Pattern Detection and Recognition*

S. H. UNGER†, MEMBER, IRE

Summary—**Two types of pattern-processing problems are discussed in this paper. The first, termed "pattern detection," consists of examining an arbitrary set of figures and selecting those having some specified form. The second problem, "pattern recognition," consists of identifying a given figure which is known to belong to one of a finite set of classes. This is the problem encountered when reading alphanumeric characters.**

Both recognition and detection have been successfully carried out on an IBM 704 computer which was programmed to simulate a spatial computer[1] (a stored-program machine comprised of a master control unit directing a network of logical modules). One of the programs tested consisted of a recognition process for reading hand-lettered sans-serif alphanumeric characters. This process permits large variations in the size, shape, and proportions of the input figures and can tolerate random noise when it is well scattered in small specks.

Programs for detecting L-shaped (or A-shaped) figures in the presence of other randomly drawn patterns have also been successfully tested.

I. Introduction

A PRINCIPAL goal of the research upon which this report is based is to determine how machines can be made to duplicate some of the remarkable feats that humans perform daily when dealing with visual images. For example:

1) Presented with a new coloring book, a five-year-old child quickly points out all outlines of dogs.
2) A post-office clerk deciphers the name of a city in a carelessly written address.
3) A hold-up victim selects a photograph of his assailant from among the collection in police files.
4) A frequent visitor to an art museum identifies the creator of a newly acquired painting merely from its general appearance and style, and the knowledge that it was done by one of the better known nineteenth-century artists.

Examples 1 and 3 illustrate what we shall term *pattern detection*. This is the process of examining a set of figures and selecting those that fall into some particular class of patterns, defined as the *target set*. Examples 2 and 4 involve a different kind of operation, which we shall define as *pattern recognition*. Here a single input pattern known to belong to one of several known classes is to be identified. In other words, the problem is to specify which of a finite number of labels should be attached to the input. Thus, in example 2, the postal clerk knew that the scrawled word that he was trying to decipher was the name of one of the cities in the state of New Jersey, and his object was to decide *which* one. Compare this with the detection problem of example 1. Here, the child was required to search through an assortment of patterns selected from an unlimited ensemble and to indicate which, if any, were pictures of dogs. Note that detection subsumes recognition in that any recognition problem can be treated as a finite number of detection problems. The converse is not true.

Both detection and recognition will be discussed here in some detail with alphanumeric characters being used as the patterns of interest. Systems have been devised and successfully tested for accomplishing the following:

1) Recognizing any given hand-printed alphanumeric character.
2) Examining an input field consisting of a set of randomly drawn patterns and detecting all L-shaped figures that are present. (A-shaped figures have also been detected.)

The above-mentioned systems consist of programs written for the spatial computer.[1] A brief description of this machine (abbreviated SPAC) follows, and a somewhat more detailed description including the order structure, is included in an appendix.

SPAC consists of a rectangular network of logical modules directed by a master control unit. Each module has direct contact with its four immediate neighbors (Fig. 1) and receives orders from the master control, which issues identical commands to all modules in the network.

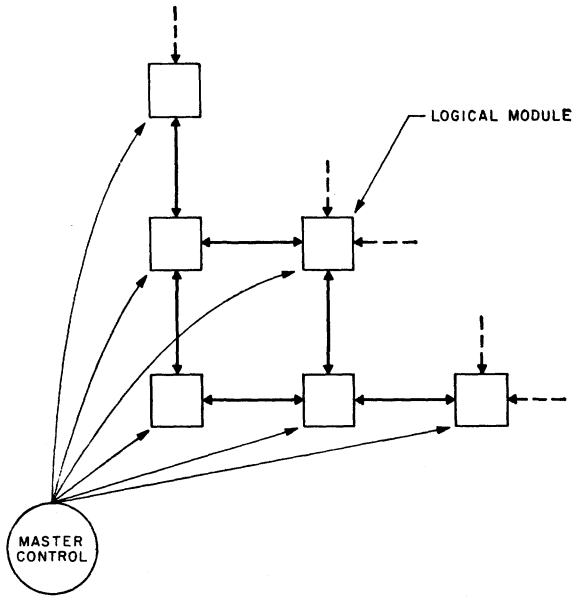


Fig. 1—Structure of spatial computer.

* Original manuscript received by the IRE, April 30, 1959; revised manuscript received, May 9, 1959.
† Bell Telephone Laboratories, Inc., Whippany, N. J.
[1] S. H. Unger, "A computer oriented toward spatial problems," PROC. IRE, vol. 46, pp. 1744–1750; October, 1958.

A module consists of some logical circuitry, a one-bit principal register (PR), and a set of one-bit memory registers (MR's) individually addressable as MR1, MR2, etc. (In the original description of SPAC,[1] the PR's are called accumulators.) A pattern can be stored in the PR's or in any of the MR arrays.

The master control includes a random-access memory for storing instructions, an instruction counter, and decoding circuitry. It operates like the control unit of a conventional digital computer, reading instructions out of memory in sequence, decoding them and sending appropriate control signals out on a set of buses feeding the modules.

The SPAC programs that will be discussed here were tested by means of a simulation program on an IBM 704 Computer. The simulated SPAC has a 36×36 array of modules, and there are nine MR's per module.

Important work in the pattern processing area has been done by Selfridge[2,3] and Dinneen[4] who performed a number of interesting experiments on a digital computer which indicated that much could be accomplished through the use of a small number of very elementary operations, such as local averaging, spatial differentiation, and blob counting. Their primary goal was to investigate the learning process and they were not concerned with finding efficient recognition or detection procedures.

Kirsch[5] and some of his colleagues at the Bureau of Standards have made use of the SEAC computer to study various pattern processing operations. Their objective was to develop a library of computer subroutines that might be useful in pattern processing. In addition to operations similar to those used by Selfridge and Dinneen, the Bureau of Standards group has demonstrated some interesting properties of an operation in which patterns are alternately complemented and spatially differentiated. No effort was made to develop specific recognition or detection systems.

A report by Greanias *et al.*[6] describes a system for recognizing members of a sixteen character alphabet (numerals and some miscellaneous printer symbols). The system has been tested via a computer program and has successfully identified samples of printed characters in the presence of background noise. As described in the paper, this method does not permit variations in the size and proportions of the input characters.

T. L. Dimond[7] has developed a process for reading hand-printed numerals, provided that they are drawn in accordance with certain constraints that restrict size, proportion, and location.

The Solartron electronic reading automaton,[8] produced commercially in England, can read 120 printed numerals per second. Only small variations in size and style are permissible.

The amount of work being done on pattern processing problems is increasing too rapidly to permit a complete review of the field, but the preceding summary represents a reasonable cross section.

II. The Effects of Quantizing Patterns on a Grid

We shall not consider multi-color patterns or those represented in half-tones. The original inputs will be assumed to be representable in black and white.

As a first step in processing patterns, it is desirable to reduce the input field to a discrete form. This can be done by superimposing a grid or matrix of squares over the figure (assumed to be black on white) and placing a one in each square if the black area within that square exceeds a certain threshold. Zeros are filled in elsewhere (although zero-cells will usually be left blank in the diagrams). All of our input fields will be of this form.

Such a quantizing process (which can be physically realized by devices such as flying spot scanners) results in a loss of information concerning the original pattern, but this loss can always be reduced by decreasing the mesh of the grid (that is, by increasing the number of squares per unit area). If the grid is too coarse, certain details of the pattern will be lost. This effect can sometimes be made to serve a useful function in filtering out insignificant irregularities that might cause confusion. For example, if a character printed by a typewriter is magnified by a factor of several thousand diameters and projected on a screen it is generally somewhat difficult to identify because of the apparently random distribution of tiny blobs of ink on the paper. Thus, for a given input field, there is a limit on the amount of useful grid resolution.

Regardless of how fine a grid is used, there are still important transformations which occur when a continuous line is mapped onto a discrete grid. Assume that the threshold is set so that a one is placed in every cell touched by any portion of a black area. A straight vertical line will be mapped as a set of one-cells (squares with ones in them) one above the other, as shown in Fig. 2(a). Except for the change in thickness, if the one cells are blacked in the resulting picture will be the same as the original line.

[2] O. G. Selfridge, "Pattern recognition and learning," Symp. on Information Theory, London, Eng.; 1955.
[3] O. G. Selfridge, "Pattern recognition and modern computers," *1955 Proc. WJCC*, pp. 91–93.
[4] G. T. Dinneen, "Programing pattern recognition," *1955 Proc. WJCC*, pp. 94–100.
[5] R. A. Kirsch, L. C. Ray, L. Cahn and G. H. Urban, "Experiments in processing pictorial information with a digital computer," *1957 Proc. EJCC*, pp. 221–230.
[6] E. C. Greanias, *et al.*, "Design of logic for recognition of printed characters by simulation," *IBM J.*, vol. 1, pp. 8–18; January, 1957.
[7] T. L. Dimond, "Devices for reading handwritten characters," *1957 Proc. EJCC*, pp. 232–237.
[8] "Typed figures translated into computer code," *Engineering*, vol. 183, pp. 348–349; March 15, 1957.

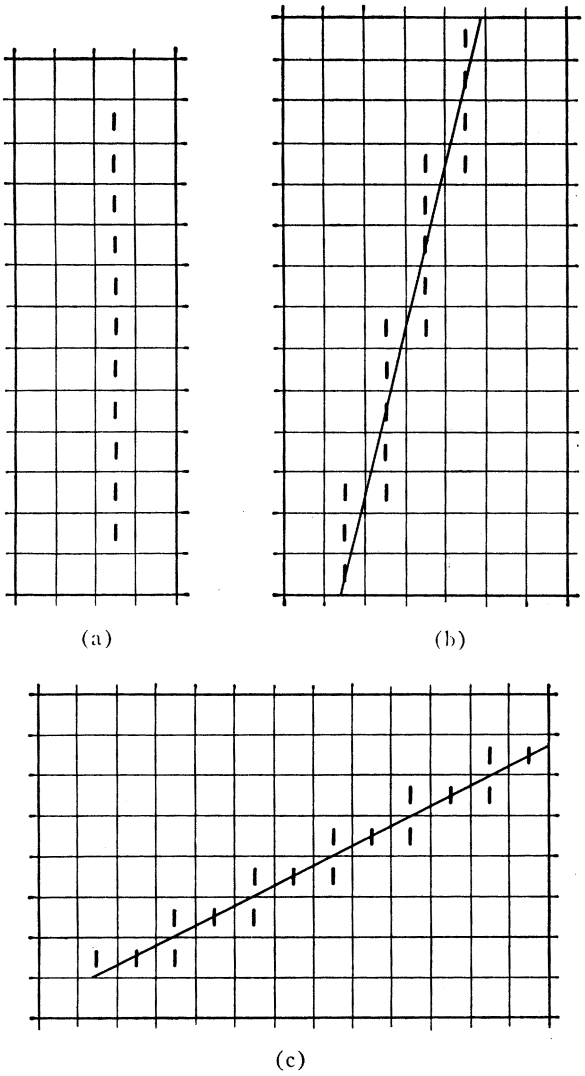


Fig. 2—Mappings of straight lines on a grid.

Consider now the mapping of a straight line whose slope exceeds unity, but is not infinite [Fig. 2(b)]. Such a map must have slope discontinuities, since the edges of the array of squares in the map must be either vertical or horizontal at all points. Thus, a map of a line with a slope anywhere between $+1$ and ∞ consists of a chain of vertical strips, with the top cell of each strip to the left of the bottom cell of the strip above. The numbers of cells in the strips corresponding to a particular straight line differ from one another by at most one (except for the uppermost and lowermost strips which may have fewer cells), and are approximately equal to the slope. Lines with absolute slopes less than one (lines closer to being horizontal) are similarly mapped into chains of overlapping horizontal strips, as shown in Fig. 2(c). In such cases, the number of one-cells per strip approximates the inverse slope of the line.

If a line is not quite straight, its map will still be qualitatively as above. The difference will show up in the form of variations in the lengths of some of the strips.

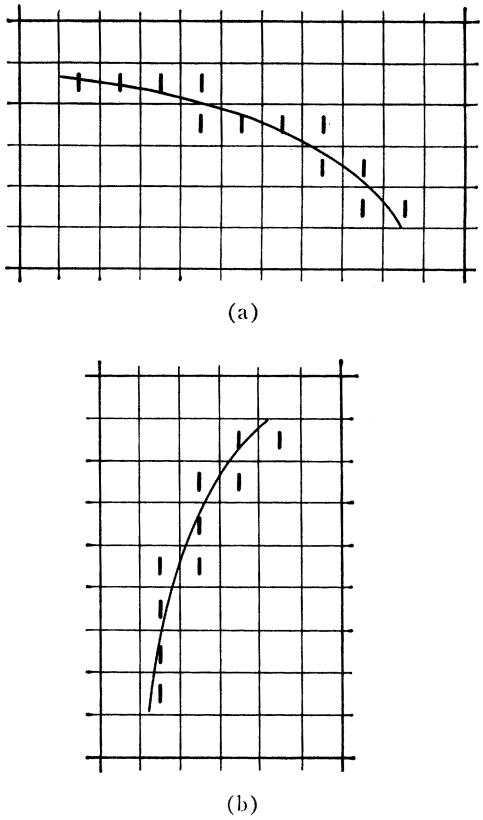


Fig. 3—Mappings of curved lines on a grid.

The mapping of a curved line onto a discrete matrix can be thought of as the union of a set of maps derived from a series of nearly straight line segments approximating the curve. A more precise approach is suggested by an examination of Fig. 3(a), a mapping of a section of a curve that is convex upward. Note that the lengths of the horizontal strips that constitute the mapping decrease monotonically from the top down. In other words, the map consists of a staircase in which every step is as long as, or longer than, the step below.

Fig. 3(b) is a map of a curve convex to the left, with slope between $+\infty$ and $+1$. Here the lengths of the vertical strips decrease monotonically to the right. This monotonicity property can be used to characterize the curvatures of given lines.

III. Smoothing

It is important that any system for pattern detection or recognition be relatively insensitive to minor irregularities in the input fields. Interchanging zeros for ones in a few isolated cells should not cause significant changes in the output of a pattern processing system. As was pointed out previously, some smoothing is achieved merely by quantizing the input. This however is not adequate, and in some cases the quantization itself introduces irregularities.

There are at least two basic approaches to this problem. One is to carry out a preliminary smoothing opera-

tion on the input fields prior to the main task. Another approach is to incorporate the smoothing into the basic recognition or detection operations by making them insensitive to variations confined to any small area. Both of these techniques are used in the programs that will be described here.

The explicit smoothing process,

1) Fills in isolated holes in otherwise black areas,
2) Fills in small notches in straight edge segments,
3) Eliminates isolated ones,
4) Eliminates small bumps along straight edge segments,
5) Replaces missing corner points (under certain conditions).

Fig. 4 shows a cell x, and the eight cells around it. Treat each cell as a binary variable. That is, $b=1$ if cell b contains a one and $b=0$ if cell b contains a zero.

a	b	c
d	x	e
f	g	h

Fig. 4—Points involved in smoothing operations.

The first step in the smoothing operation places a one in x if the Boolean function $f_1=x+bg(d+e)+de(b+g)$ is unity. (That is, x is changed from zero to one if three of the four variables b, d, e, and f are ones.) If this process is carried out simultaneously for every cell in the matrix, then steps 1 and 2 will be accomplished. (In the preceding sentence, "simultaneously" means that the next state of each cell is determined before any of the other cells have been changed.)

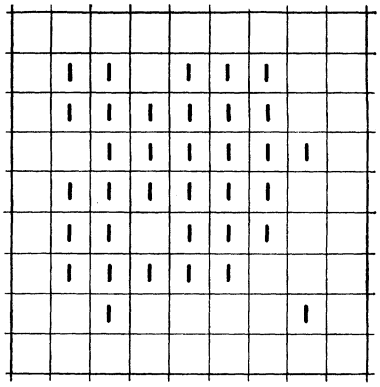
Steps 3 and 4 can be accomplished by replacing the contents of each cell x with $f_2=x[(a+b+d)(e+g+h)+(b+c+e)(d+f+g)]$. A series of four more operations of a similar nature serve to carry out step 5. The smoothing will transform Fig. 5(a) into Fig. 5(b).

About one hundred SPAC orders (each executed once) are required to accomplish the preceding operations.

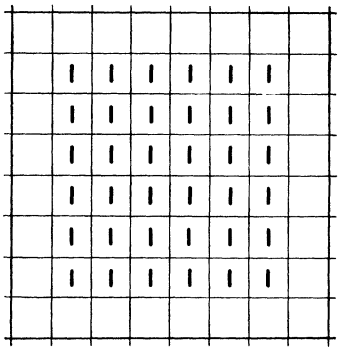
A simple 25-step subroutine which closes breaks in lines, if the gaps are no wider than one unit, can be inserted ahead of the above described smoothing routine.

These programs will clean up only small noise "specks" affecting isolated cells or in some cases pairs of cells. However, smoothing operations for eliminating larger irregularities might be devised by considering larger regions around each cell and exploiting continuity properties.

Note that the smoothing operations discussed thus far are size dependent in an absolute sense. That is, the irregularities are smoothed out on the basis of absolute size, not on the basis of size relative to that of the overall figure.



(a)



(b)

Fig. 5—Effects of smoothing.

This situation is satisfactory only if the figures being processed do not vary in size by more than a factor of two or three in magnitude. If wider variations are to be permitted in the sizes of the input figures, then it would be necessary to insert parameters in the smoothing programs which would be functions of some dimension of the input figure, such as the over-all height. A preliminary part of the program would then measure this dimension and insert appropriate parameters in the smoothing program. These parameters would control the sizes of the irregularities to be smoothed out. All of this is doubteless possible, but it would add considerable complexity to the program. Possibly, the way to avoid the necessity for such an approach would be to use some sort of analog controlled lens system that would keep the sizes of input figures within a range of two or three to one.

IV. Edge Sequences

In order to be able to detect patterns in the sense defined in the introduction, it is necessary to be able to state precisely, for any target set, just what characteristics distinguish this set from all others. For most interesting target sets these characteristics must be independent of size, location, and, to a considerable extent, of the proportions of component parts.

An important property which goes far toward satisfying these requirements will now be introduced.

Consider the letter "L" in Fig. 6(a) (regarding it as a figure covering a non-zero area, not as a pair of intersecting line segments). Beginning at the starred corner and proceeding in a counter-clockwise direction along the edges (boundaries) of the figure, the following succession of edges is found: top, left, bottom, right, top, right. This sequence, abbreviated at TLBRTR (T for top, L for left, etc.), is cyclic, and if another starting point is used, then an equivalent form such as BRTRTL will be found. Such a sequence, which will be referred to as an *edge sequence*, is not affected by changes in size, proportion, or location.

The edge sequence of Fig. 6(b) is LBR (BL) B (TR), where (BL) and (TR) represent diagonal edges facing "bottom-leftward" and "top-rightward" respectively. Similar notation can be devised for curved edges.

In the case of a figure with a hole, such as is shown in Fig. 6(c), two edge sequences are required for a description; one for the outside, and one for the inside. For this case they are LBRT and R (TL) B, respectively.

Edge sequences constitute a powerful tool for pattern detection. As will be shown in a later section, the ideas concerning edges that were introduced earlier can be utilized by a machine to find the figures in a given input field that have edge sequences corresponding to the target set.

Edge sequences alone are not in general sufficient to completely characterize many interesting target sets. A particular edge sequence may, for example, be a necessary condition for membership in some target set, but there may be other requirements as well. Consider for example Fig. 7(a). This figure has the same edge sequence as a certain alphabetic character drawn in block capital form without serifs. But few readers would classify this figure as an H, although both (a) and (b) of Fig. 7 share the same edge sequence. An additional constraint that must be satisfied before a figure can be classified with 7(b) as an "H" is that the edge labelled p in Fig. 7(b) be directly above the edge labelled q. This condition is not satisfied by the corresponding edges in Fig. 7(a). Such "relative-edge-position" tests are frequently necessary after the edge sequence requirement has been met.

Questions of proportion may also arise. For example, it would be reasonable to exclude Fig. 8(a) from the H-set, while admitting Fig. 8(b). Both have the appropriate edge sequences and their edges are in the proper relation to one another. It is difficult to specify precisely which figures intermediate between the pair in Fig. 8 should be considered as acceptable H's. However, an arbitrary set of standards may be chosen for each target set, such as the requirement (as applied to the present example): "the width of the wider leg must not exceed twice the width of the narrower leg." The research reported on here has not yielded any more specific results concerning this aspect of the problem.

Still another question is posed by Fig. 9(a). Is this a respectable member of the H-set previously discussed? At first, the results of a lifetime of conditioning would lead one to immediately answer "yes." But, on second thought, it becomes obvious that the form of Fig. 9(a) is certainly significantly different from that of a block capital H without serifs, which is the target set under discussion [see Fig. 7(b)]. Only long established custom decrees that this is also an "H," and it would be perfectly possible to devise an alphabet in which the two forms represented different letters. The difference between Fig. 9(b) and Fig. 9(c) is of the same nature, and is smaller in magnitude than the difference between the two H's, and yet one is called a "C," and the other a "G."

Thus, it may be convenient to think of certain target sets as being *compound;* that is to say, of being composed of the union of two or more simpler sets. Just as it would not be wise to expect a single description to suffice for both a small "h" and a capital "H," it is similarly unreasonable to expect the same description to cover different styles of capital "H"'s.

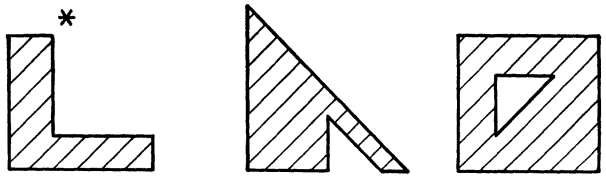


Fig. 6—Illustrating the use of edge sequences.

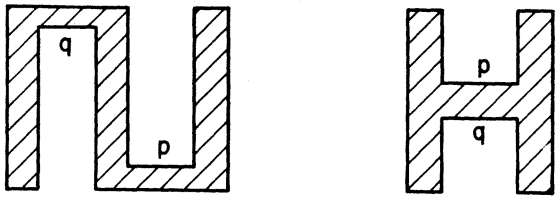


Fig. 7—Limitations of edge sequence description.

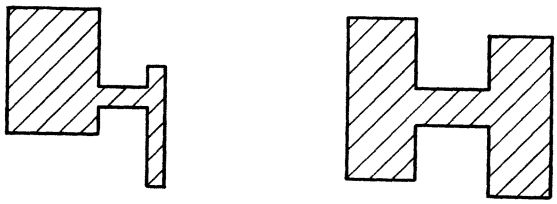


Fig. 8—Variations in proportion.

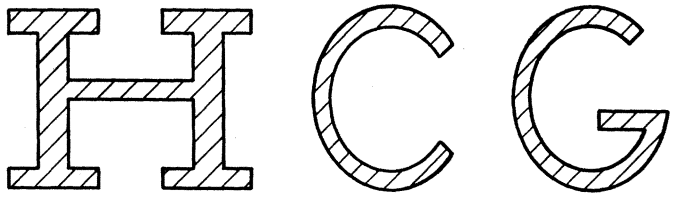


Fig. 9—Serifs.

Note that this argument holds only for the detection problem. Where pattern recognition is involved, it may well be feasible to devise (for example) a single scheme which can indicate that both forms of the capital "H" are sufficiently different from all other members of the alphabet as to be definitely classed as "H"'s.

V. Pattern Detection

In this section it will be shown how the ideas previously introduced can be utilized for the purposes of pattern detection. A SPAC program for detecting L-shaped figures will be presented and discussed in order to illustrate the principles involved. (The Appendix and an article mentioned previously[1] describe SPAC.)

The L-detection program has been executed successfully on an IBM 704 computer programmed to simulate SPAC. In order to make perfectly clear what the L-

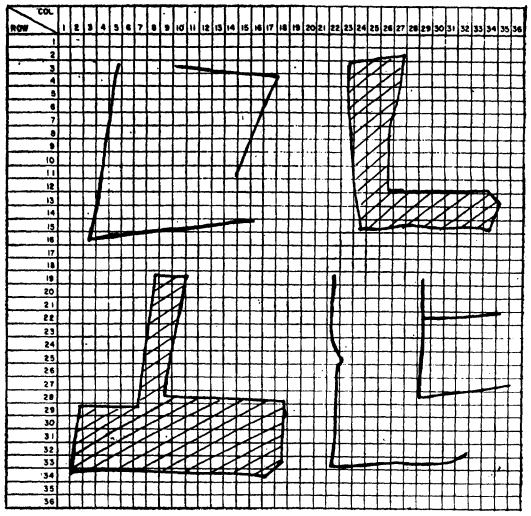


Fig. 10—Original drawing.

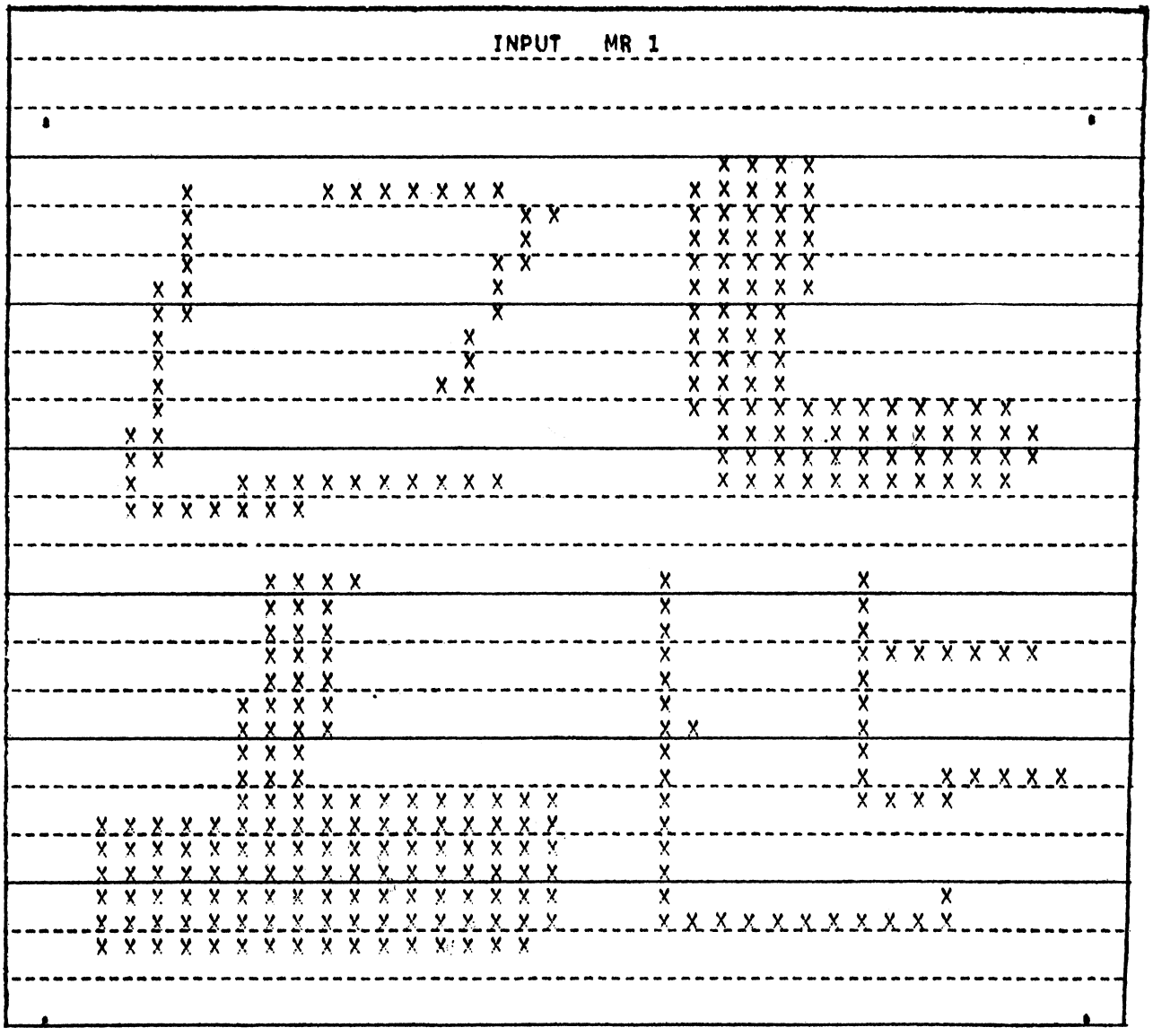


Fig. 11—Input to SPAC L-detection program.

detection program does, the results of one of the experimental 704 runs will be presented now.

The input to the system is a set of figures drawn freehand on a sheet of coordinate paper (Fig. 10). Since a suitable automatic scanning device was not yet available, its action was simulated by a key-punch operator, who was instructed to punch a set of IBM cards to represent the pattern; one column corresponding to each cell of the coordinate paper. A one was punched where the corresponding cell was black and a blank was punched where the corresponding cell was blank. The resulting deck of cards served as the input to the machine. (As can be seen in Fig. 10, a 36×36 array was used.) A print-out of this quantized input is shown in Fig. 11, with x's replacing ones in order to make the form of the figures more conspicuous.

The first step by SPAC was to smooth the input field using a routine carrying out the steps described in the section on smoothing. Details of the smoothing routine will not be presented here.

SPAC then executed the L-detection program, which will be discussed next, and then printed out the result shown in Fig. 12, in which only L-shaped figures (after smoothing) remain.

The program first selects those figures whose edge sequences include LBRTRT, where the first R-edge and the second T-edge can be arbitrarily small, while the other four edges must each be at least four units long. (This permits the detection of "thin" as well as "thick" L's.) From this set, the program eliminates all figures with any edges not in the LBRTRT sequence, leaving figures such as those shown in Fig. 6(b) and Fig. 13(a).

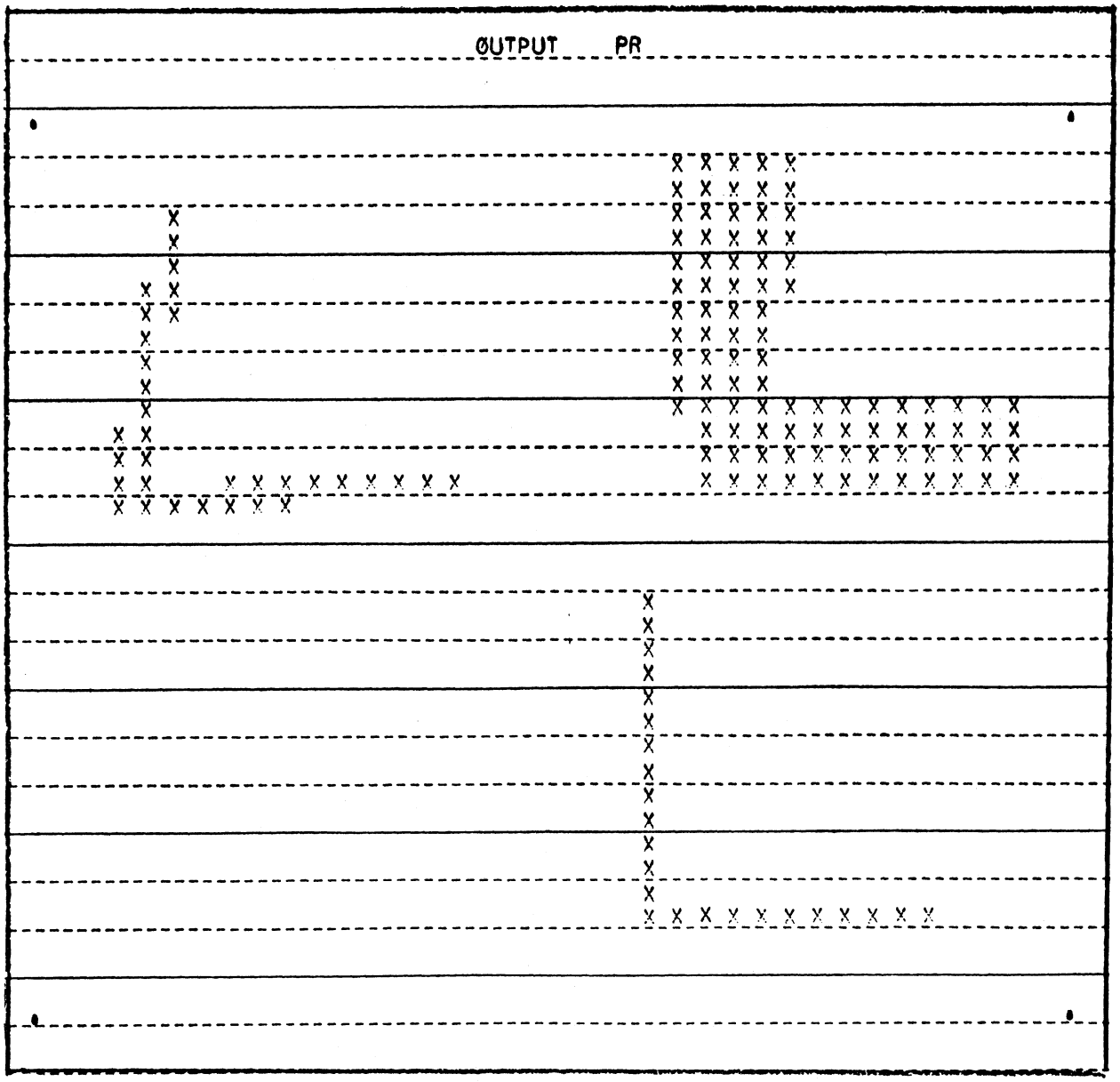
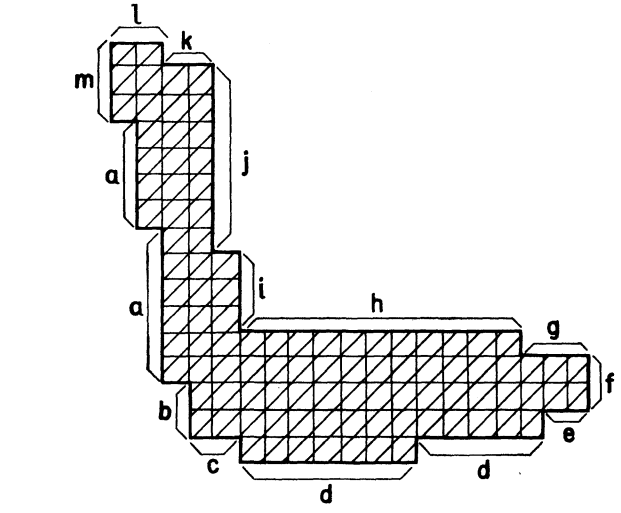


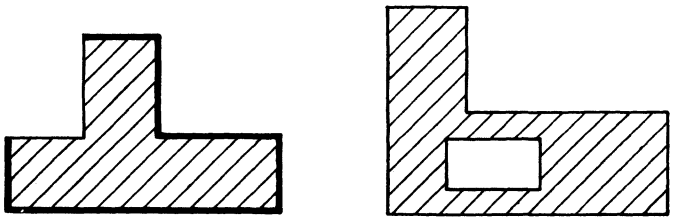
Fig. 12—SPAC output.

The following explanatory notes correspond to the parenthetically numbered points in the program shown in Table I, and the references to lettered edges in these notes pertain to Fig. 13(a).

(1) Left edge points (one-points with zeros to the left) which are in vertical strips of length at least equal to 4 (a-edges).
(2) Adds b-edges (vertical strips of left edge points diagonally below a-edges) to the a-edges found in (1).
(3) Corner points such as the one common to b and c.
(4) MR5 now includes edges a, b, c, and d.
(5) Adds edge e to MR5.
(6) Adds edge f to the edges already in MR5. Note that whereas the left edge of the L, consisting of m, a, and b-edges, must include an a-edge which is at least 4 units long, the portion of the L corresponding to f can be as small as one unit.
(7) Adds g-edge to MR5.
(8) Adds edge h to MR5.
(9) Edge i is added to MR5. The inside corner was "turned" 10 instructions prior to this order by the SHF L and SHF U instructions.
(10)–(13) Edges j, k, and l, are added to MR5.
(14) The PR now contains edge m (which is now also in MR3).
(15) MR2 now contains all edges which are in LBRTR sequences (or subsequences) and *only* such edges.
(16) MR3 now contains the edges found in (15) and the regions enclosed by these edges. In the case of Fig. 13(a), MR3 contains the entire figure. Since only the darkened edges of Fig. 13(b) comprise LBRTRT sequences, MR3 will contain only that part of this figure which is in MR2 (the darkened edges—which enclose no area). MR3 will contain Fig. 13(c) with the hole filled in. Thus, only in the case of true L's such as Fig. 13(a) will MR3 include the whole figure without additions. The remainder of the program consists of eliminating figures which are only partially contained in MR3 [Fig. 13(b)] and figures such as Fig. 13(c) which do not completely cover their counterparts in MR3.
(17) Figures such as 13(c) which have holes.
(18) PR now contains figures such as 12(b) with outer edge sequences including edges other than those in MR2.
(19) Figures found in (17) and (18) are eliminated, leaving only the desired L-shaped figures.



(a)



(b) (c)

Fig. 13—Illustrating L-detection program.

TABLE I
L-DETECTION PROGRAM

SMOOTH	WRT 3	STR 3
WRT 1	SHF U	WRT 1
SHF R	ADD L, R	SHF D
INV	MPY 4	INV
MPY 1	EXP V, P, N	MPY 1
STR 2	ADD 3	STR 2
LNK	ADM 5 (6)	LNK
MPY U	SHF U	WRT 1
MPY U	ADD L, R	SHF L
MPY U	ADM 3	SHF D
ADD D	WRT 2	INV
ADD D	LNK	MPY 2, 3
ADD D	MPY 3	EXP H
STR 5 (1)	EXP V	STR 3
SHF D	ADM 5	WRT 2
ADD L, R	STR 3	MPY R
MPY 2	WRT 1	MPY R
EXP V	SHF D	MPY R
STR 3	INV	ADD L
ADM 5 (2)	MPY 1	ADD L
WRT 1	STR 2	ADD L
SHF U	LNK	LNK
INV	WRT 1	STR 4
MPY 1	SHF L	WRT 3
STR 2	SHF D	SHF L
LNK	INV	ADD U, D
MPM 3	MPY 2, 3	MPY 4
WRT 1	EXP H	EXP H, P, N
SHF R	ADM 5 (7)	ADD 3
SHF U	SHF L	ADM 5 (12)
INV	ADD U, D	SHF L
MPY 3 (3)	STR 3	ADD U, D
EXP H	WRT 2	ADM 3
ADM 5	MPY L	WRT 2
SHF R	MPY L	LNK
ADD U, D	MPY L	MPY 3
STR 3	ADD R	EXP H
WRT 2	ADD R	ADM 5 (13)
MPY R	ADD R	STR 3
MPY R	LNK	WRT 1
MPY R	MPY 3	SHF R
ADD L	EXP H, P, N	INV
ADD L	ADM 5 (8)	MPY 1
ADD L	SHF L	LNK
LNK	ADD U, D	STR 2
MPY 3	STR 3	WRT 1
EXP HPN	WRT 2	SHF R
ADM 5 (4)	LNK	SHF D
SHF R	MPY 3	INV
ADD U, D	EXP H	MPY 2, 3
STR 3	ADM 5	EXP V
WRT 2	SHF L	STR 3 (14)
LNK	SHF U	ADD 5
MPY 3	STR 3	LNK
EXP H	WRT 1	WRT 3
STR 3	SHF L	EXP H, V, P, N
ADM 5 (5)	INV	STR 2 (15)
WRT 1	MPY 1	INV
SHF L	STR 2	LNK
INV	LNK	MPY 2
MPY 1	MPY 3	ADR
STR 2	EXP V	EXP H, V
LNK	ADM 5 (9)	INV
WRT 1	SHF U	STR 3 (16)
SHF L	ADD L, R	LNK
SHF U	STR 3	WRT 1
INV	WRT 4	INV
MPY 2, 3	LNK	MPY 3
EXP V	MPY 3	EXP H, V, P, N
STR 3	EXP V, P, N	STR 4 (17)
WRT 2	ADM 5 (10)	WRT 1
MPY D	SHF U	LNK
MPY D	ADD L, R	WRT 3
MPY D	STR 3	INV
ADD U	WRT 2	MPY 1 (18)
ADD U	LNK	ADD 4
ADD U	MPY 3	EXP H, V, P, N
LNK	EXP V	INV
STR 4	ADM 5 (11)	MPY 1 (19)

Note that there are no loops in the program, so that each of the 237 SPAC instructions is executed exactly once.

Programs for detecting figures with holes are somewhat more complex. Both inner and outer edge sequences are traced, and, in place of the contents of MR3 described in (16) it is necessary to find the area enclosed between the outer and inner borders. This area is then used in the same manner as the MR3 contents were used in the L-program. However, it is then necessary to eliminate figures with the wrong *number* of holes, or with improperly located holes. This can be done in a variety of ways. One method is to eliminate points between one outer edge of the figure and the hole(s) and then check the edge sequence of the resulting figure, which no longer has any holes.

Figs. 14, 15 and 16 show the results of a test of an A-

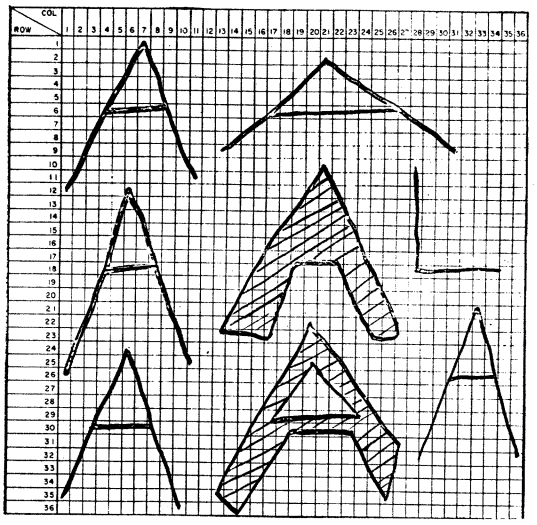


Fig. 14—Original drawing.

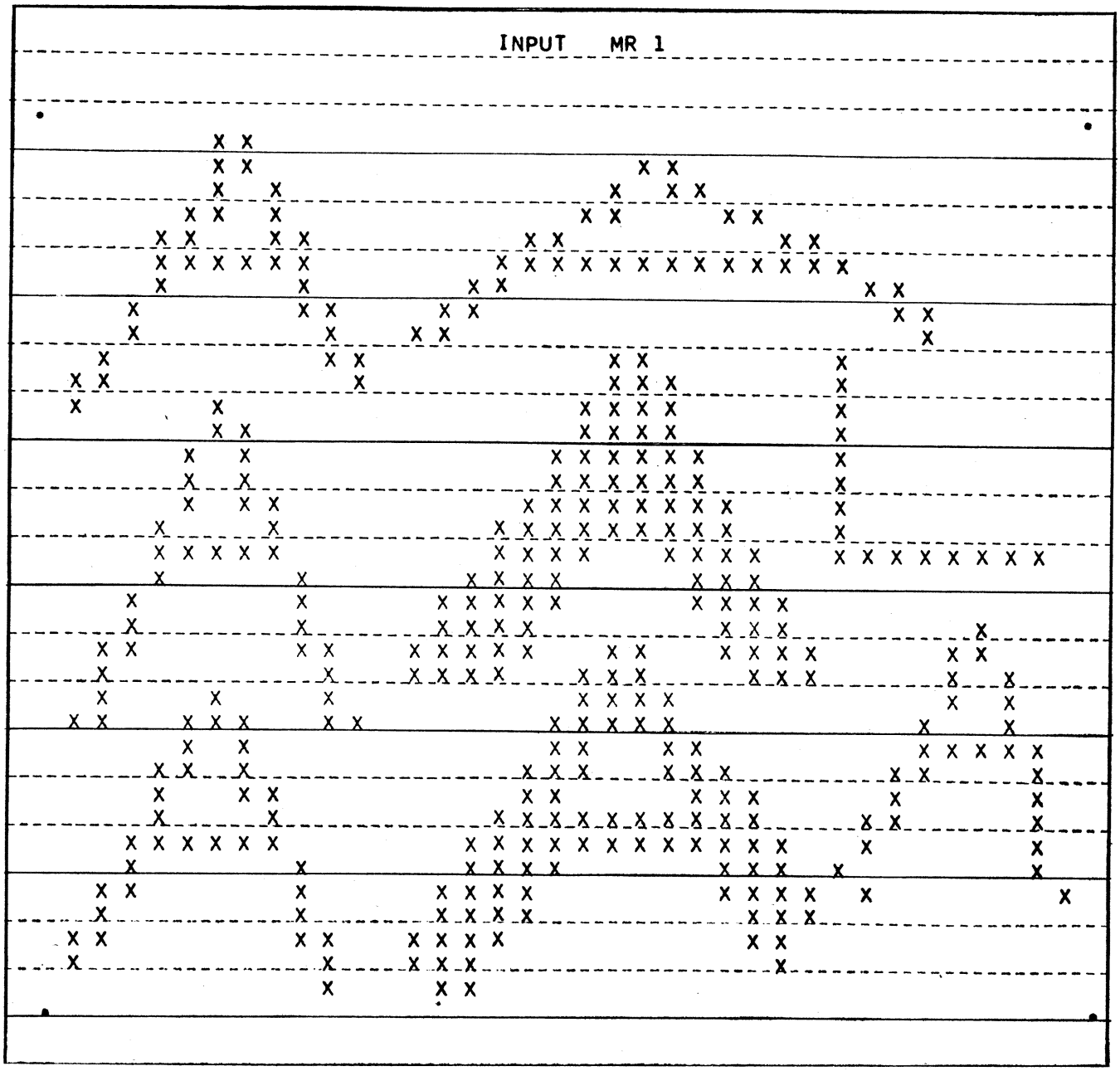


Fig. 15—Input to SPAC L-detection program.

detection program consisting of about 500 orders (including smoothing).

VI. Pattern Recognition

The pattern recognition problem defined earlier in this paper consists of identifying an input figure, given that it belongs to some member of a finite set of categories. For example, the set of categories (henceforth to be referred to as the *alphabet*) might consist of the digits from zero through nine inclusive.

For any alphabet there must exist at least one finite set of characteristics that can be used to distinguish amongst the members. That is, there must exist a set of yes or no questions, such that if these questions are answered with respect to any given figure, then there will be only one member of the alphabet to which this figure can belong. The questions might, for example, concern the presence or absence of such features as holes, straight left edges, or concavities open to the right.

It is not necessary to ask all of the questions in every case. A sequential process can be carried out in which the answer to the first question determines a subset of the alphabet to which the input figure may belong. Depending upon the subset, a second question is asked, the answer to which further narrows the field of possible categories. This process is repeated until a single candidate remains. If the questions can be so chosen that each one halves the number of remaining candidates, then the number of questions necessary to identify any given figure is about $\log_2 n$, where n is the number of members of the alphabet. This is the minimum average

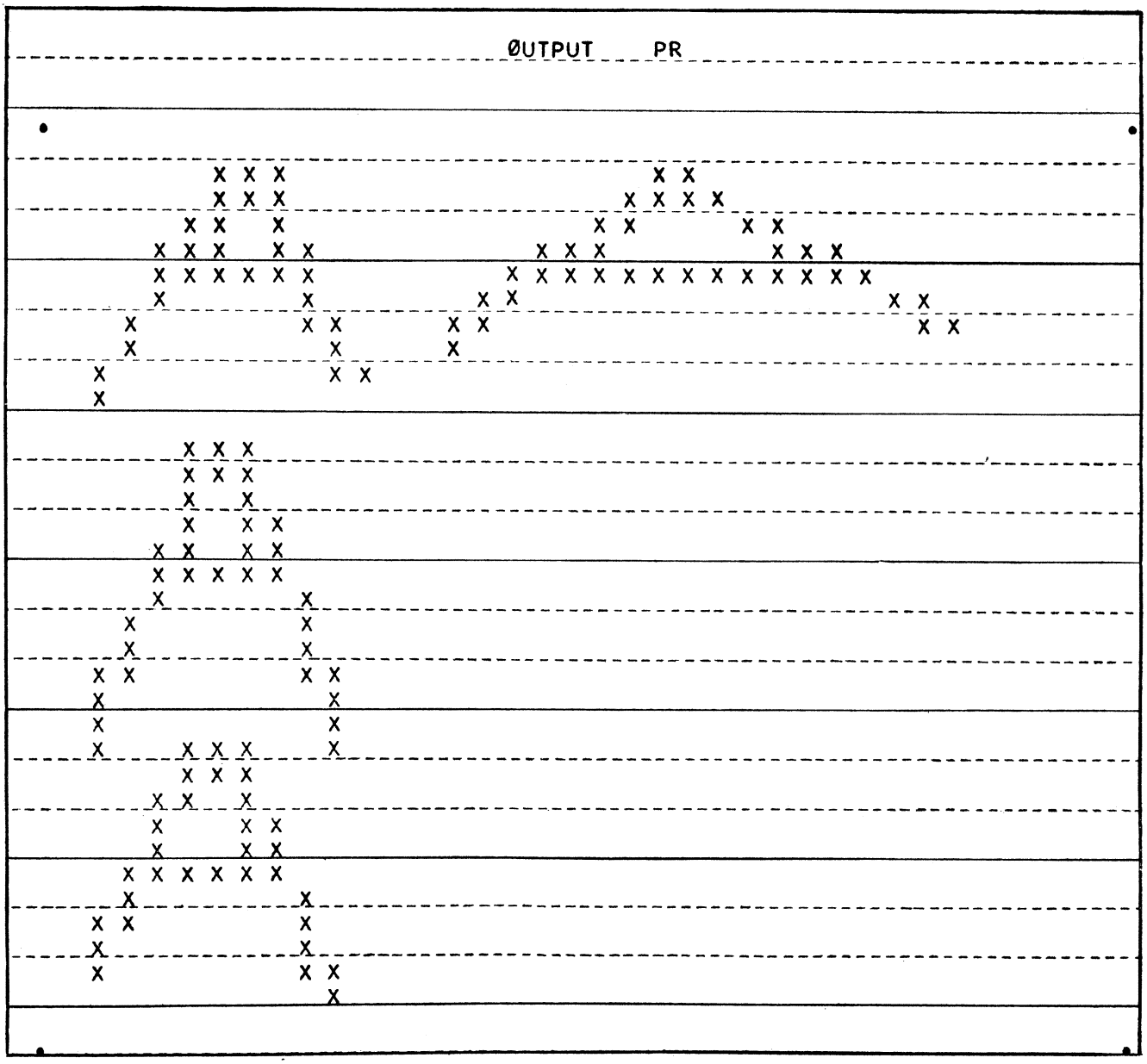


Fig. 16—SPAC output.

number of questions, assuming that the input figure is equally likely to belong to any of the n categories.

A good set of questions should have the following characteristics (insofar as they are mutually compatible):

1) They should be sufficient to distinguish any pair of allowable input patterns that belong to different members of the alphabet.
2) The necessary number of tests that must be performed for each identification should be minimal.
3) The number of orders necessary to perform each test should be minimal.
4) The total number of tests should be minimal.

An ideal property is thus easy to test for, possessed by half of the members of the alphabet (each test must produce a *different* dichotomy), and invariant with respect to all distortions that leave a pattern in the same category within the alphabet. Needless to say, it is not generally possible to attain such an ideal.

A pattern recognition scheme for alphanumeric characters will now be described. It is intended to recognize hand drawn sans-serif capital letters and numerals with very broad limits on the permissible over-all sizes and thicknesses of the input figures. A preliminary smoothing routine as described in Section III serves to cancel out the effects of noise, provided that the noise is not concentrated in any one locale.

The 34 properties described in Table II are used to distinguish among the 34 members of the alphabet (ones do not differ from I's, and zeroes do not differ from O's). Fig. 17 is a flow chart indicating how the tests are used. Starting with property 2 at the top, movement is downward. At each branch point the left branch is taken if the answer to the question referred to by the number is no. The right branch is taken for a yes answer. The following two definitions pertain to Table II:

1) A figure is *vertically concave* if there exist points not on the figure whose vertical projections *both* upward and downward touch the figure. Such points constitute *vertical cavities*. Similar definitions apply to horizontally concave figures, and to $+45°$ and $-45°$ concave figures. Parts (a), (b), (c), and (d) of Fig. 18 illustrate vertical, horizontal, $-45°$ and $+45°$ concave figures respectively, the dotted regions indicating the respective cavities. (This definition will be further qualified later on in the text.)
2) A cavity is *open to the right* if a directed path can be found starting at a point in the cavity, passing through points not on the figure, with no component in a leftward direction, and terminating at a point on the margin of the matrix. Cavities open to the left, or in other specified directions, are similarly defined.

TABLE II

FEATURES USED IN CHARACTER RECOGNITION PROGRAM

1) Horizontal cavity open above, (begj).
2) Vertical cavity open to the right, (aghk).
3) Vertical cavity open to the left, (efghl).
4) Horizontal cavity open below, (fhk).
5) Horizontal cavity, (befghijkl).
6) A hole, (eil).
7) Two vertical cavities lying on a common vertical line with the region between them consisting of points on the figure. For at least part of each cavity, no points on the figure are directly to the left of the cavity. (This latter requirement is a somewhat stricter version of "open to the left"), (e).
8) $+45°$ concave, (defghijkl).
9) A region such that a vertical line segment can be drawn starting on the figure, passing down through a vertical cavity, passing through points on the figure again, then (still moving down) entering another vertical cavity, and finally terminating on the figure, (efghl).
10) Not used.
11) Same as 9), but with a horizontal instead of vertical orientation, (ijk).
12) Vertical cavity, (aefghikl).
13) $-45°$ cavity, (cefghijkl).
14) $-45°$ cavity open in a "left-bottomward" direction, (efghkl).
15) Horizontal cavity not part of a hole, (befghjk).
16) $+45°$ cavity not part of a hole, (defghjkl).
17) Holes whose inverse (complement) is horizontally concave, (i).
18) -45 cavity not part of a hole, (cefghjkl).
19) A vertical left edge with right-angled corners above and below, (acdefjk).
20) Two holes, (i).
21) A horizontal top edge with inside corners on both ends, (ej).
22) Same as 7) except that the cavities are horizontal and they face upward, (j).
23) Same as 7) except that the cavities are horizontal and they face downward, (k).
24) A vertical cavity, the lower end of which is above a bottom edge connected to an upper-left inside corner, (the intersections of a bottom edge and a right edge), (ek).
25) A cavity which is both vertical and horizontal, (efghil).
26) Hole whose inverse is vertically concave, (e).
27) Same as 9) except that the lower vertical cavity is inside a hole, (el).
28) Leftmost point of a vertical cavity open to right higher than the rightmost point of a vertical cavity open to the left, (g).
29) Right vertical edge, (bcdefijk).
30) Not used.
31) Height of the left leg of a U-shaped figure less than half the height of the right leg. (Used only to distinguish a "J" from a "U".)
32) Vertical cavity open to the left. This differs from 3 in that it includes cases where the boundaries of the cavity have nearly vertical slopes. (A fuller explanation of this distinction will be presented in the text.)
33) A right vertical edge with no horizontal cavity on the same level with it. (A Y has this property—but not a V.)
34) Leftmost point of a vertical cavity open to the right lies in the upper two thirds of the figure.
35) Vertical cavity open to right. (See 32.)
36) Horizontal cavity above a vertical cavity separated by part of the figure (efh).

Some of the descriptions in the table are of a qualitative nature, since more detailed statements would be beyond the scope of this paper. Following the description of each feature in the table is a parenthetic note referring to just those parts of Fig. 18 which have that feature. It is hoped that these illustrations will serve to clarify some of the more complex descriptions.

Before discussing some of the interesting features of this problem and our solution let us see how the system would operate in a particular case. Suppose, for example, that the figure to be identified happens to be an

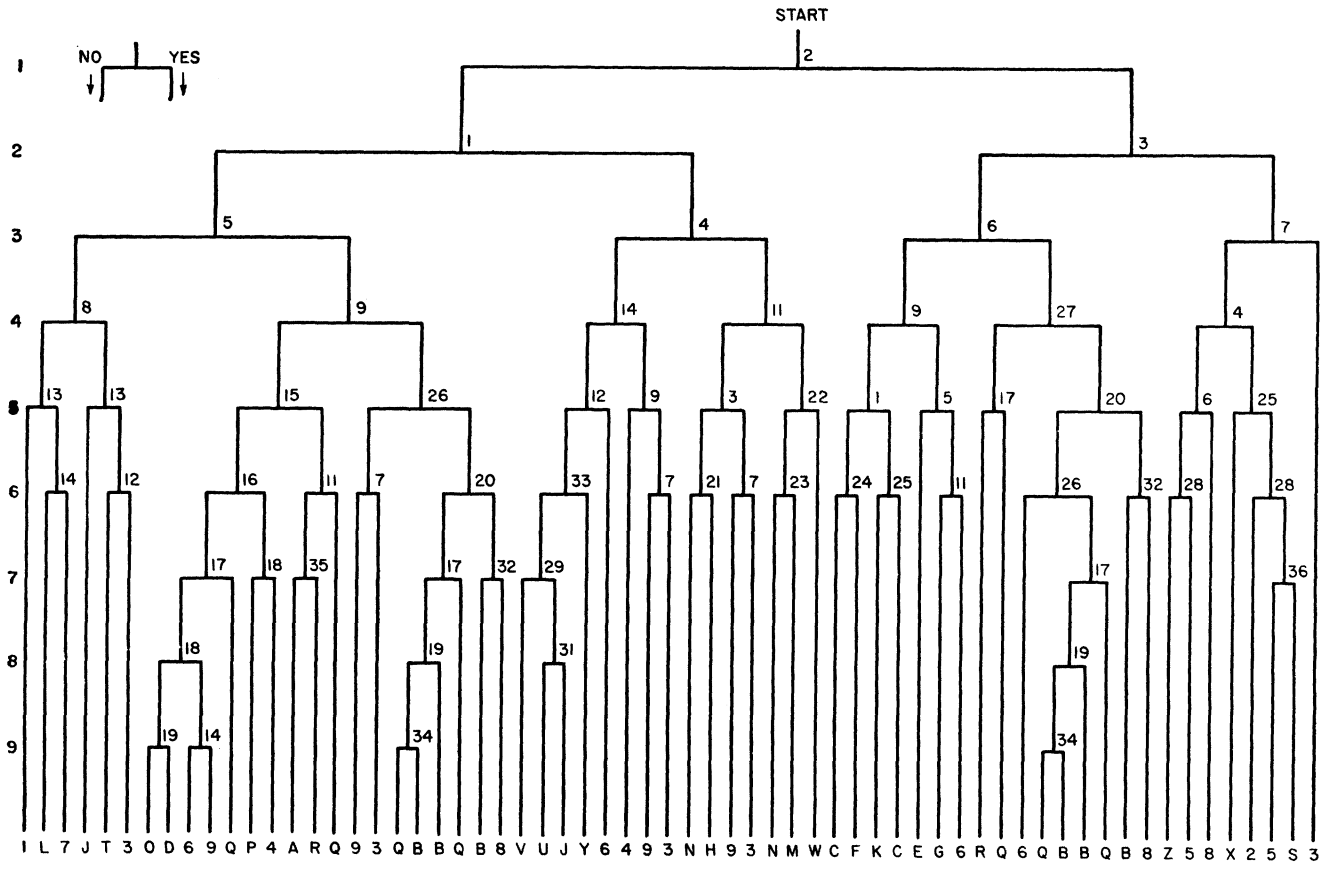


Fig. 17—Flow chart for recognition program.

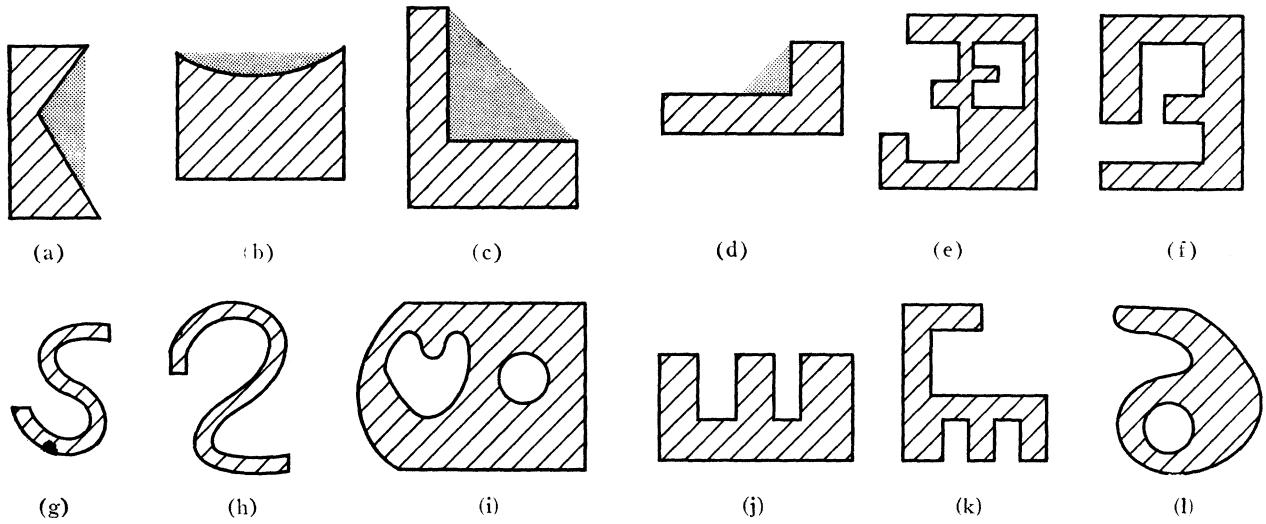
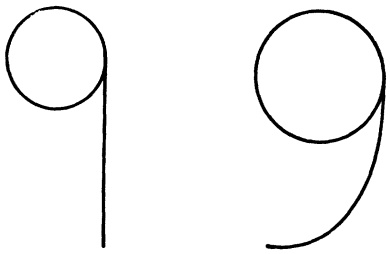


Fig. 18—Illustrations for recognition program features.

"E." Referring to Fig. 18 we see that the first question (2) tests whether there is a concavity open to the right. Since the answer is yes, the right branch is taken, and the next question is 3, "Is there a concavity open to the left?" The answer being no, the left branch is taken and the next question is 6. Questions 9 and 5 follow in succession and a negative answer to the latter leads to the conclusion that the input is an "E."

In order to allow for variations in the forms of certain characters, these characters (such as "3" and "Q") appear at several different terminal points of the flow chart. A "9" for instance may be successfully recognized as drawn in either (a) or (b) of Fig. 19. Thus there are 63 termination points for the 34 characters.

Tests for each characteristic listed in Table II can be carried out by means of subroutines for SPAC. An average of about forty-five orders are required for each test. Thus, roughly three to five hundred orders are re-



(a) (b)

Fig. 19—Illustrating allowable style variations.

quired for each input figure (including one hundred orders to carry out the smoothing process described in Section III). There are about 1600 orders in the entire program, which has been successfully tested by means of the simulated SPAC. A lack of space precludes the presentation of the entire program in this paper, but a representative subroutine will be included.

The subroutine for each test consists of a series of operations carried out on the given figure so that at the end of the routine the field is cleared unless the figure has the property being tested for. Table III consists of a portion of a calling sequence that realizes the first few levels of the flow chart of Fig. 17. The order SBR XXX calls in the subroutine XXX. (For example, to test for property 2 we use the order SBR RO2.) After this subroutine has been executed the next order in the sequence is carried out. The order JMP PRY causes the computer to print a statment to the effect that the input character is a Y.

TABLE III

SBR RO2	TRZ 3
TRZ 1	JMP PR3
SBR RO3	TRA 60 (where a stop order is located)
TRZ 2	1 SBR RO1
SBR RO7	TRZ 4 etc.

An example of a SPAC program for carrying out a typical test is shown in Table IV.

TABLE IV

SPAC SUBROUTINE FOR TEST 9 (TABLE II)

WRT 1	MPY 9	ADD U
ADD L, R	EXP V (3)	STR 7
INV	ADD U	WRT 9
STR 9 (1)	SHF D	LNK
LNK	MPY 1 (4)	MPY 7
WRT 1	STR 7	EXP V
INV	WRT 1	ADD U
STR 8	LNK	SHF D
WRT 1	WRT 7	MPY 1 (6)
SHF D	EXP V	STP
MPY 8 (2)	SHF D	
ADD U	MPY 8 (5)	

The following explanatory notes refer to the numbered instructions in the program.

(1) Zeros, with zeros to the left and right of them.
(2) Points just below bottom edges.
(3) Zeros with upward projections touching ones.
(4) Zeros just below parts of the figure which are below vertical cavities.
(5) Zeros just below parts of the figure which are below vertical cavities.
(6) Ones below two vertical cavities, as required in Table II.

Note that the use of the points obtained in (1) as the possible vertical cavity points prevents a slightly crooked vertical line from being considered as vertically concave.

It will not be possible to discuss in detail all of the problems that lead to the use of the various tests employed in the system described here, but a few samples may serve to give the general flavor.

Consider first the test for a vertical cavity open to the right (test 2). A strict interpretation of this test would mean that a positive result would be obtained in cases typified by a "T" whose vertical member is slanted as shown in Fig. 20. In order to prevent this,

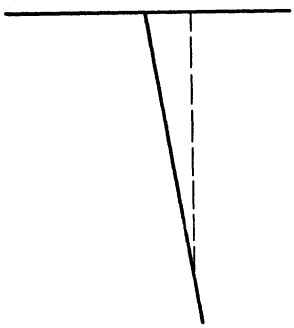


Fig. 20—Illustrating test 2.

the program for test 2 is written so that necessary conditions for a point to be considered as being in a vertical cavity are:

1) It must not be horizontally adjacent to a one-cell;
2) All zero-points in the cavity that are in the same column as the given point must meet condition 1, except possibly those at the top and bottom of the cavity.

These requirements insure that parts of the upper and lower borders of a vertical cavity must both be nearly horizontal. Hence Fig. 18(a) does *not* possess feature 2 as indicated in Table II. In certain cases it may be desirable to detect the presence of vertical cavities whose boundaries do not meet condition 2. Such a cavity is shown in Fig. 21. Test 35 accomplishes this function, in that it does not require the upper and lower cavity boundaries to be nearly horizontal. This characteristic is useful in distinguishing R's from A's.

In order to distinguish Y's from V's, it was found necessary to look for a right vertical edge below the V-shaped notch at the top of the figure. This is property 33 in Table II. A right vertical edge in this case is defined as an edge containing either one vertical strip of

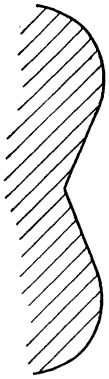


Fig. 21—Illustrating need for test 35.

length at least equal to six or two vertical strips each of length at least equal to three.

The recognition program is preceded by the smoothing routine described earlier in this paper, which eliminates small irregularities. In addition, the various subroutines for detecting features are generally insensitive to fine details, so that the over-all system will operate fairly well in the presence of scattered noise. Fig. 22 is a photograph of a 704 print-out which consists of an input figure and the output statement produced by the recognition program. Such tests have been carried out on about 90 figures. The clumsiness of the procedure for generating input data has precluded more exhaustive experimentation, but the success of the program has been clearly established.

VII. Conclusion

Methods applicable to two classes of problems, pattern recognition and pattern detection, have been developed. Solutions to several specific problems in these areas have been described, namely the recognition of alphanumeric characters, and the detection of L-shaped patterns. The methods and solutions presented here have all been successfully tested by means of a 704 simulation program, and some of the results of these tests have been included.

The matrix size of 36×36 used in these experiments was chosen for its convenience in simulating on the 704, and because it is a reasonably satisfactory size for the complexity of the patterns used in the tests. For applications in which fine details of more complex figures must be studied, a larger matrix will be necessary. A smaller matrix might suffice for simpler applications, although for small matrices the permissible variation in the size of the input image is quite limited, and an important advantage of the techniques described here is lost. On the other hand, as was mentioned in Section II, a matrix that is not too fine will smooth out some of the random irregularities in the input. Of course the principal reason for limiting the matrix size is economic.

Since SPAC does not yet exist as hardware, it is not possible to give actual execution times for the programs included here. However, in view of the simplicity of the logic required for each SPAC order,[1] and the speed of components now in the development stage, it is not unreasonable to assume that, within five years, a speed of one microsecond per order will be attainable. Under this assumption, the character recognition program given here could operate at a rate of about 2500 characters per second (ignoring input time). The time required by the 704 simulation program to execute each SPAC order varies widely as a function of both the data and the instruction. A rough estimate of the time for a "typical" program is 10 msec per order.

In addition to being a fascinating field of study, automatic pattern processing has numerous conceivable applications. Primary interest today is in devices for character recognition, and it is easy to list applications such as transmitting hand written programs and input data to computers without the need for key punch operators, automatic sorting of mail, machine reading of telephone toll tickets written by operators, and machine reading of texts to be automatically translated into other languages.

Pattern detection methods might be useful in biological studies where searches must be made for particular cell structures, bacteria, or viruses among large numbers of microphotographs.[9] Organic chemistry is another field in which important data exist in pictorial form, namely molecular diagrams. Searches for particular kinds of molecular structures among large numbers of records (perhaps in a patent office) may well be facilitated by machines.[10] No doubt other applications will be found as the techniques improve.

VIII. Appendix

The Order Structure of the Spatial Computer (SPAC)

The order structure of SPAC is summarized in Table V, and a brief amplification of some of the descriptions follows. A more complete treatment will be found in a paper mentioned earlier.[1] (Note that some of the abbreviations in this paper will differ slightly from those used in the original description of SPAC.[1])

Suppose, for example, that the order ADD 1, 4 is given. Then, within every module, simultaneously, the contents of MR1 and MR4 will be added to the PR contents (Boolean additions). Note that each module operates on the data in its own memory cells, independently of what is going on in the other modules. The order ADD L, 2 results in the contents of the PR of each module being incremented (again Boolean addition) by the contents of MR2 and the *original* contents of the PR of the neighboring module to the left. Note that ADD L, 2 is not equivalent to ADD 2 followed by ADD L. Other references to neighboring modules used

[9] H. P. Mansberg, "Automatic particle and bacterial colony counter," *Science*, pp. 823–827; October 25, 1957.

[10] L. C. Ray, R. A. Kirsch, "Finding chemical records by digital computers," *Science*, pp. 814–819; October 25, 1957.

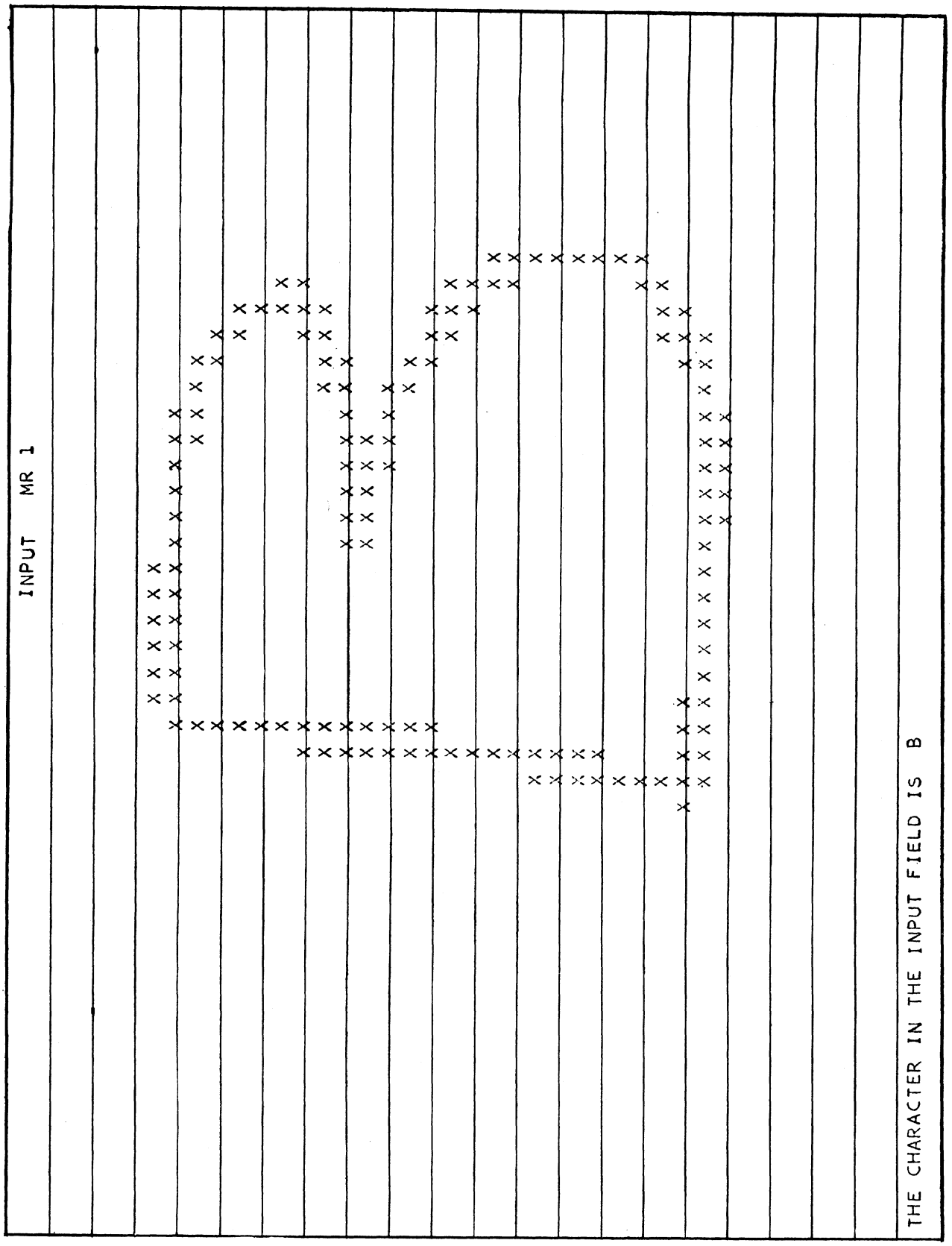


Fig. 22—Example of character recognition.

TABLE V
SPAC ORDER STRUCTURE

Order	Symbol	Interpretation
Write	WRT	Write the contents of the indicated MR into the PR without altering the MR contents.
Store	STR	Store the contents of the PR in each of the specified memory cells without altering the PR contents.
Invert	INV	Change the contents of every PR.
Add in memory	ADM	Add logically to the contents of each specified MR, the contents of the PR, without changing the latter.
Multiply in memory	MPM	Multiply logically the contents of each specified MR by the contents of the PR, leaving the PR contents undisturbed.
Add	ADD	Add logically to the contents of each PR, the contents of the specified MR's and neighboring PR's.
Multiply	MPY	Multiply logically the contents of the PR, the specified MR's and neighboring PR's, placing the result in the PR.
Add Reference	ADR	Add a one to the PR of the module in the lower left corner of the matrix.
Shift	SHF	Write the contents of each PR in the contents of the PR of the module in the specified direction (L, R, U, or D).
Transfer	TRA	Execute the instruction at the indicated address and then continue from there in sequence.
Transfer if zero	TRZ	If every PR contains a zero, then TRA to the indicated address. Otherwise continue with next order.
Link	LNK	Activate link elements between those adjacent pairs of modules in which both PR's contain ones. De-activate all other link elements.
Expand	EXP	Add ones to all PR's connected to PR's with ones through a chain of activated link elements of the orientation(s) specified in this order (H, V, P, N—or any combinations).
Stop	STP	End of program or subroutine.

with the ADD, MPY and SHF orders are R, U, and D referring to right, above (up) and below (down) respectively. Incidentally, it is assumed that a border of modules whose PR contents are permanently zero surrounds the entire matrix.

The link and expand orders require some further clarification. Although they are not shown in Fig. 1, there are one-bit memory elements between every adjacent pair of modules, including diagonal neighbors (so that each module shares eight such elements with its neighbors). These link cells become activated (or de-activated) only when a link order is given, in accordance with the description of that order in Table V. Note that there are four kinds of link cells, depending upon the orientation of the line connecting the two modules that they connect. These are horizontal (H), vertical (V), diagonal with positive slope (P), and diagonal with negative slope (N). An expand order can refer to any combination of these four orientations. When an expand order is given, the activated link cells of the specified orientation(s) establish two-way channels between the modules they connect. A propagation of ones takes place starting from those PR's which contained ones when the EXP order was given, and spreading to all other PR's connected to them via the network of channels.

Although not specifically discussed here, there are means for loading patterns into the module array prior to the executions of a program, and also means for unloading the output. There are also instructions making possible a printed output (required for the recognition program). At this time SPAC exists only in simulated form.

ACKNOWLEDGMENT

The author would like to thank Miss D. M. Habbart of the Bell Telephone Laboratories who wrote the SPAC simulation program which made possible the testing of the ideas described in this paper.