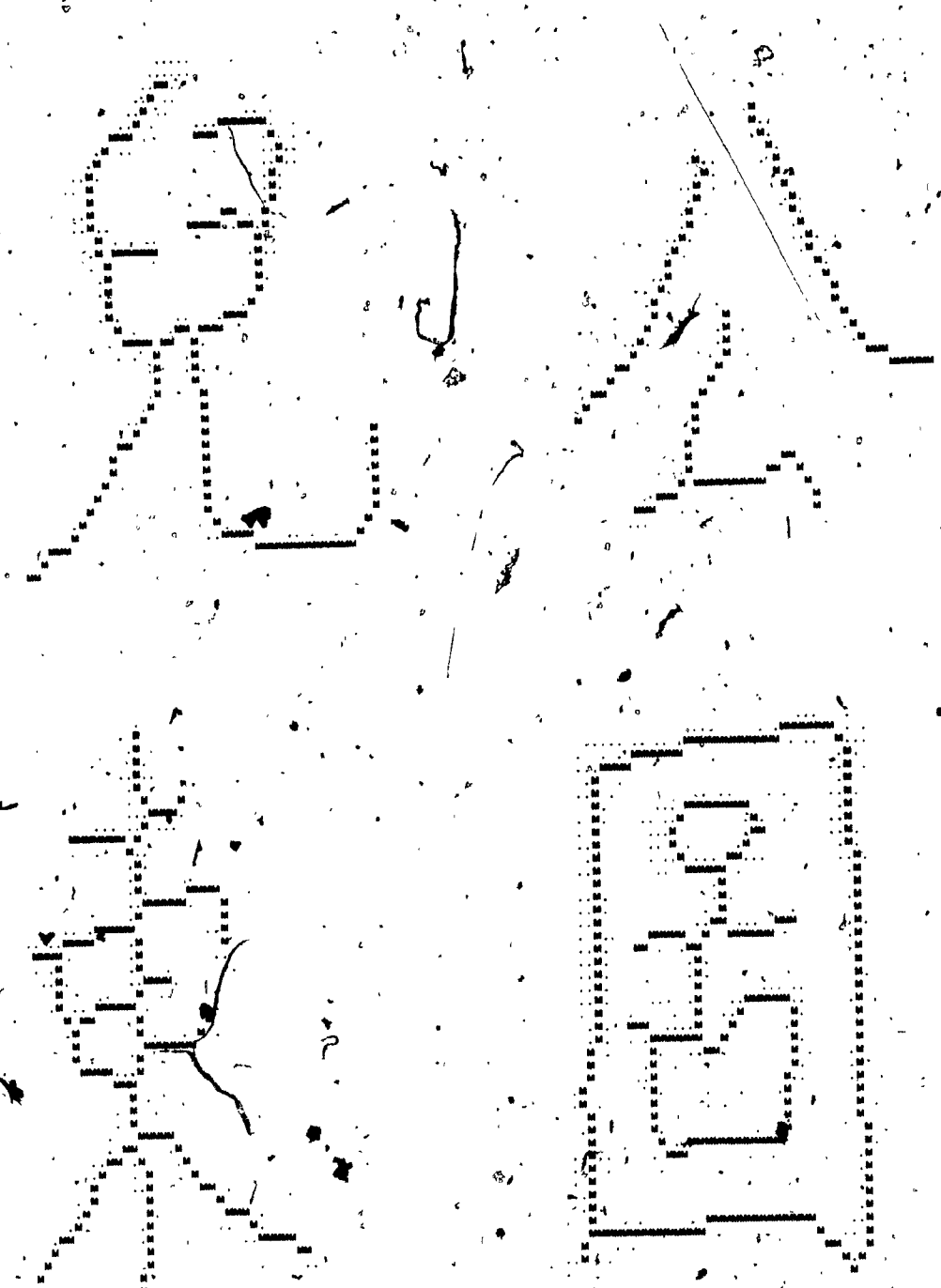


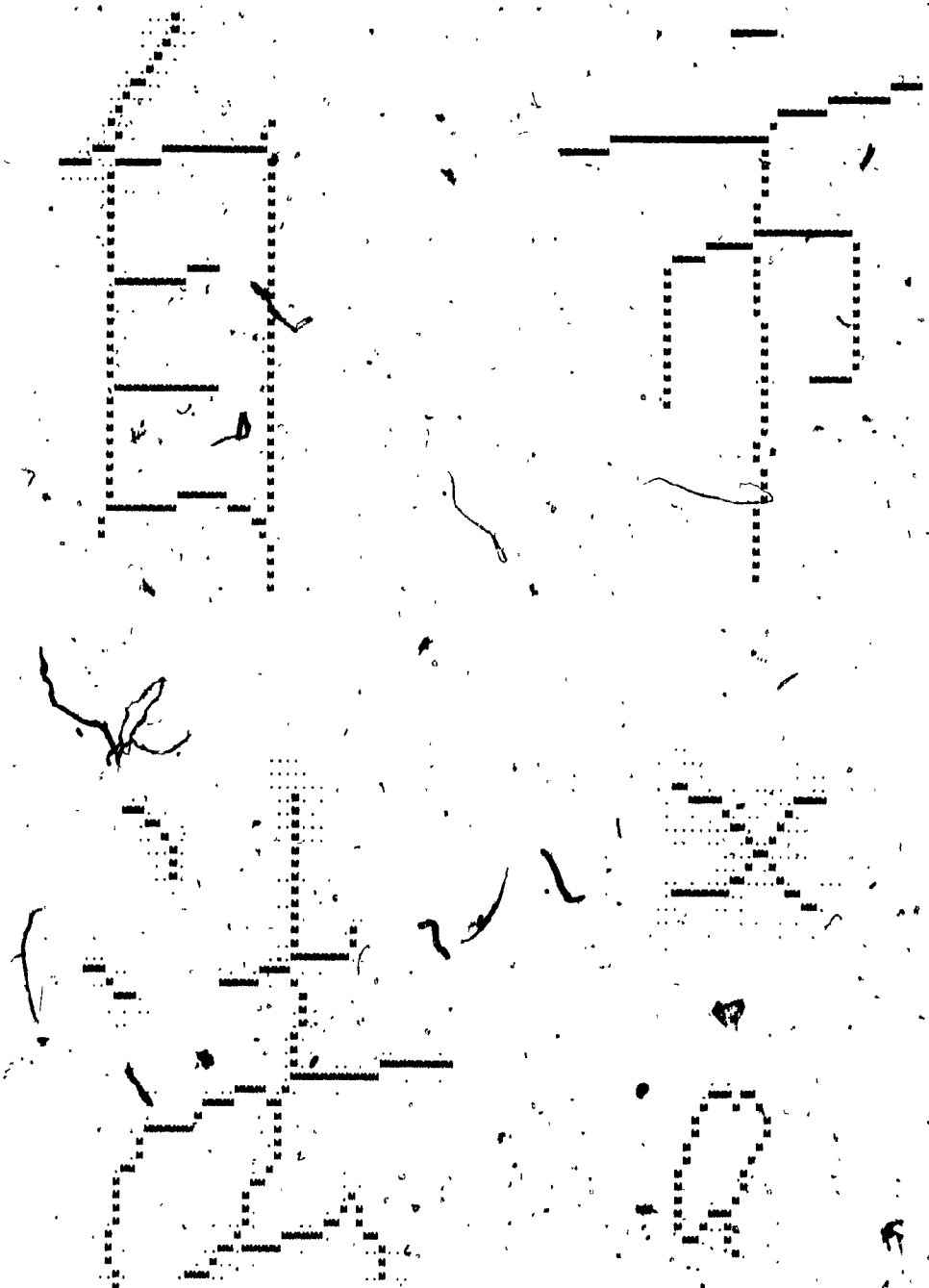


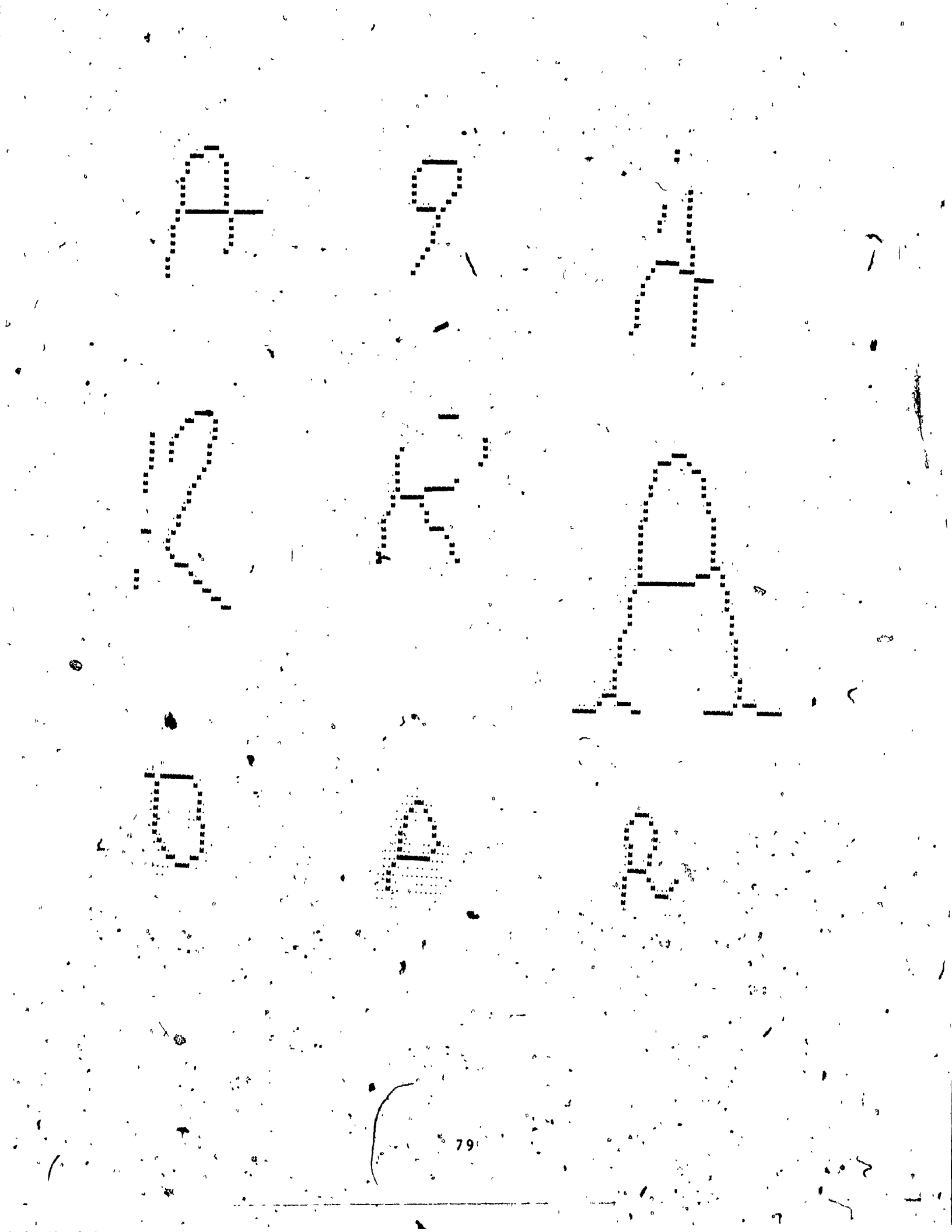


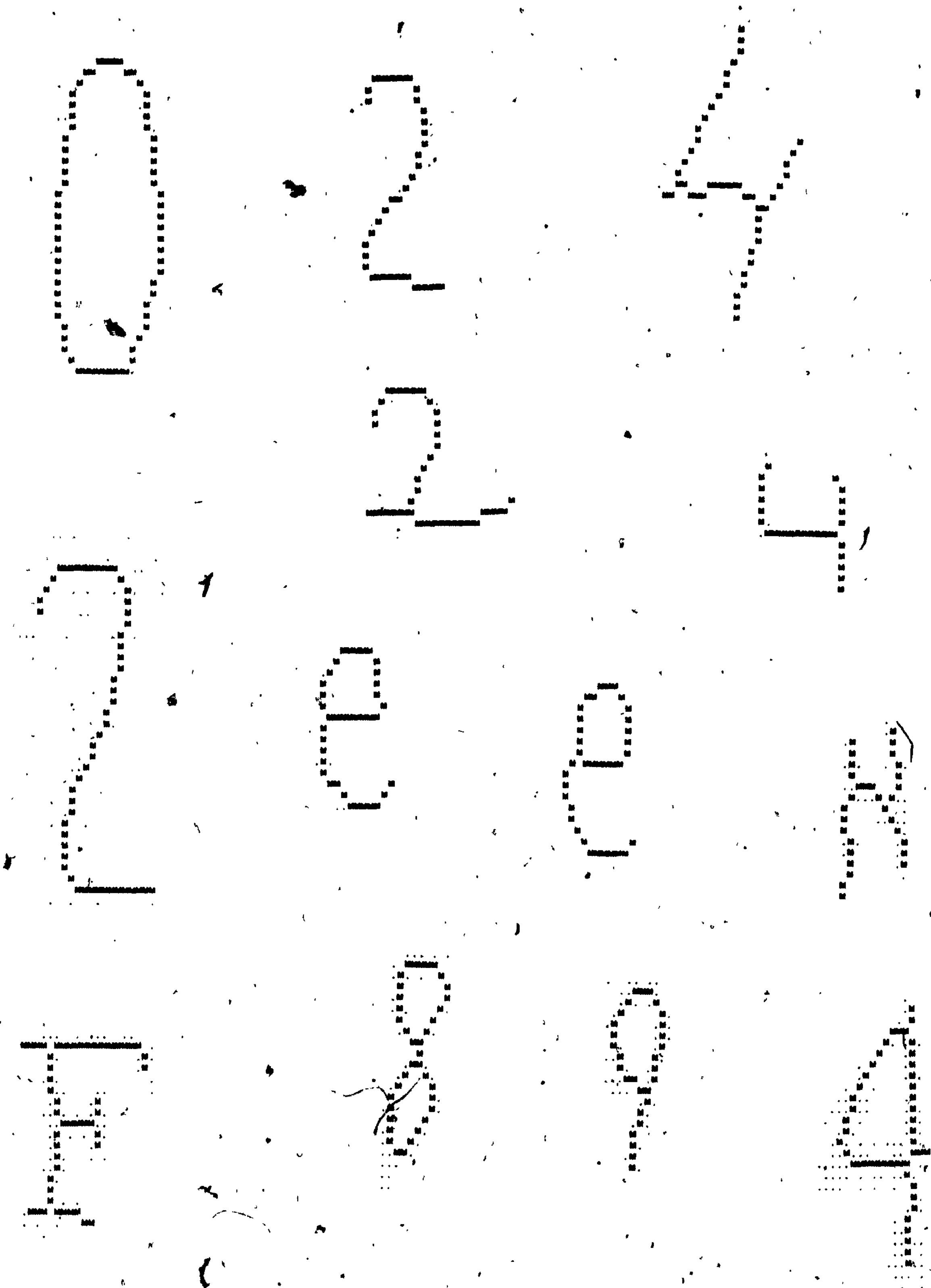


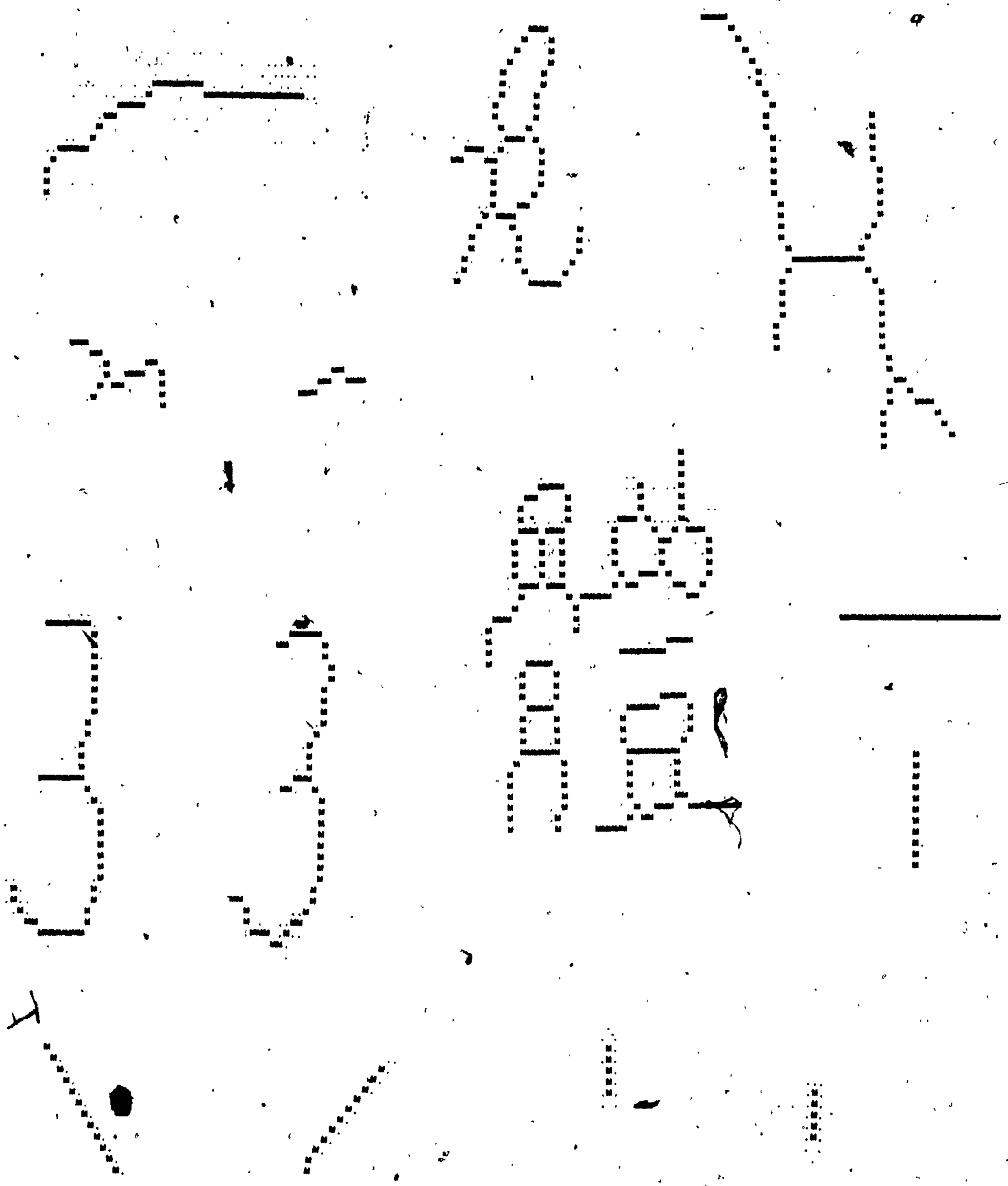


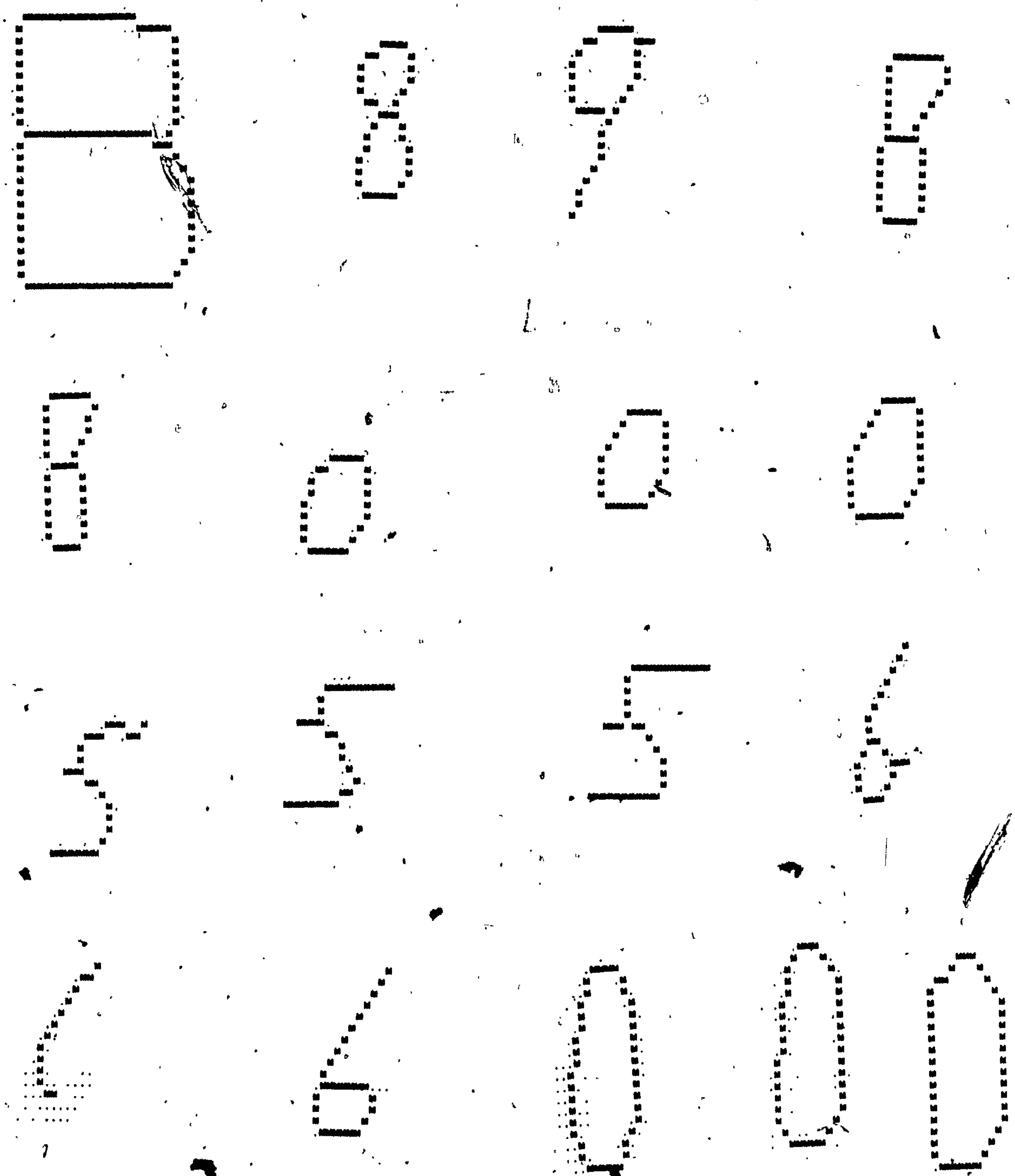


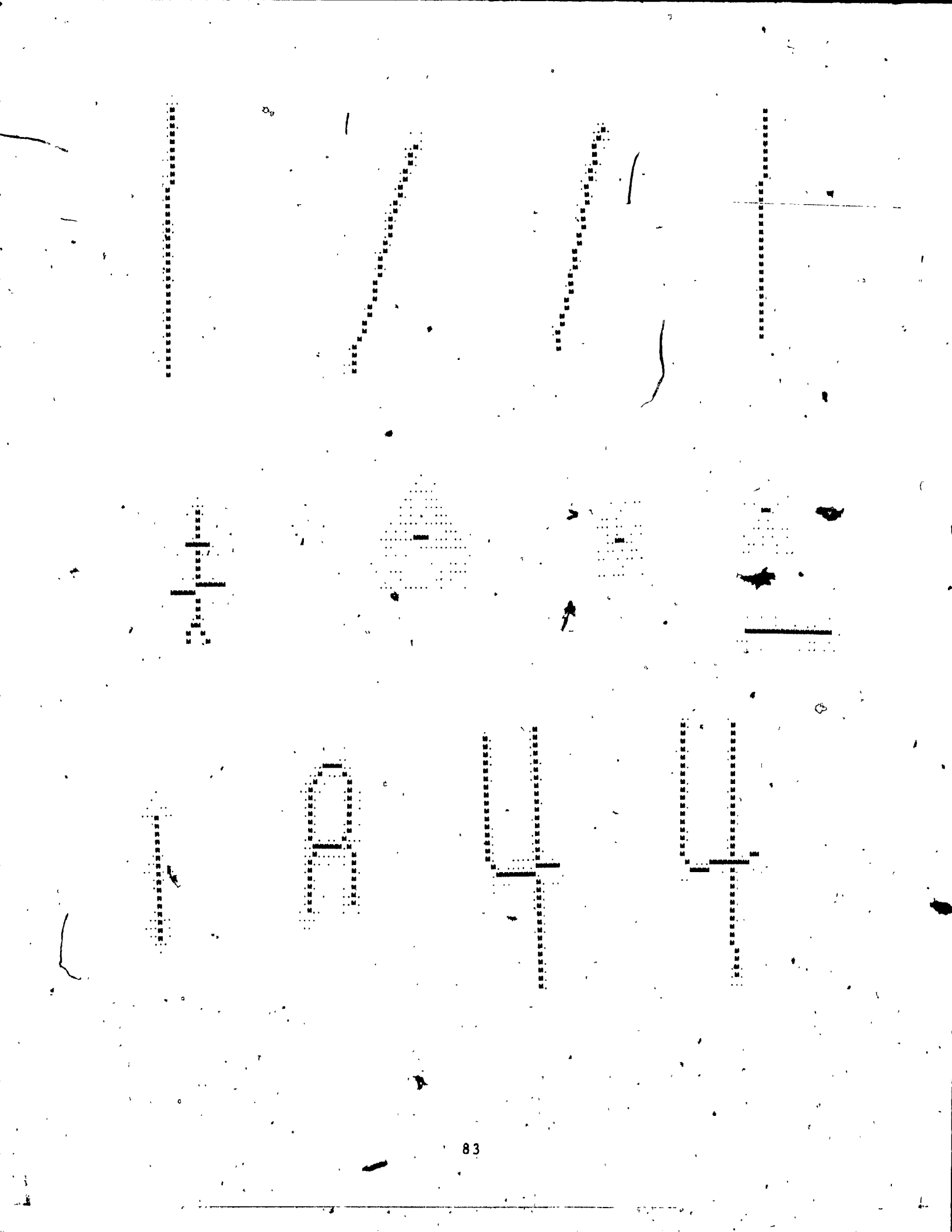


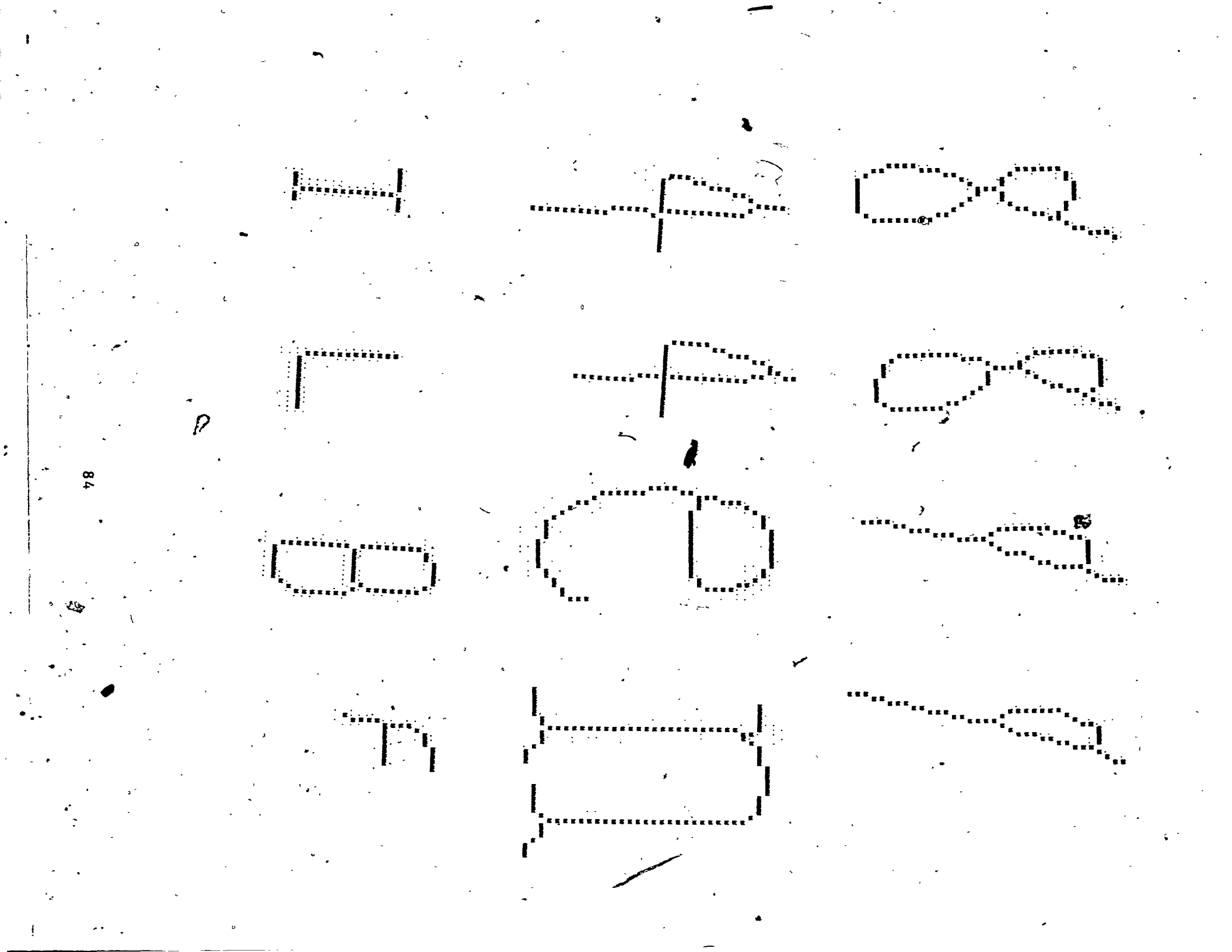


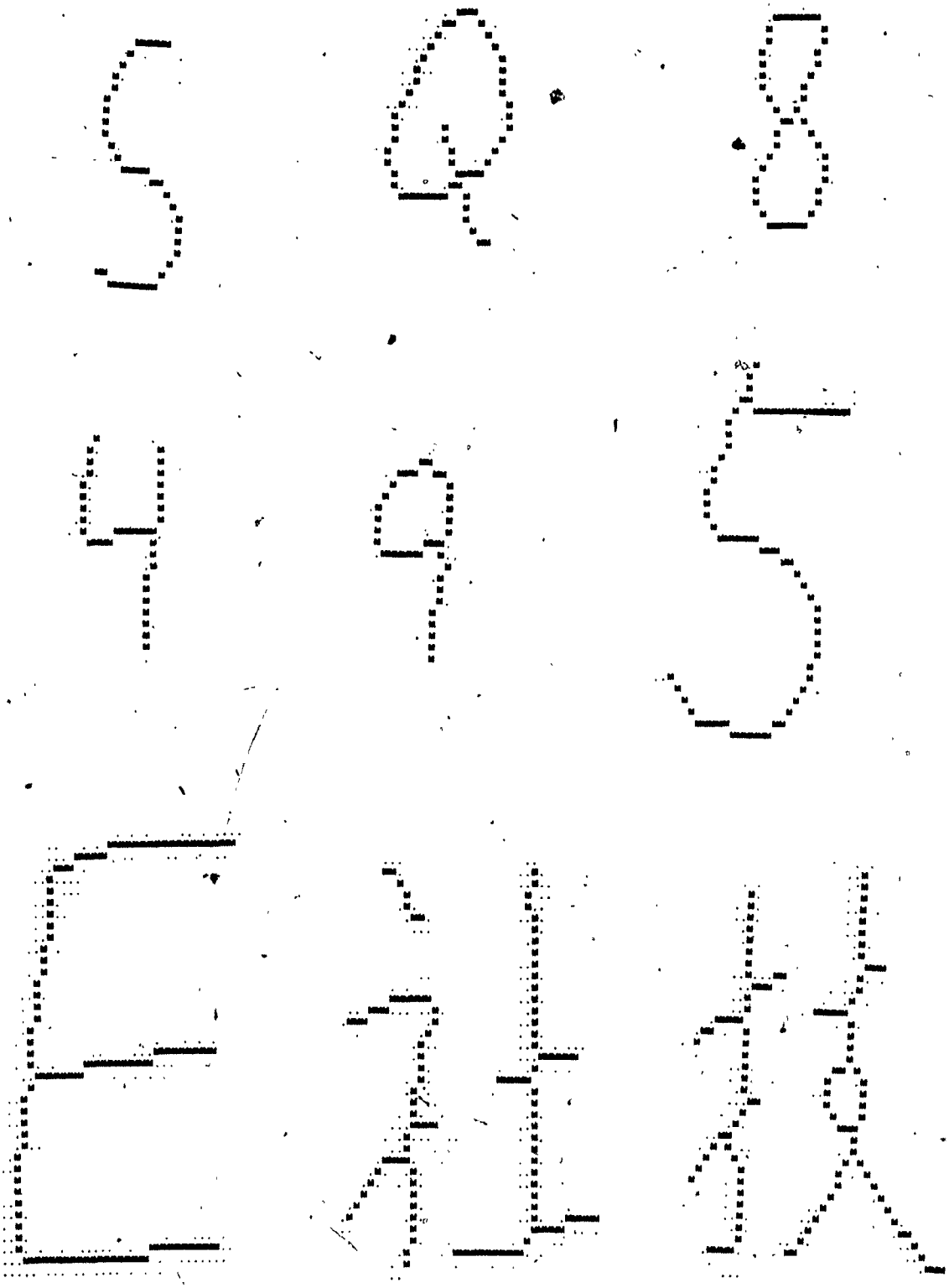












6. REFERENCES

1. H. Sherman, "A quasi-topological method for the recognition of line patterns", Information Processing, Proc. UNESCO Conf. pp. 232-238. Butterworths, London (1959).

2. A.P. Pullen, "Automatic visual inspection of complex industrial components", paper read at BPRA meeting in University College, London, February (1977).

3. R.N. Dixon and C.J. Taylor, "Automated asbestos fibre counting", Proc. IOP Conf. on Machine-aided Image Analysis, pp. 178-185. Institute of Physics, London (1979).

4. C.J. Hilditch, "Linear skeletons from square cupboards", Machine Intellingence IV, eds. B. Meltzer and D. Michie, pp. 403-420 Edinburgh University Press, Edinburgh (1969).

5. J.F. O'Callaghan and J. Loveday, "Quantitative measurements of soil cracking patterns", Pattern Recognition, Vol. 5, pp. 83-98 (1973).

6. C.V.K. Rao, B. Prasada and K.R. Sarma, "An automatic fingerprint classification system", Proc. 2nd Int. Joint Conf. on Pattern Recognition, pp. 180-4 (1974).

7. J.D. Dessimoz, "Specialized edge-trackers for contour extraction and line-thinning", Signal Processing, Vol. 2, pp. 71-73 (1980).

8. M. Beun, "A flexible method for automatic reading of handwritten numerals", Philips Tech. Rev. Vol. 33, pp. 89-101 and Vol. 33, pp. 130-137 (1973).

9. K.J. Udupa and I.S.N. Murthy, "Some new concepts for encoding line patterns", Pattern Recognition Vol. 7, pp. 225-233 (1975).

10. T. Pavlidis, "An asynchronous thinning algorithm", Computer Graphics and Image Processing, Vol. 20, pp. 133-157 (1982).

11. G.F.P. Deeker and J.P. Penny, "On interactive map storage and retrieval", Information, Vol. 10, pp. 62-74 (1972).

12. T. Kreifelts, "Skelettierung und Linienverfolgung in Raster digitalisierten Linienstrukturen", GL/NTG Congress on Digital Image Processing, Munich, March (1977). Springer, Informatik-Fachberichte, Nr 8.

13. P. Seuffert, "An application of line and character recognition in cartography", Proc IEEE Congress on Pattern Recognition, pp. 338-343 Troy, New York, July (1977).

14. D. Rutovitz, Pattern recognition. J.R. statist. Soc. 129, pp. 504-30 (1966).

6. REFERENCES (Cont'd)

15. R.N. Jones and M.C. Fairhurst, "Skeletonisation of binary patterns: a heuristic approach", Electron. Lett. Vol. 14, pp. 265-266 (1978).

16. E.S. Deutsch, "Thinning algorithms on rectangular, hexagonal, and triangular arrays", Communications of ACM, Volume 15, No. 9, pp. 827-837, September (1972).

17. R. Stafanelli and A. Rosenfeld, "Some parallel thinning algorithms for digital pictures", Journal of the Association for Computing Machinery, Vol. 18, No. 2, pp. 255-264, April (1971).

18. Hideyuki Tamura, "A comparison of line thinning algorithms from digital geometry viewpoint", Proc. 4th Int. Joint Conf. on Pattern Recognition, pp. 715-719 (1978).

19. S. Tsuruoka - From H. Tamura

20. T.Y. Zhang and C.Y. Suen, "A fast parallel algorithm for thinning digital patterns", Communications of the ACM, Volume 27, No. 3, pp. 236-239, March (1984).

21. C. Arcelli, "A condition for digital points removal", Signal processing, Volume 1, pp. 283-285 (1979).

22. H. Ma, "A comparative study of thinning algorithms in pattern processing", Master's Major Report, Dept. of Computer Science, Concorida University, November (1983).

23. S. Zhang and K.S. Fu, "A thinning algorithm for discrete binary images", The First International Conference on Computers and Applications, Beijing, China, pp. 879-886, June 20-22 (1984).

24. N.J. Naccache, "Skeletonization of Binary Patterns: A proposed Algorithm and a multiprocessor network", Master thesis, Dept. of Computer Science, Concordia University, May (1984).

25. N.J. Naccache and R. Shinghal, "SPTA - A proposed algorithm for thinning binary patterns", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-14, No.3, pp. 409-418, May/June (1984).

26. E.R. Davies and A.P.N. Plummer, "Thinning algorithm: A critique and a new methodology", Pattern Recognition, Vol. 14, Nos. 1-6, pp. 53-63 (1981).