Pattern Recognition and Reading by Machine

W. W. BLEDSOE† AND I. BROWNING†

Introduction

MANY EFFORTS have been made to discriminate, categorize, and quantitate patterns, and to reduce them into a usable machine language. The results have ordinarily been methods or devices with a high degree of specificity. For example, some devices require a special type font; others can read only one type font; still others require magnetic ink.

We have an interest in decision-making circuits with the following qualities: (1) measurable high reliability in decision making, (2) either a high or a low reliability input, and (3) possibly low reliability components. The high specificity of the devices and methods mentioned above was felt to be a drawback for our purposes. All of these approaches prove upon inspection to center upon analysis of the specific characteristics of patterns into parts, followed by a synthesis of the whole from the parts. In these studies, pattern recognition of the whole, that is, Gestalt recognition, was chosen as a more fruitful avenue of approach and as a satisfactory problem for the initial phases of the over-all study.

In addition, we chose to concentrate upon the recognition of alphanumeric patterns, rather than upon other pattern types, for the following reasons:

(1) *Convenience.* Results can be handled easily since it is possible to use conventional printout equipment. Furthermore, we could exploit our own familiarity with letters and words.

(2) *Background.* Research on alphanumeric pattern recognition has been vigorously pursued, and we were therefore able to make use of the relatively large literature on the subject.

(3) *Usefulness.* Success in our efforts would make available a technique which society needs and can use immediately, even though such a result would be only a by-product of our over-all study.

Because typewritten numbers were recognized without error in the cases considered, the investigation quickly shifted to hand-blocked print and finally handwritten script characters as displaying greater complexity and increasing individual variability. In this way the decision making powers of the system were more fully challenged.

Since a numerical output is the inherent mode of expression of a digital computer, our work was aimed at developing a numerical score for each pattern examined. The basic method employed to obtain these scores and to use them to identify each pattern uniquely will be described in the following section. Then various expansions and variations of the method will be covered. Finally, a method of extending identification by contextual relationships will be described briefly.

† Sandia Corporation, Albuquerque, New Mexico

It may be mentioned at this point that this system is highly general — that is:

(1) It handles all kinds of patterns with equal facility.

(2) Because it does not depend upon absolute pattern-matching, it can identify a pattern which is not exactly like, but only similar to, a pattern it has previously learned.

(3) It does not depend significantly upon the location of a pattern on the photomosaic for identification.

(4) It is only partially dependent upon the orientation and magnitude of a pattern for identification.

It would also be well to mention the two major disadvantages of the system:

(1) When the learned patterns are quite variable, the memory can be saturated, especially in certain cases.

(2) A very coarse mosaic, especially if it has inconstant photocell performance, produces images of small letters which do not contain enough information for recognition. See, for example, the sixth character, an *e*, in Fig. 2b. The large letters, however, do not present this problem.

However, both of these disadvantages can be at least partially overcome; the first, by various techniques to be described later; the second, by using a mosaic with more photocells.

Basic Method

Of prime importance in this method is the way in which pattern discrimination is provided. The best way to describe the process is by example.

We start with a 10×15 photocell mosaic (this size being chosen because of immediate availability), the elements of which are related to one another as 75 randomly chosen, exclusive pairs. Fig. 1a shows the mosaic and two such randomly chosen pairs ($1_1 1_2$ and $2_1 2_2$). Images, letters for example, projected on the

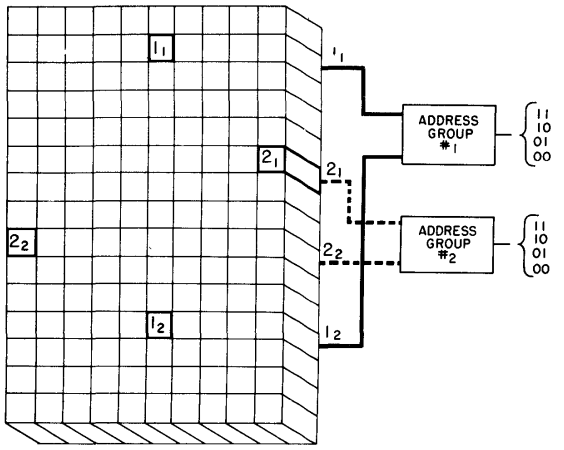

Fig. 1(a)—The photomosiac and two of the randomly chosen photocell pairs. The four digital groups to the right are the four possible states of each photocell pair.

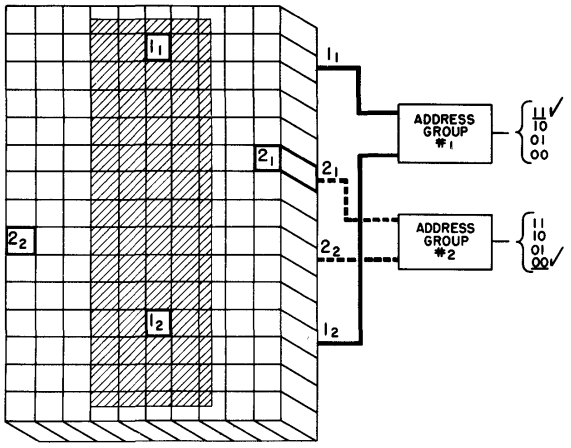

Fig. 1(b)—The system learning the letter I in a central position. Only two of the 75 pairs are shown.

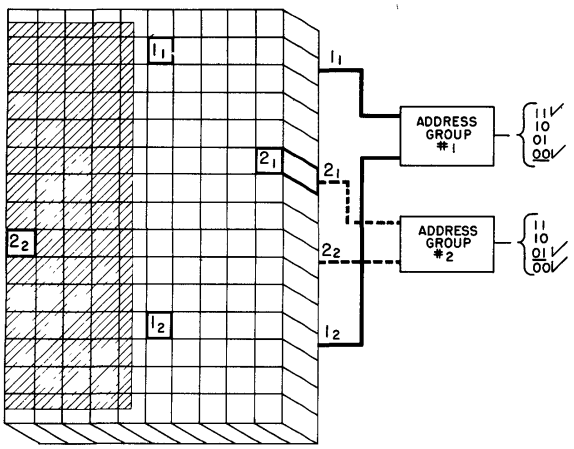

Fig. 1(c)—The system learning the letter I in another position. Note that the memory experience shown in the previous figure remains.

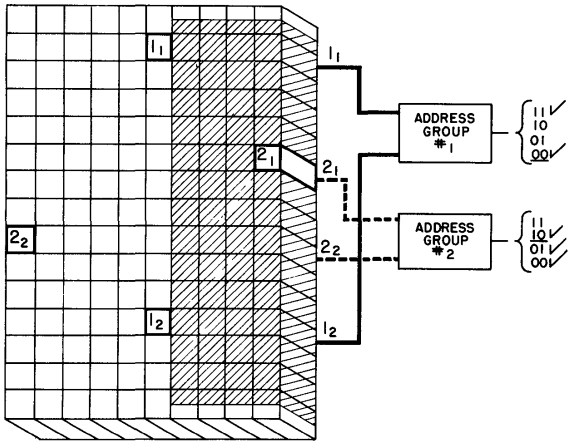

Fig. 1(d)—The system learning the letter I in a third position. The check marks to the right show all possible combinations of these two photocell pairs for the letter I.

mosaic will produce characteristic patterns, examples of which are shown in Figs. 2a and 2b as they appear on IBM cards. For computer convenience, the light values of an image on the mosaic are rendered in a binary system which treats dark as 1 and light as 0. When an image is on the mosaic, each pair of photocells (the members of which are ordered for this purpose) will represent the light values of the image as a two-bit number. Each pair of photocells has therefore four possible states — 00, 01, 10, and 11.

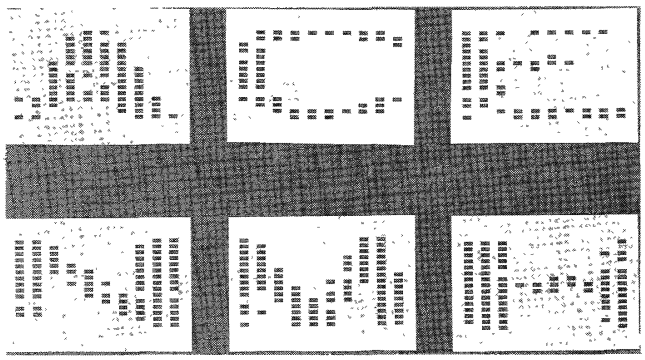

Fig. 2(a)—Hand-block print as it appears on IBM cards. (Top—A, C, E; Bottom—N, M, H.)

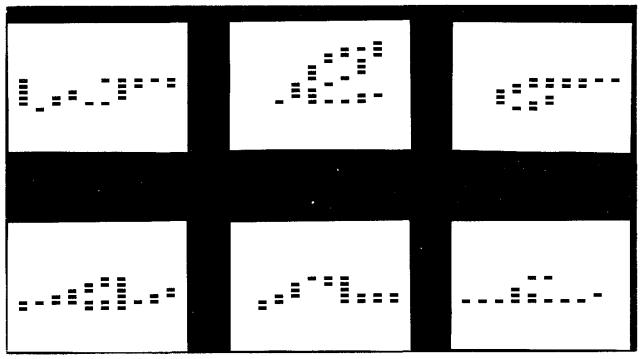

Fig. 2(b)—Handwritten script characters as they appear on IBM cards. (Top — w, l, o; Bottom — s, r, e.)

In the memory matrix of the computer, a 36-bit computer word is assigned to each state of each pair, giving four words for each photocell pair or 300 computer words for the 75 pairs. Furthermore, each bit position in the 36-bit computer words is assigned a pattern nomenclature. The sequence used in our experiment was:

Position

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 . . . 35 36

Nomenclature

1 2 3 4 5 6 7 8 9 a b c d e . . . y z

This nomenclature sequence will hereafter be referred to as an "alphabet."

In order to demonstrate how patterns are "learned," we will use as an example the letter *I*. First, a letter *I* is projected on the photocell mosaic (Fig. 1b). Its image on the mosaic produces one of the pair states (00, 10, 01, 11) for each of the pairs, depending upon the amount of light falling on the pair. Since all 75 pairs are involved, the resulting 75 states address 75 words in the memory matrix. For each word addressed, a binary 1 is entered in the nineteenth position, the position corresponding to the letter being learned, *I*. Obviously, if the letter *A* were being learned, a binary 1 would be entered in the eleventh or *A* position, and so forth. The process described constitutes the learning of a single letter *I*, but whole series of letter I's, differing in shape or position or both, can be learned. For example, Figs. 1b, 1c, and 1d show the same *I* being learned in different positions, while Fig. 3 shows a case in which two *G*'s have been learned.

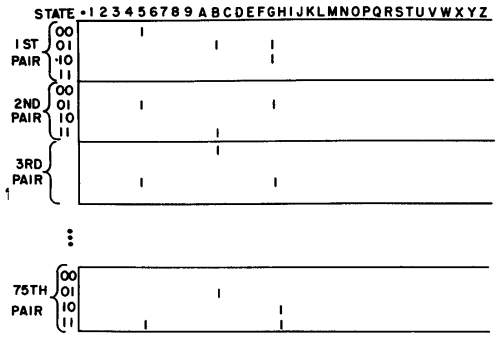

Fig. 3—The memory matrix with the characters B, G, and 5 learned. Note that two G's have been learned.

Since not all the letter *I*'s will be in the same position as the first, *some* different computer words will be addressed. That is to say, there is a degree of individual character variability. However, no letter *I* or combination of *I*'s will normally address the same 75 computer words as, say, a letter *A* would. This is a key point: *the very shape of a character, such as the letter I, forbids certain states for certain pairs. The existence of these forbidden states lies at the heart of our method, for without them the logic would saturate.* In sum, different patterns have different forbidden states and consequently score differently.

Now, suppose that we have taught the logic several alphabets, proceeding for each character as for the letter *I* above. We can then identify a specific unlearned character, an *A* for example. A letter *A* is "read" by imaging it on the photomosaic. Its image will address the 75 computer words in the memory matrix to correspond to the active states of the 75 pairs. Identification of the specific pattern in question is made by comparing the unknown image with the previously learned characters. In practice this is done in the following way:

(1) The binary 1's in position one (the position corresponding to .) are added up for all of the 75 computer words addressed by the unknown pattern. The score obtained shows the similarity of the unknown pattern to the . pattern.

(2) The same process is repeated for the other 35 positions, with the result that 36 numerical scores are obtained.

(3) These scores are compared by the computer, and the highest score wins. That is, the unknown pattern is identified with the character occupying the position scoring highest. If there is a tie for highest score, the computer arbitrarily selects one of the highest scores as the winner. Note that the highest score possible is 75.

Fig. 4 shows an example of scoring for hand-block *A* and *T*. Fig. 5 shows scoring for much more highly variable patterns, namely, handwritten *a* and *t*.

It will be noted that if an image corresponding *exactly* to the unknown image had been learned before by the matrix, a score of 75 would be made at that position. Again, if by learning several similar patterns (*A*'s, for example), all of the pair states now being addressed had been learned, a second 75 would be made. However, in most cases, an unlearned character will not make a perfect score. The degree of simi-

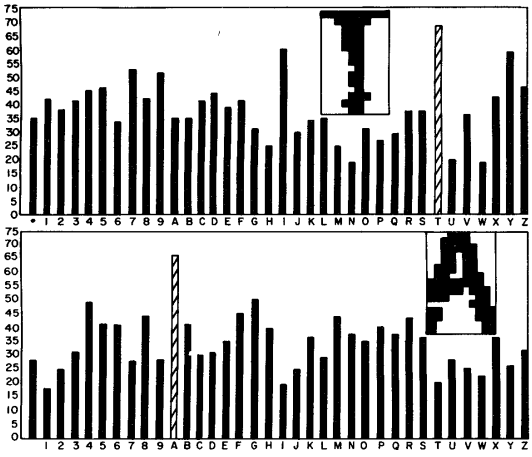

Fig. 4—Comparative scores of hand-block letters.

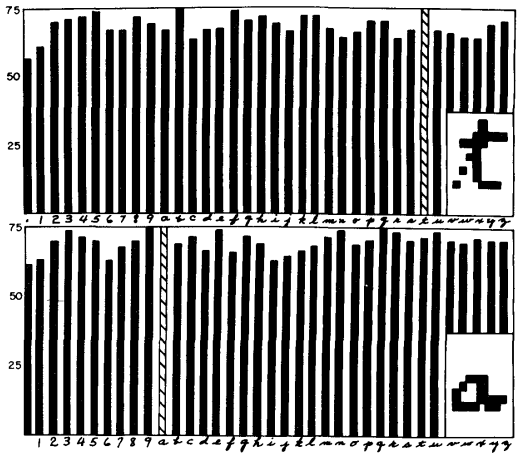

Fig. 5—Comparative scores of handwritten letters.

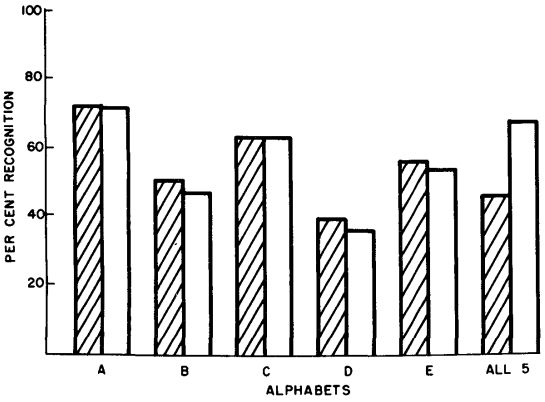

Fig. 6—Comparisons of the percent recognized for hand-block print read with different n-tuplings: $n = 1$ (hatched bars) and $n = 2$ (solid bars). Note that when all five alphabets are learned together, the percent for $n = 2$ improves. In other words, for $n = 2$, the ability to read improves with additional learning in the memory matrix.

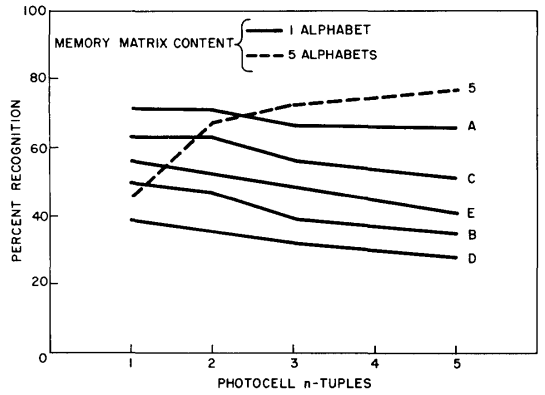

Fig. 7—Comparison of percentage recognition of hand-block print with different n-tuples. Five alphabets (labelled A, B, C, D, and E) are considered singly, and then together.

larity is measured by comparing the magnitude of the various scores with a perfect score of 75. Discrimination is defined as the difference between the score of the correct character and the next highest score. It can be seen that what actually happens in this process is that the images, both those learned and those being read, are transformed into a new space (the memory matrix) and are there compared for identification.

Logic Expansion and Manipulation

Our studies and experiments moved outwards from the basic method to include a variety of modifications and variations. An attempt is made below to evaluate each variation in terms of its final effect. It should be noted that the combination of two or more of the methods to be described results in substantial increases in correct readings.

Different Photocell Groupings

In the examples cited, the photocells were grouped as exclusive pairs. However, it is obviously possible to use n-tuples in which n has any value from 1 to 150. Let us begin by comparing the system employing photocell pairing ($n = 2$) with a system in which $n = 1$. In the latter case, each individual photocell addresses only two computer words, since its possible states are 0 and 1. The difference in the behavior of the systems is striking. If we re-examine Figs. 1b, 1c, and 1d, we note that in learning several images of the letter I, with $n = 1$, every single photocell would exhibit the values 1 and 0: this is so because the position of the letter I changes. In other words, unless the image on the mosaic is held within narrow limits, the memory loses most of its discrimination value with $n = 1$.

We can say then, that position is very critical in the case of $n = 1$, and that it has less importance for $n = 2$. A direct consequence of this difference is found when the matrix is taught more than one position or more than one example of a pattern. The scores will improve if $n = 2$; for $n = 1$, they will not improve and will probably deteriorate. Figs. 6 and 7 illustrate this characteristic with respect to five alphabets learned separately and then in combination. Marked improvement in the reading of this message, which was written in hand-block print, was achieved when $n = 2$ rather than $n = 1$. For the five alphabets learned separately, the average percent of recognition with $n = 1$ was 56.12 percent; for $n = 2$, 54.01 percent. But for the same five alphabets learned together, the percentages are 46.42 for $n = 1$, and 67.63 for $n = 2$. (See also Figs. 8a and 8b.)

Remembering that n can equal any number from 1 to 150, we can ask what effect is produced when higher n-tupling is used. The problem of pattern recognition with a multichannelled system, such as the one simulated for discussion here, has traditionally been approached from one of the two extremes, that is, $n = 1$ or $n = 150$. Consider the formula

$$S^n \times \frac{N}{n} \times C = L\,,$$

where

S = the number of operational states of the photocell. In the case being considered $S = 2$, for the possible photocell states are 0 and 1.

n = the parameter for n-tupling.

N = the number of photocells.

C = the number of categories of patterns learned and read (36 in the previous examples).

L = the number of storage sites in the memory matrix.

The factors held arbitrarily constant in our experiment were $n = 2$, $N = 150$, and $C = 36$. The traditional cases, as mentioned before, have involved $n = 1$ and $n = N = 150$. But the former has been shown to deteriorate or at least not to improve appreciably with learning. The latter, on the other hand, requires a prohibitively large memory matrix (36×2^{150}, using the same values as above), although its reading ability would be perfect if enough learning experience could be provided.

Let us summarize concerning these two extreme conditions. If $n = 1$, there are no forbidden combinations and therefore the memory will saturate with the learning of successive characters which vary in size, area, shape, or position. Such a logic has, consequently, an extremely limited use. If $n = 150$, saturation is impossible. But, even apart from the impossibility of having 2^{150} computer addresses available, images being read successfully would be restricted to exactly those that had been learned before. This logic, then, has even more severe limitations.

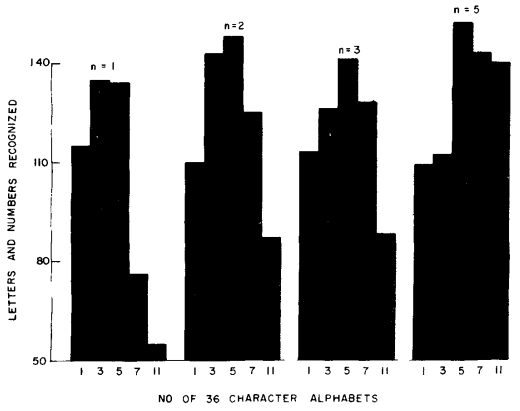

Fig. 8(a)—Scores made on handwritten script letters, showing that for larger values of n, larger amounts of learning are useful.

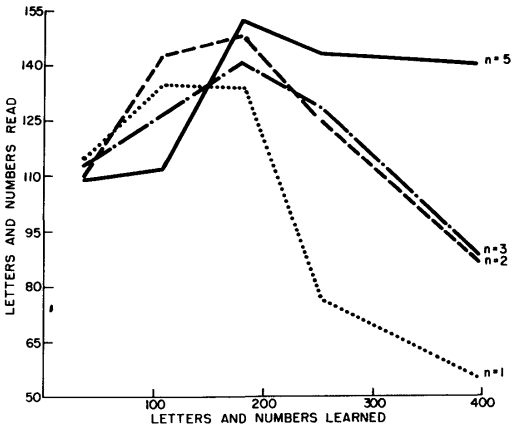

Fig. 8(b)—Material of Fig. 8(a) presented in different form.

Our method avoids these several disadvantages by concerning itself with intermediate values of n, values which provide the learning advantages of a large exponential matrix but which retain a memory matrix more comparable in size to the photomosaic matrix. For example, with $n = 2$, the formula for the logic used gives:

$$2^2 \times \frac{150}{2} \times 36 = 10{,}800$$

The number of bits in the memory matrix for the simplest case of a system not position sensitive, under these conditions, is therefore 10,800.

Let us introduce another quantity, M, which will be the number of photocell n-tuples utilized in a given experiment. While M will normally be given by N/n, larger M values can be obtained by non-exclusive n-tupling of the photocells. We will have more to say about the non-exclusive cases later.

In any event, it is obvious from the formula that a larger memory matrix can be utilized if any of the variables are increased. During the course of our experiments, we used the following values:

$n = 1, 2, 3, 5, 8$

$M = 30, 50, 75, 150, 128, 256, 512, 1024$

$C = 10$ and 36

The experimental data suggest that a greater amount of logic produces better discrimination. The primary effect of varying n is that as n increases, the percent of recognition increases with increased learning (Figs. 7, 8a, and 8b). However, a balance must be preserved among the various parameters in order to utilize to best advantage a given amount of logic and to minimize computing time.

Non-exclusive n-tupling

Some experiments were made in which non-exclusive n-tupling was used for the photocells. The number of n-tuples (M) used could in these cases have any value. Tables I and II show that non-exclusive pairing resulted in some improvement in the percent of characters recognized. But this improvement was at the expense of more storage space and longer computing time. We feel that a larger gain in percent recognized can be realized, for the same amount of storage and same length of computing time, by increasing the number of photocells (N) and continuing to use exclusive n-tuples. In other words, we see no real advantages in non-exclusive grouping.

TABLE I

PROGRESS IN READING HAND BLOCK PRINT

n-Tupling		Alphabets Learned	Manipulation	Percent Read
Exclusive	Non-exclusive			
1		1	None	39–72
5		1	None	28–66
1		5	None	46
3		3	None	78
2,3,5		2	Probability	77–84
2,3,5		3	Distribution	80–84
	2,3,5	1	None	80–85
2,3,5		4	Rotating Origin	88–92
	3	1	Rotating Origin	96
2,3,5		1	Context	94–100
2,3,5		1	Context-Positioning	98–100

TABLE II

PROGRESS IN READING HANDWRITING

n-Tupling		Alphabets Learned	Manipulation	Percent Read
Exclusive	Non-exclusive			
1		1	None	26.14
1		3	None	30.68
2		1	None	25.00
2		5	None	33.64
5		5	None	34.55
5		3	Distribution	43.84
	5	5	None	24.55
	5	5	Positioning	53.15
	5	11	None	50.00
	5	11	Positioning	58.56
	5	11	Rotating Origin	60.00
5		11	Context-Positioning	94.32

Positioning

A procedure for pre-positioning characters for learning and reading by rotating an origin was attempted and found to be profitable in special cases. This rotating-origin technique is useful for digits and for print, but will not work with handwritten script. That is to say, if a character or pattern is separate and distinct, it can have an origin rotated with respect to some reference. Handwriting (as contrasted with the separate handwritten characters which we used) has continuity, and there is no obvious origin from which to start. Some method for separating handwriting into its components would be required before the origin of such components could be rotated profitably.

For each character an origin is arbitrarily defined. The character is then successively repositioned about this origin in the following sequence of x, y values: 0,0; 1,0; 1,1; 0,1; −1,0; −1,−1; 0,−1; 1,−1; 2,0; etc. Scores are obtained for each value, and the maximum score made by a character in any of the positions is chosen as the identifying score. This program involved a considerable amount of computer time, and is of interest mainly in connection with the possibility of simulating conditions for "servoing" the "eyeball." Such a feedback system appears feasible, since effective score criteria were found.

In a variation of the positioning program, the characters were all relocated by the computer to the upper left hand corner of the rectangle. This positioning, combined with the rotating-origin program just described, gives the maximum probability of reclaiming position-dependent data. This combination provides the largest increases in effectiveness for the $n = 1$ cases, those cases which we have seen are most sensitive to position. Typical increases in percent recognized for hand-block print with these techniques are:

Original	*Positioning*	*Rotating origin*
80	84	89
72	88	90

Distribution Processing

A method of processing the data obtained from the pattern scores was tried which was based on the entire scoring pattern rather than upon the maximum score only. The principle involved becomes clear at once if Fig. 5 is re-examined. Note that the sets of scores with respect to the previously learned letters are quite different for a and t. These different values are apparently consistent in their differences. For example, t scores high for b, while a scores low for b, and so forth.

The procedure is first to teach the memory matrix several alphabets as a primary experience. Scores made by one or more additional alphabets, constituting a secondary experience, are then averaged to give a score distribution typical of each character. An unknown pattern is compared with the memory matrix in the usual way to obtain its distribution of scores. This distribution is then compared with the typical distributions and the one most similar to it is chosen. For convenience, all of the scores were normalized, so that the sum of the scores in each distribution was one. Comparisons between two distributions were made in these experiments by summing the absolute values of the differences of the corresponding scores. It might well prove useful to employ a correlation technique in which a sum is taken of the products of corresponding scores, but this has not yet been tried.

As an example of results, in one case in which handwritten script characters were being read ($n = 5$, 3 alphabets learned), we found:

Undistributed	32.3% recognized
Distributed	45% recognized

A final approach in this effort was to introduce ten arbitrary shapes for the primary experience (Fig. 9).

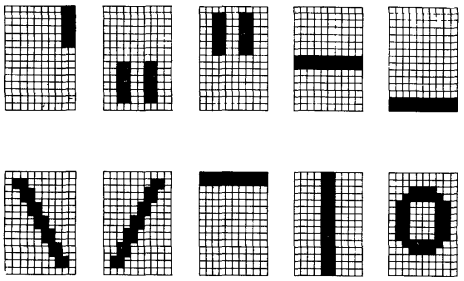

Fig. 9—Arbitrary shapes which were taught to the system as a basic distribution pattern for the subsequent reading of alphanumeric handwritten characters. Each shape was learned in the position shown and also in several positions resulting from lateral displacement.

After these were taught to the memory matrix, three alphabets were compared with the matrix to obtain a ten-component distribution analogous to the 36-component bar graph of Figs. 4 and 5. The three ten-component distributions were averaged. New alphabets could then be read by the distribution-comparison program. For handwritten script the results of this program were:

Undistributed	32.3% recognized
Ten-Component Distribution	51% recognized

This program was novel in that it involved two steps of disorder; that is, two arbitrary operations — random pairing and comparison with arbitrary configurations — were performed on patterns before attempting to read order out of them. It is also important to note that by using only 10 shapes instead of 36, a considerable saving in computer time is realized.

Probability

The method of reading characters described previously utilizes a memory matrix which is taught by a given set of experience patterns. Another method was tried in which the contents of several such memory matrices were *averaged* to obtain a "probability" matrix which was then used as the memory matrix in the reading phase. The memory matrices used in the averaging can be taught by different sets of experience patterns. An interesting (but not very successful) special case is one in which each of the matrices being averaged is taught only one alphabet of experience patterns.

In the few cases tried with this method, the percent of handwritten characters recognized was increased as follows:

Original	28% recognized
Probability Matrix Used	52% recognized

Certain variations of this "probability" method will undoubtedly yield some increase in percent recognition.

Discrimination Criteria

The scores obtained for each pattern read by any of the described methods lend themselves readily to the establishment of discrimination criteria. That is, if the standard of minimum margin is not met for a given image, a secondary program can be evoked which utilizes one of the higher (and probably slower) logic treatments for higher resolution and/or discrimination. Such a program would give the computer a second, and "more careful look" at a pattern which was not clearly recognized on the first trial.

Randomness

Since the elements of the photomosaic are related to each other by randomly chosen n-tuples, it was decided to test the sensitivity of reading ability to changes in the particular organization used. The random (actually pseudo-random) n-tuples were generated by the following program. First a random permutation, $k(1)$, $k(2)$, . . . , $k(150)$ of the numbers 1, 2, 3, . . . , 150 was generated. Then the elements of the mosaic EE_1, E_2, . . . , E_{150}, were related in this manner:

$$(E_{k(1)}, E_{k(2)}, \ldots, E_{k(n)}, (E_{k(n+1)}, \ldots, E_{k(2n)}, \ldots, (E_{k(150-n)}), \ldots, E_{k(150)}).$$

The test was made by using five different randomly chosen permutations to read the same set of patterns. The results are shown in Fig. 10. Although admittedly the sample was rather limited, indications are that the percent recognized is fairly insensitive to the variation, especially when the precent recognized is high.

	CHOICE NUMBER						
n	I	II	III	IV	V	MEAN	σ
HANDWRITING							
2	31%	27%	32%	35%	35%	32%	3.0
5	36%	33%	33%	37%	36%	35%	1.7
HAND BLOCK PRINT:							
2	78.5%	78.2%	77.2%	77.8%	80.1%	78.4%	1.0

Fig. 10—Percentages of recognition for five different choices of random n-tupling.

Context

Another method to extend the basic technique deserves special attention, for it produced the highest percentage of correct readings. It is identification of letters by word context, and it operates as follows:

1. Establish the length of an unknown word by counting the number of characters between spaces.

	.	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
T	30	35	42	46	44	45	39	43	46	45	39	48	37	40	40	43	42	46	42	43	43	48	38	40	35	41	48	39	39	(49)	39	36	36	39	43	41
H	25	37	40	40	40	45	47	37	45	38	35	48	32	37	33	46	43	(47)	45	41	48	47	31	31	34	37	44	31	31	47	33	33	30	33	37	41
E	38	43	43	47	49	46	42	42	46	46	48	46	49	39	(50)	42	46	42	42	41	41	44	42	50	44	48	48	49	47	47	50	46	44	48	46	43

NOTE: WINNERS WERE T; K; U AND E TIED n = 3 M = 50 CI = 396 Mu = 0

CORRECT WORD:

t = 49; *h* = 47; *e* = 50; *the* = 49 + 47 + 50 = 146

OTHER WORDS:

a = 39; *r* = 31; *e* = 50; *are* = 39 + 31 + 50 = 120

l = 48; *i* = 45; *e* = 50; *lie* = 48 + 45 + 50 = 143

t = 49; *i* = 45; *e* = 50; *tie* = 49 + 45 + 50 = 144

Fig. 11—Scoring by context.

2. Establish, by the techniques previously described, all the scores for all of the letters constituting the unknown word.
3. Using a vocabulary of words of the length in question, add in their proper order the letter scores of each word in the vocabulary to obtain a total score for each word.
4. The highest score wins.

Fig. 11 illustrates the whole process. Words can be read by context (see Fig. 12), or, since each word has a score, it would be possible to establish a similar program to deal with phrase context.

The word *the* in the message in Fig. 11 won against 100 other three-letter words even though it was badly misread letter by letter. The results shown

MESSAGE

THE COMPUTATION IS DONE BY THE USUAL MACHINE

FOR n = 2 (11 ALPHABETS)

LETTERS

TKU GXMPYTYTTEN LU DEYT FY TTU UUEET MNQHTUU

CONTEXT

THE COMPUTATION IT DONE BY THE GREAT MACHINE

FOR n = 5 (11 ALPHABETS)

LETTERS

TKE GVMSUTUTIVN 2U DVUM BY TKU USMAD MNCHTUE

CONTEXT

THE COMPUTATION IS DONE BY THE USUAL MACHINE

Fig. 12—Handwritten letters read by context.
Letters and words incorrectly identified are underscored.

in the table were obtained using a vocabulary of 677 most commonly used short words. Obviously a larger vocabulary would result in decreased recognition.

Tables I and II summarize the results obtained for reading hand-block print and handwritten script by the basic method and by the various modifications described.

Discussion

The problem posed by this investigation was: Can a general program be utilized to attenuate the information contained in a higher-order matrix pattern, while at the same time retaining enough of the essence of the information to categorize the pattern. Our results clearly indicate that this is possible. And although such a program will be useful at once for purposes of character recognition as is required in general reading machines, it has a much broader import.

A general program of this sort — as opposed to such specific logical programs as pattern matching or analytical character differentiation — will be useful as a basic tool in our investigation of decision-making circuits. This method could be expanded into such areas as phrase context, the automatic reading of books as a service to language translation programs, etc. It should perhaps be re-emphasized that the program identified typewritten numbers without error in the cases considered. Handwritten script was purposely introduced to challenge the program by offering it patterns of high variability.

Acknowledgments

The authors wish to express their sincere gratitude to Miss Tomasa Santos for her extensive work in programming these concepts and running them on the computer, to Dr. L. S. Lockingen for his large part in interpreting the data, and to Mr. S. D. Stearns for his generous assistance and the use of his equipment.