L. A. Kamentsky
C. N. Liu

Computer-Automated Design of Multifont Print Recognition Logic

Abstract: A computer program has been written to design character recognition logic based on the processing of data samples. This program consists of two subroutines: (1) to search for logic circuits having certain constraints on hardware design, and (2) to evaluate these logics in terms of their discriminating ability over samples of the character set they are expected to recognize. An executive routine is used to apply these subroutines to select a complete logic with a given performance and complexity. This logic consists of 39 to 96 AND gates connected to a shift register and a table look-up or resistance network comparison system.

The methods were applied to the design of recognition logics for the 52 upper and lower case characters of IBM Electric Modern Pica type font and lower case Cyrillic characters scanned from Russian text. In both cases when the logics were tested on data different from that used to design the logics, the substitution rate was about one error per thousand. A single logic was designed to read two different Cyrillic fonts. For this design, an error rate of one error per hundred characters was observed.

Several experiments are reported on a number of logics designed for typewritten data, and single- and two-font Cyrillic data. The performances of different recognition systems are compared as a function of the complexity of the recognition logics.

Introduction

This paper describes a program, to be called the *Recognition Logic Designer*, which was written to apply the systematic and rapid data-handling capabilities of the computer to the problem of designing logic for large-alphabet, single-font, and multifont print recognition machines. To date, most work in character recognition[1,2] either has involved devices where templates of characters are stored and matched to unknown characters or has been concerned with methods of describing the attributes of the patterns formed by these characters; these attributes have been derived through experience and intuition. It is our opinion that even when one can alter character shapes to improve discrimination in large character sets, the process of designing recognition logic requires the screening of a vast amount of data. The design of high-performance recognition logics, even for limited problems like single-font print recognition, is difficult because designers cannot easily manipulate all of the possible bit patterns that a useful machine may be required to handle.

During the last few years we have witnessed the application of the computer to the solution of many decision processes which are nonnumerical in nature. It has been found that the computer can be economically applied to the study of the performance of complex systems without constructing them. Of much interest has been the use of the computer to simulate models of human problem-solving situations such as theorem proving.[3,4] One problem that humans solve very effectively is visual pattern recognition. However, few usable algorithms or heuristic procedures exist for the solution to the problem of designing recognition logics by machine.

Most character recognition methods may be conveniently considered as performing two functions:

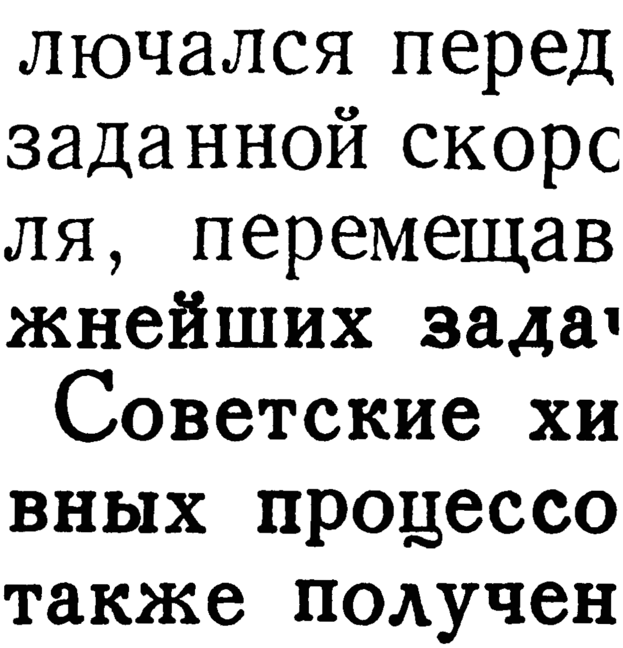
(1) to measure certain attributes of the patterns to be recognized and (2) to decide into which character classes the patterns belong, on the basis of the results of these measurements. Such measurements have been given as individual pattern elements,[7] pairs of elements,[8] geometric features,[9] or randomly chosen groups of pattern elements.[10, 23] With the exception of the recent work of Bonner,[5] Lewis,[6] and Uhr and Vossler,[24] the automatic construction of recognition measurements has not been considered to any extent. The basis for measurements has usually been selected intuitively. Well-known decision procedures, implemented by a computer program, a logic or resistance network, or optical masks, were then adapted to recognition of membership in classes. This paper concerns the design of measurements and presents several usable heuristic procedures for obtaining good recognition measurements with a computer.

The present design procedure is based on a statistical analysis of actual samples representative of those the recognition machine will be called upon to recognize. Statistical design procedures in which recognition designers have used the computer as a design aid have been reported.[11-14] It was our aim, however, to arrive at a completely automatic computer design procedure based on the analysis of representative data and on the automation of some of the methods of the human designer.

A flying-spot scanner was used to reduce the images of characters on photographs of printed text to records on magnetic tape. Each character was isolated from its neighbors by circuits in the scanner and was represented on magnetic tape as a pattern of ONES and ZEROS. A large group of different characters was manually identified. The character identities and corresponding bit patterns were then read into the IBM 7090 computer. In two to three hours the computer produced a wiring table for a recognition logic.

During those two to three hours, the computer (1) generated circuits from a large group of implementable logics; (2) evaluated each circuit against a set of samples, called "analysis data," in terms of its discriminating ability; (3) selected the best logic circuits it had evaluated; (4) tried the complete logical system on a set of different samples; and (5) tried the previous steps again if recognition was poor. On these retrials, the computer was made to emphasize the input characters which the logic previously misidentified. A computer program embodying these principles has been developed and has been used to produce logic for both recognizing typewritten text and printing in two fonts taken from journal text. The logic designed by the computer program consists of 39 to 96 AND gates connected to a shift register and a table look-up or resistance network comparison system.

Three typical problems given to the program and performance of their solutions are given below:



лючался перед
заданной скорс
ля, перемещав
жнейших зада'
Советские хи
вных процессо
также получен

Figure 1 **Two Cyrillic fonts from Russian journal text which were scanned in the character recognition experiment.**

1. Three typewriters were used to provide 1560 prints of various upper and lower case IBM Electric Modern Pica characters that were scanned by a flying-spot scanner and used by the program to design a logic. The logic was tried on 1300 samples, comprising 25 each of the 52 upper and lower case alphabetic characters. These test data were of the same font but were taken from three different typewriters. All but two characters were read correctly; an "l" was called an "i" and a "j" was called an "l". Alternatively, a decision criterion could be adjusted so that no characters were misread if four characters out of 1300 were rejected.

2. The two Cyrillic fonts from Russian journal text shown in Fig. 1 were scanned. Thirty samples from each of the two fonts of the 26 lower case characters that occur most frequently were used to design the logic. The three characters л, д, and ц were so different in each of the two fonts that they were treated as separate characters in each of the fonts. The logic was tested on 870 characters different from those used to design the logic. These data had many broken and filled-in characters. Eight characters were misread. Alternatively, an error rate of 0.46% could be maintained at a rejection rate of 0.7%. A different logic was designed using only the first font. There was one error when this logic was tested on 1180 different characters of the same font.

3. In the third problem, 4793 of the 32 lower case Cyrillic characters were scanned. Typical binary patterns for these data are shown in Fig. 2. Because of the experimental conditions, parts of some characters were

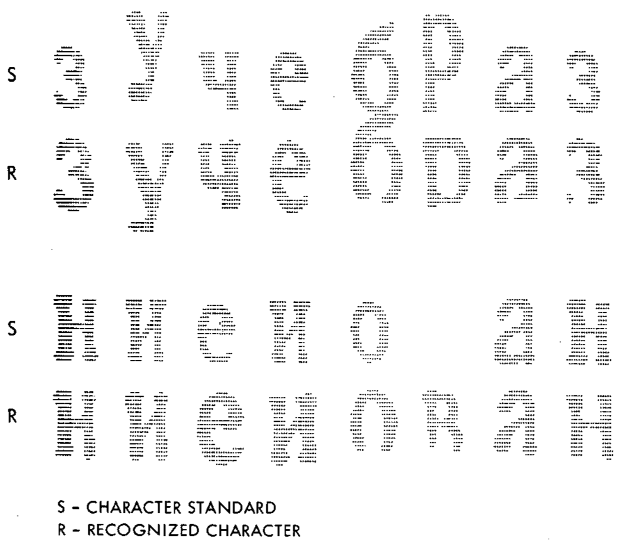
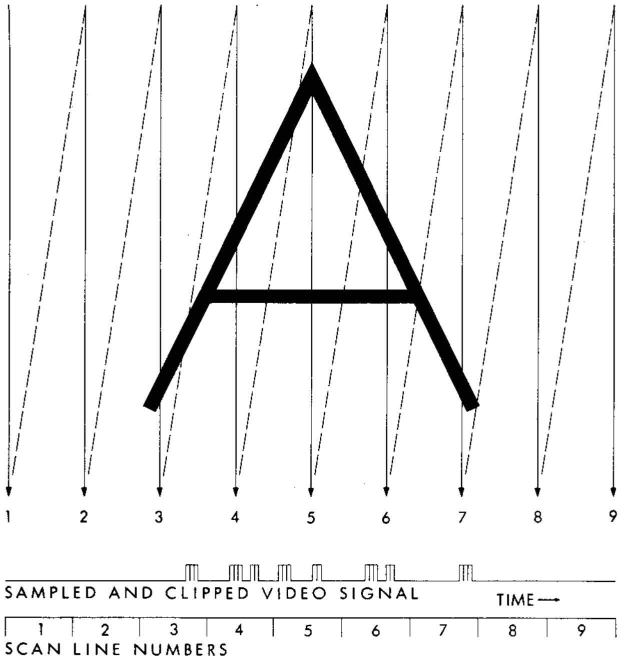


Figure 2 **Typical binary patterns for Cyrillic characters.** *S: character standard. R: recognized character*

Figure 3 **CRT scan pattern and output.**



missing and the bit patterns of characters differed in size as the data were scanned. These two effects are evident in Fig. 2. With a logic designed with 1008 characters, four out of 3785 different lower case Cyrillic characters were misrecognized.

Computer design of recognition logic

The recognition logics to be reported on in this study are characterized by two important features:

1. They depend on a fixed set of measurements on every character, these measurements being independent of character registration. A measurement consists of a determination of the occurrence in the scanned character of a set of black and white points satisfying some prescribed spatial relationship. This feature guarantees that recognition will be independent of misregistration inherent in the source document, or instabilities and inaccuracies in document handling or in the optical equipment.[15]

2. An identification is made by comparing the results of the set of measurements with sets of reference values, one set for each character in the alphabet to be recognized. The comparison may take the form of any of several statistical decision criteria.[16,17]

This paper is primarily concerned with methods of finding good measurements. However, a number of different decisions of different criteria complexities were studied and were found to differ in performance. These will be reported on in the experimental section of this paper. The next sections describe a computer-automated process to find M measurements consisting of switching functions on an N-bit representation of character patterns which have good discrimination over a set of input characters. The design of this system of logic is to be based on typical samples of the input. In order to realize this objective, we have adopted three principles, which are the topics of the next three sections:

1. There must be a way to restrict the number of switching functions that are to be considered in the design procedure. In effect, an efficient search procedure for logic must be found.

2. There must be a quantitative measure of the discriminating ability of a set of switching functions used over actual samples of the character set they are expected to recognize.

3. There must be an executive routine to apply the above two principles to actual samples, to select good switching functions, to test the recognition ability of the chosen logic and to repeat the procedure, emphasizing improperly identified characters until the logic performs adequately.

Methods of logic generation

To reduce the number of logics considered, the following three constraints have been imposed on the generation of recognition switching functions:

1. Only recognition logics which are invariant to translation of the input characters with respect to the logic were considered.

2. Recognition logics consisting of only certain types of n-tuples, with conditions on the positions of each of the n points with respect to each other were considered.

3. Specific switching functions on these n-tuples such

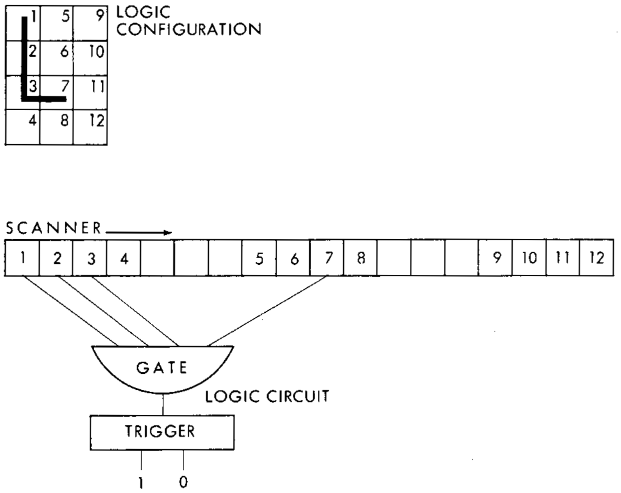


Figure 4 **The shift register with one digital mask connected in an "L" configuration.**

as AND-ing only or majority logics of the n points were used.

It must be admitted that, even with constraints using these three principles, the number of possible switching functions that can be generated is still quite large, requiring the use of a selection strategy. We feel, however, that the yield of useful switching functions picked with these constraints is much higher than if they were picked using any reasonable selection strategy without constraints.

A model of a recognition system will now be described and the embodiment of these constraints will be related to hardware. The character field is scanned sequentially, as shown in Fig. 3, and the bits obtained are shifted into the register shown in Fig. 4. If the characters to be read are R bits high, $R - 1$ blank bits are placed into the register after each scan. Consider now a particular logic configuration such as the "L" in Fig. 4. The bits upon which this configuration fall are connected to a combinatorial logic circuit to be called L.C. If the contents of the register are shifted to the right, the same configuration will appear as input to L.C. but in a different position relative to the input matrix. In the system studied, L.C. is set to ONE if a particular state of its inputs occurs for any shift position. M translation invariant switching functions such as L.C. are connected to the shift register.

The possible configurations of the input are further reduced by considering constraints on the bit configurations. The constraints on the bit configurations being used are shown in Fig. 5. In each of the three constraints shown, n points are selected from a possible 64 positions. These n points are each assigned to a ZERO or to a ONE state. We are now using the specific logic of AND-ing the states of input matrix points at the relative positions of the "masks". If the mask contains a ONE, that input position state is AND-ed; if a ZERO is contained, the input state is inverted and AND-ed. A total of n states are AND-ed to determine the output state of each L.C. The remaining $64 - n$ cells of each L.C. are DON'T-CARE conditions. The choice of n was based on a study of the discrimination of the logics and the frequency of match of the logics and data, as a function of the complexity of the logics. The results of this study are described in the Experimental Results section. The constraints were chosen so that local features of the characters are emphasized but some global information is extracted. Other constraints

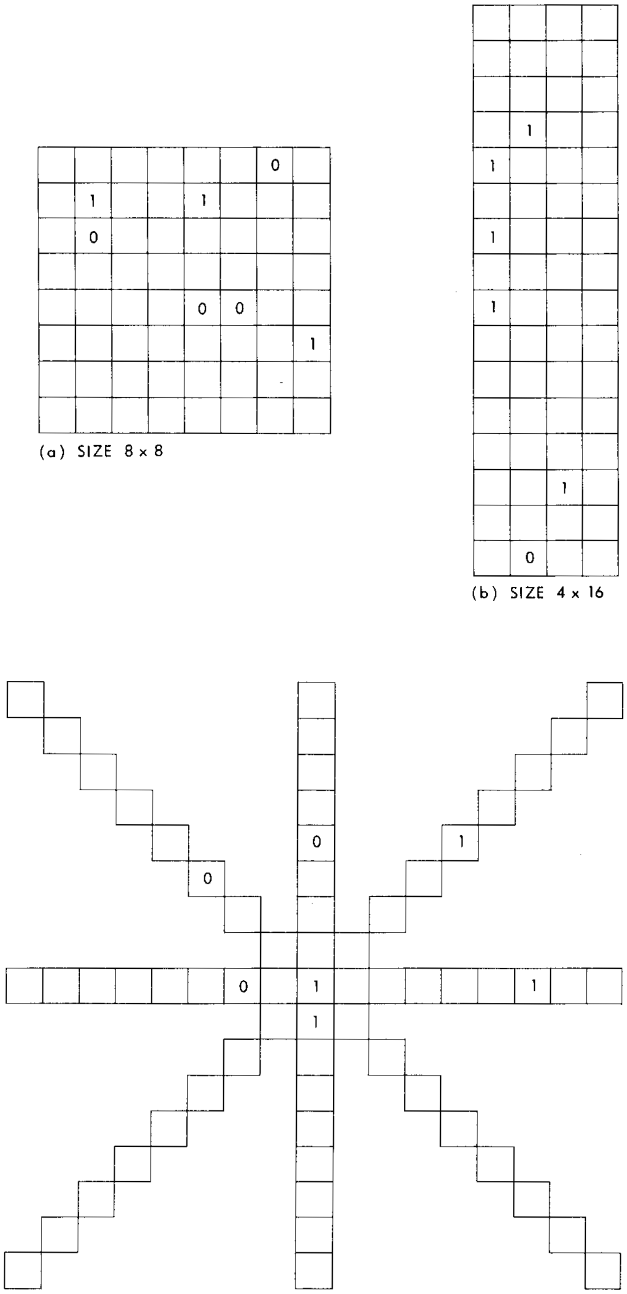


Figure 5 **Three typical digital masks.**

can be applied to reduce the number of positions required in the shift register.

Two search strategies for selecting cells from the mask constraint have been considered. The first strategy selects the best $r + 1$-th tuple (based on an evaluation function) using an r-tuple as a base. By repeating this process for $r = 1$ to $n - 1$, n-tuples may be synthesized by adding complexity until the value of each logic synthesized no longer increases with n. This procedure was found to require too much computer time and only simple random selection of constrained n-tuples is reported here. Two pseudorandom numbers are used to generate the locations and the states of each mask point. The generated L.C.'s are recorded on punched cards to be used later for their evaluation and to be used to construct a wiring table.

Methods of evaluating recognition logics

A measure is used to determine the worth of the particular parameters extracted by each logic L.C. A second measure is used to determine the redundancies in partitioning the character set of a group of logics. These measures are described below:

- *An information measure*

An ideal observer (similar to the ideal observer of Woodward's radar theory)[18] can be postulated for a character reader. The ideal observer specifies the distribution of the states of the input signal to a physical system, based on the state of the output. In this case the input is a character pattern and the output the state of a set of parameters. The ideal observer conserves all information relevant to specifying the input, but no more. Consider a character reader with M parameters, $x_1, x_2 \cdots x_M$ used to classify m different characters, $c_1, c_2 \cdots c_m$. Then the conditional probability distribution of the character set $P\{c_i|\mathbf{x}\}$, given the particular state of the M parameters $\mathbf{x}$, completely describes the input for any state of the M parameters. If for any probable $\mathbf{x}$, this distribution is peaked, that is, one of the characters has probability near one and the other $m - 1$ characters have probability near zero, then this is a good set of parameters. If on the other hand, the probabilities are all nearly equal, then the parameter set is poor. The following measure derived in the Appendix is being used to get a quantitative value for this property:[19]

$$I = \log_2 m + \sum_{\mathbf{x}} P\{\mathbf{x}\} \sum_{i=1}^{m} P\{c_i|\mathbf{x}\} \log_2 P\{c_i|\mathbf{x}\} \; .$$

$P\{\mathbf{x}\}$ is the probability of the parameter state $\mathbf{x}$ and the first sum is taken over all states of the parameter set.

Table 1 **Probabilities of matching for four different logics.**

Character	*Logic 1* $I = 0.784$	*Logic 2* $I = 0.380$	*Logic 3* $I = 0.215$	*Logic 4* $I = 0.025$
A	0.76	0	0	1.00
B	1.00	0.88	0.27	1.00
C	0.94	0.24	0	1.00
D	0.97	0.91	0.03	1.00
E	0.97	0.85	0.12	1.00
F	0.21	0.67	0.03	1.00
G	0.91	0.30	0.03	1.00
H	1.00	0.89	0.09	1.00
I	0	0.73	0.03	1.00
J	0	0.97	0.03	1.00
K	1.00	0.89	0.21	1.00
L	1.00	0.82	0	1.00
M	1.00	0.24	0.06	1.00
N	0.61	0.36	0.06	1.00
O	0.82	0.33	0	1.00
P	1.00	0.97	0.06	1.00
Q	1.00	0.24	0.03	1.00
R	1.00	0.82	0.21	1.00
S	1.00	0	0.12	1.00
T	0	0.91	0	1.00
U	1.00	1.00	0.06	1.00
V	1.00	0.45	0.09	1.00
W	1.00	0.45	0.09	1.00
X	1.00	0.89	0.45	1.00
Y	1.00	0.76	0.12	1.00
Z	0.03	0.48	0.12	1.00
a	0	0.27	0.33	1.00
b	0.39	0.42	0.03	1.00
c	0	0.09	0	1.00
d	0	0.06	0.03	1.00
e	0	0.55	0.24	1.00
f	0	0.36	0	1.00
g	0	0.48	0.91	1.00
h	0.12	0.36	0	1.00
i	0	0	0	1.00
j	0	0.03	0	1.00
k	0	0.67	0.03	1.00
l	0	0	0	1.00
m	1.00	1.00	0.15	1.00
n	0.52	0.70	0	1.00
o	0	0.12	0.06	1.00
p	0.45	0.94	0.03	1.00
q	0	0.15	0	1.00
r	0	0.06	0	1.00
s	0	0.18	0.97	1.00
t	0	0.70	0	0.89
u	0.06	0.06	0.68	1.00
v	0.27	0.09	0.03	1.00
w	1.00	0.24	0.21	1.00
x	0.48	0.15	0.27	1.00
y	0.33	0.06	0.58	0.97
z	0	0.33	0.33	1.00

This measure is applicable for determining the value of the complete set of parameters $\mathbf{x}$. It has been applied to evaluating a particular logic circuit j. In this case, x_j has two states and the probabilities $P\{c_i|x_j\}$ and $P\{x_j\}$ are easily computed. The experiments (determining each x_j) are, of course, not independent and the individual information values obtained are not additive. However, the values of I obtained for each L.C. agree with our intuitive judgment of its worth. The value I is 0 if the L.C. has no discrimination; all characters either match or don't match. The value of I is a maximum of one bit when every character of one-half of the set of characters matches and the other characters do not match the L.C. Table 1 shows the values for I and the frequencies of matching on a specific set of data for four different but typical L.C.'s.

An evaluation is first made over the complete set of data to select L.C.'s which best divide the character set into two parts. At a later stage the evaluation function is only applied to specific characters to force the system to choose L.C.'s which are less efficient but will resolve confusion sets.

● *A redundancy measure*

When using the previous measure there is no assurance that each of the measurements will not divide the alphabet into the same two parts. Experimentally, we found that the random choice of masks did lead to a random partitioning. Another question that arose during the study is the number of logics that must be selected to meet a certain specified error rate. Theoretically, if m characters are to be classified, $\log_2 m$ binary logics which correctly partition the character alphabet into two parts will be necessary and sufficient. However, since the data samples may have many unpredictable variations, some redundancy must be provided if confusions between classes are to be eliminated. Thus, feature code representations of different classes must maintain a certain distance. Following this line of reasoning we have used the following method for minimizing a large number of logic circuits to yield a smaller set with a given minimum distance between pairs of characters.

Let us assume that a set of N L.C.'s has been designed to recognize a set of m classes successfully. We can compute the pairwise information measure $I_{lk,j}$ for every pair of classes c_l and c_k and the parameter of each L.C. x_j,

$$I_{lk,j} = 1 + \sum_{x_j=0,1} P\{x_j\}[P\{c_l|x_j\}\log_2 P\{c_l|x_j\} + P\{c_k|x_j\}\log_2 P\{c_k|x_j\}] .$$

The values of $I_{lk,j}$ may be viewed as the elements of a $C_2{}^m$ by N matrix, (I_{ij}) where each information value, $I_{ij} = I_{lk,j}$, indicates the separation power of the j^{th} measurement on the i^{th} pair of characters c_l and c_k.

The elements of (I_{ij}) are now quantized into ZEROS and ONES. Each element I_{ij} is set equal to one if it is greater than a certain threshold value θ and is made zero otherwise. Let us assume that r measurements are required to distinguish each pair, that is, a minimum separation distance of r is needed. A threshold value θ is chosen that will produce at least r ONES in each row of the matrix. Next, the rows are rearranged such that the number of ONES in each row increases as i increases, and the number of ONES in each column decreases as j increases. Each row in the matrix is then checked and the columns marked that will produce r ONES in each row. Only the marked measurements are preserved.

The following example illustrates the procedure. The matrix shown below represents a typical quantized and rearranged pairwise information matrix (I_{ij}). It is required that this matrix be reduced so that the minimum distance r is three and thus at least three ONES be left in each row after reduction. The marked columns below represent the reduced set of measurements.

$$\begin{array}{cccccccccccc}
\checkmark & \checkmark & \checkmark & \checkmark & \checkmark & & \checkmark & \checkmark & & & & \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0
\end{array}$$

Executive routine

In this section the different steps that are executed during the design of a recognition logic will be outlined. Before doing this we will describe how the data is prepared for the Recognition Logic Designer.

Characters are scanned from photographic film reproductions of typewritten or journal text by a cathode ray tube flying-spot scanner. Rows of characters are digitized and written as records on magnetic tape. Each row of characters is contained within a scan column 32 bits high; lower case characters are about 15 bits high and 10 bits wide. This magnetic tape is the input to a computer run which automatically separates adjacent characters, assigns a sequence number to them, and prints out the bit patterns of the scanned characters along with their sequence numbers. This printout is manually read and the true identity of each of the characters is associated with its sequence number.

Using these identified bit patterns the Recognition Logic Designer works as follows:

1. About 1000 logics derived from each of the three constraints are generated, stored on magnetic tape, and punched out on cards to be used later for preparing a wiring table. A computer run is made with the bit patterns of a set of scanned data to be used for analysis as one input and the pregenerated logics as the other input. A compiler transforms each of the logics into a list of logical statements to be executed by the program. The data bit patterns are transformed by the logical statements to produce a binary word for each character with a component bit for each logic which is ONE if that logical statement is met and is ZERO if it is not met. These words, to be called *feature codes*, are the basic input to all of the routines which follow

and correspond to a binary representation of the state vector **x**.

2. The information content for each logic is computed. A predetermined number of the logics having the largest information value I are selected. The feature code for each character of the analysis data is reduced in size so that only the selected logics are represented by it.

3. For each class of characters c_i, the frequency of each of the states for each selected logic x_j is computed. These conditional probabilities $P\{x_j|c_i\}$ are used as the weights of a Bayes' Rule decision procedure.

4. A new set of data is selected. These data, treating each of the characters as unknowns, are recognized by the selected logic with conditional probabilities developed from analysis data by using the decision rule:

$$\text{Maximum}_i \left[G_i = \prod_{j=1}^{M} P\{x_j|c_i\} \right].$$

The state of each bit x_j of the feature code determines which of two probabilities, $P\{x_j = 1|c_i\}$ or $P\{x_j = 0|c_i\} = 1 - P\{x_j = 1|c_i\}$, is multiplied for each character c_i to compute G_i. G_i is proportional to the inverse probability $P\{c_i|\mathbf{x}\}$ with the assumption of independence on the x_j's and equal *a priori* probabilities $P\{c_i\}$.

The most probable identification (the maximum G_i) is found. (The ratio of the probabilities of the two most likely choices will be used later as a rejection criterion. If this ratio is less than a certain pre-set level, the unknown is rejected and no positive identification is made in this case.)

5. If all of the characters are not recognized correctly, each character class which has at least one character misrecognized or substituted for another character is a member of a confusion set. Steps 1 and 2 are repeated on the original analysis data. However, only those character classes which are members of the confusion set are used during this run. Logics are selected which have a high information value when they resolve members of the confusion set. These additional logics are included with the previously generated and selected logics and Steps 3 and 4 are repeated with the enlarged measurement set. Steps 1 to 4 are repeated until the error rate of the system reaches a certain desired level.

6. The logics generated are next tested to eliminate unnecessary redundancies. Using the analysis data, the logics are reduced in number until there is a predetermined distance r between all pairs of characters.

Experimental results

Four types of data were digitized by a flying-spot scanner and were available for application of the methods that have been described in this paper. Since three of these will be referred to in each of the first five experiments to be described below we will first describe those data.

Typewriter. Each of six electric typewriters was used to supply ten samples each of the 62 upper/lower case alphabetic and numeric characters. All typewriters used Modern Pica type font. The typing was done by the users of the particular typewriters. The conditions of the typewriters were as we found them. In the subsequent reports, confusions between upper and lower case of the same character are not considered to be errors.

Single-font Cyrillic. Pages from the Russian *Journal of the Academy of Sciences* were photographed and the photographs scanned. A sample of a few lines from this journal is shown in the top half of Fig. 1. The infrequent occurrence of upper case and certain lower case letters from text allowed us to obtain reasonable analysis and test data sample sizes for the Cyrillic alphabet for 26 lower case letters only.

Two-font Cyrillic. Pages from the Russian *Journal of Applied Physics* were photographed and scanned. A few lines of these data are shown in the bottom half of Fig. 1. The two-font samples included data from each of the two Russian journals. As we pointed out earlier, the three Cyrillic characters л, д, and ц are so unlike in each of the fonts that they were treated as six independent characters during the experiments.

Experiment 1

Discrimination as a function of the complexity of each logic

The first experiment is concerned with the number of points n in each mask to be AND-ed to form a logic. Masks were generated at random using only the eight-by-eight box constraint. Logic circuits were formed from these masks by AND-ing four, five, six and seven points and these logics were evaluated. The information value I of each logic was determined using both typewriter and two-font data. A yield, defined as the percentage of logics having a value of $I \geqq 0.5$, was determined for each value of n for each type of data. The results are summarized in Table 2.

It should be noted that for the experiment with typewriter data there were 52 different characters and

Table 2 **Yield as a function of *n*.**

Number of points n		4	5	6	7
Percent of logics with $I \geqq 0.5$	Two-font	3%	12%	13%	14%
	Typewriter	2%	5%	6%	6%

for the experiment with two-font data, samples of only 26 lower-case characters were used. Therefore, the values of computed I were in general higher for two-font data. From the results tabulated above, it was apparent that seven-point logics would be more informative than logics with fewer numbers of points.

• *Experiment 2*

Performance with and without selection

This experiment was performed to determine if recognition results are significantly better if the logics are selected using the information measure I. First 1050 logic circuits with $n = 7$ were generated using the three constraints. Then 75 of these were chosen at random to form a recognition logic. Based on 1560 samples of alphabetic typewritten data, the 75 logics with the largest I were selected to form a second recognition logic. Both logics were tested on 780 samples of typewritten data taken from different typewriters than those used to select the logics. A Bayes' decision rule was used with conditional probabilities computed from the data used to select the logics. If the ratio of the two largest values of G_i was smaller than 10 for any character in the test data that character was rejected. The results are shown in Table 3.

• *Experiment 3*

Comparison of decision methods

Four different decision methods have been compared using the logic generated by the Recognition Logic Designer as the basis for determining the set of parameters used to describe the data. These decision rules may be implemented by either stored logic[16] or by a resistance network.[17] For either of these it is important that the precision required of the storage devices or the resistors be determined. We have tested recognition performance with the precision of the decision method as a parameter.

Three of the decision methods are based on maximizing an *a posteriori* probability with equal *a priori* probabilities for each of the different character classes (Bayes' Rule). The conditional probabilities have been quantized into three different precisions. The Bayes decision rule described in the preceding Section is based on a computation of conditional probabilities from 30 samples of each character. These conditional probabilities: (1) have been used unquantized, (2) have been quantized into eight levels so they can each be represented by three bits and (3) have been quantized into three levels so that they can each be represented by two bits.

Table 3 **Performance with and without selection.**

Logic Selection Method	*Random (No Selection)*	*Information Measure Selection*
Error rate	1.54%	0.13%
Reject rate	10.62	0.25

The fourth decision method assigns the unknown character to the class having the minimum distance from it. Each of the conditional probabilities for each character and logic is converted into a new representation S_{ij} based on the following assignments:

$P\{x_i = 1 \mid c_j\}$	S_{ij}
$\geqq 0.7$	1
$< 0.7 > 0.2$	don't care
$\leqq 0.2$	0

For each unknown character, the elements of its feature code $\mathbf{x}$ are compared to the elements of each column of the matrix S_{ij} representing the state vector for the class c_j. A distance to each class c_j is computed as the number of times $x_i = 1$ when $S_{ij} = 0$ and $x_i = 0$ when $S_{ij} = 1$.

Each of the four decision methods was tested on typewriter, single-font Cyrillic, and two-font data. In each case the same set of 30 samples of each character was used to select the logic and to determine the parameters in the decision rule. This logic was then tried for all four decision methods on a set of data different from the data used to design the logic. The results are shown on the following page in Table 4.

• *Experiment 4*

The effect of redundancy on performance

If m character classes are to be recognized, in principle only $\log_2 m$ measurements need be used. This implies that each measurement has an information value of one; that is, all characters of a given class will produce the same state for every measurement. Furthermore, it implies that each additional measurement m breaks the character set into 2^m parts, giving a minimum distance r (as defined in the section on evaluating recognition logics) of one between all pairs of characters. We have determined the effect of redundancy, based on a measure of the minimum distance between pairs of characters, on the recognition performance of a set of logics. Using the alphabetic typewritten data, 30 samples of each character were used to select a set of 75 logics. The minimum distance between all pairs of the 52 characters was six. Using the reduction technique described previously, the system was reduced to 48 logics with the minimum distance of five. The system was reduced again to 40 logics with a minimum distance of four. These three sets of logics were tested on 1300 samples different from those used to select the logics or to reduce the number of logics. The results in Table 5 show the performance for the different systems with the unquantized and three-level quantized Bayes' decision rules.

In another experiment using the same data, the performance of the system was determined as a function of the sum of the information values of the individual logics used. The most informative logics were selected for each point from the set of 75 logics described above. A Bayes' decision rule was used in this experiment. The results are presented in Fig. 6.

- *Experiment 5*

Performance as function of analysis data sample size

To design logic for a character recognition machine, the Recognition Logic Designer must be supplied with actual samples of patterns representative of those a recognition machine will be called upon to identify. The method we have described is self-designing in the following sense. Given representative samples of the set of characters to be recognized, there is a specific procedure for producing a wiring diagram or a list of values in a storage medium to generate a machine to recognize the input set. The performance of a machine will depend to some degree on the variety of the samples supplied to the machine designer. Generally, the more these input samples cover the different variations to be encountered in practice, the better the machine will perform.

It is therefore important to understand the performance of the recognition design procedure with various sample sizes. We have found that the values of the conditional probabilities used in the decision systems are the most sensitive elements of the recognition system when the sample size is varied. Table 6 illustrates this. Logics were designed with 30 samples of each of the 52 characters of typewriter data. A Bayes' decision procedure was used to recognize 1300 different samples. The conditional probabilities used in the decision rule were computed using 10, 20, and 30 samples of each character. We have also found that the recognition rate is the same for both the new data and the data used to design the logic if the sample size is 30.

- *Experiment 6*

Performance with large variations in the data

At the present time we are carrying out a series of experiments using the Recognition Logic Designer on more varied data than that which was described earlier. These variations include changes in character size and the recognition of characters in fonts not used during the design procedure.

To determine the ability of the logics to recognize data of a different size, tests were run on 425 new typewritten alphabetic characters whose size was 5% to 7% smaller than those used to design the logics or for computing the conditional probabilities. Using a Bayes' decision rule with a rejection ratio of 10, there were one error and two rejects.

In journal text, there is no control over the sequences or frequency of characters. Since these characters must be identified to the Logic Designer program, some means must be accomplished for associating the identities with the bit patterns of each input.

We are presently using an approach to data compilation in which a primitive form of the logic is used to compile data for the design of a better recognition logic.

Table 4 **Performance with different decision methods.**

Decision Method	*Alphabetic Typewriter,* 1300 *Samples* 75 *Logics*		*Single-font Cyrillic,* 1180 *Samples* 54 *Logics*		*Two-font Cyrillic,* 870 *Samples* 73 *Logics*		*Numeric Typewriter,* 464 *Samples* 39 *Logics*
	Reject	*Error*	*Reject*	*Error*	*Reject*	*Error*	
Bayes	0.3% 0	0 % 0.15	0.34% 0	0 % 0.08	0.7% 0	0.46% 0.92	No Errors or Rejects
Bayes (8 Levels)	1.1 0	0 0.4	0.64* 0	0.32 0.43	1.6 0	0.7 2.3	
Bayes (3 Levels)	1.4 0	0.08 0.54					
Minimum Distance	1.77 0	0.08 0.65	0.43* 0	0.32 0.54	1.3 0	1.6 2.3	No Errors or Rejects

* 936 samples were tested in these cases.

The recognition program is first structured by the Recognition Logic Designer based on a few samples which have been manually identified. The program is then used to identify the remaining samples. Although its present performance is inferior to its ultimate performance, it is adequate to identify a large percentage of the input at a high rate of speed. The computer is arranged to print out, for each character, the input bit pattern along with a bit pattern taken from a library of standard characters indicating its recognition for that character. Samples of this printout are shown in Fig. 2. The first and third rows illustrate the programs' recognition for the input characters below each of the "character standards". (In this case, the machine had been given 15 samples of each of the lower-case Cyrillic letters.) The machine errors are quickly identified manually and the identifications corrected. The larger set of samples identified in this way is used to design a better performing machine.

In analyzing these data the recognition logic, although structured only with the lower-case Cyrillic alphabet, was found to recognize correctly most italicized characters and upper-case characters if the lower-case version of these characters was the same general shape. This is illustrated in Fig. 2. We are presently studying tolerance to large variations in size and font. The data shown in Fig. 2 are typical of that resulting with a scanner ripple component which was introduced to affect the horizontal size of the scanning field such that this field varied by an average size difference of 20% between characters. Using these data, the set of 32 lower-case Cyrillic characters was recognized. The logic design and the conditional probabilities that were used in the decision procedure were based on 1008 samples different from those that were tested for recognition. The results using various decision rules on this data are shown in Table 7.

Table 5 **Performance with different numbers of logics.**

Number of Logics	Alphabetic typewriter, 1300 samples			
	Bayes' Decision			
	Not Quantized		Three-Level	
	Reject	Error	Reject	Error
75	0.3% 0	0 % 0.15	1.4% 0	0.08% 0.54
48	0.8 0	0 0.2	2.4 0	0.6 1.4
40	1.4 0	0.2 1.0	5.6 0	0.8 1.8

Conclusion

A computer program has been developed for designing specific character recognition logics using actual samples of data. This program has been applied, thus

Figure 6 **Performance as a function of total information.**

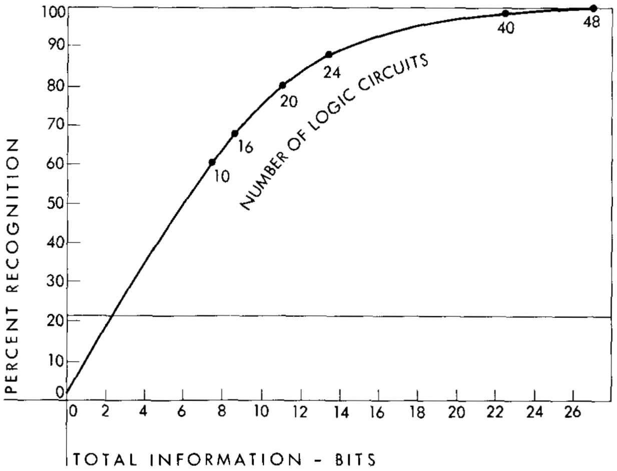


Table 6 **Performance for various sample sizes.**

Number of samples to compute $P\{x_j\|c_i\}$	*Error rate with no rejects*
10	5.6 %
20	1.2
30	0.15

Table 7 **Performance with large width variations.**

Decision Method	*Single-font Cyrillic* 4095 *Samples* 96 *Logics*	
	Reject	*Error*
Bayes	0.21%* 0	0.05% 0.19
Bayes (8 Levels)	0.37 0	0.15 0.34
Minimum Distance	0.8 0	0.07 0.44

*3785 Samples in this test.

far, to the design of recognition logics of practical complexity for reading single- and two-font printed alphabets. Substitution rates in the order of one error per thousand characters were achieved on a wide range of printing quality. The application of contextual analyses[10, 16] to word recognition, using the methods described in this paper for the character recognition, should produce machine designs with word error rates below one word error per thousand words.

In the application of this method of designing recognition logic, it has become apparent that there are relationships between logic complexity and performance of a recognition system. System constraints to control the amount of hardware required to implement a system can be specified to form a part of the design procedure. Some experimental relations were reported on the effect of these constraints on the performance of a recognition machine. These constraints involve the complexity of the logic circuits and the number of logic circuits. Furthermore, we showed the performance resulting from different methods of treating the outputs from the logic circuits. Finally, results were reported as a function of the quality and variety of characters handled by the machine. For example, a given logic system worked better on a smaller alphabet than on a larger one; recognition of more than a single font by a single logic yielded inferior performance than that logic applied to a single font. Conversely, equivalent performance with poorer quality or more varied inputs could be obtained at the expense of more complex logic.

Acknowledgments

The authors would like to acknowledge the programming assistance and suggestions given to them by John Cocke and K. C. Hu. They would also like to thank E. N. Adams and M. C. Andrews for their support and helpful comments during this study.

Appendix

The parameterization of the input is to yield M different parameter values, $x_j, j = 1, 2, \cdots M$. All of the x_j are presented to the classifier at the same time and will be considered here as a point $\mathbf{x}$ in a M dimension space. The parameter values are considered to be discrete; they may be binary, they may represent a multistate measurement, or they may represent samples of a continuous but frequency-band-limited measurement.

The classifier, using all of the parameter values, produces one of m possible codes c_i, $i = 1, 2, \cdots m$. In the simplest case, each code c_i represents one alphanumeric symbol of m possible symbols. Only this case will be used here. Although we will use only single symbols, there is no reason why the theory to be developed can not be applied to groups of symbols.

It will be assumed that the *a priori* distribution of the code sequences $c_1, c_2, \cdots c_i$, $P\{c_i\} = P\{c_1, c_2, \cdots c_i\}$ has been measured and is known and stationary for some finite number of symbols $r \geqq 1$. If $r = 1$ only the symbol frequencies are known. In effect, r is the range of contextual influence.

The recognition logic, R.L., is presented with an unknown input symbol S. Based on measurements of M parameters x_j of S, R.L. must associate one of m possible codes c_i with S. The codes c_i are mutually exclusive and exhaustive; that is, every S belongs to one and only one of the c_i. Let us consider for a particular set of measurements of S yielding the value $\mathbf{x}$, the conditional probability $P\{c_i|\mathbf{x}\}$. This is the probability that the input symbol S is associated with the code c_i based on the parameter values $\mathbf{x}$. *This set of conditional probabilities describes all of the information in the parameter values* $\mathbf{x}$ *relevant to classifying* S *independently of any decision method.*

Woodward[18] has described the ideal receiver as one which specifies the distribution of the states of the input signal based on knowledge of the output signal. The ideal receiver conserves all information relevant to specifying the input but no more. The ideal classifier conserves all information relevant to specifying S given $\mathbf{x}$ and no more. Thus, given $P\{c_i|\mathbf{x}\}$ a decision method need not be considered as part of the character reader if we are interested in studying the ability of a given parameterization to extract information about S.

Some insight into the problem of rating parameterizations might be gained by considering the following example. Fig. A-1 represents the set of $P\{c_i|\mathbf{x}\}$'s for a particular $\mathbf{x}$ of a parameterization α. In general, the set of probabilities will be different for each $\mathbf{x}$ but consider this set in Fig. A-1 to be typical of α. Fig. A-2 represents a typical set of $P\{c_i|\mathbf{x}\}$'s for a parameterization β. We feel intuitively that more is known about S from parameterization β than from α if the particular $\mathbf{x}$ in each case is typical. This is true because the distribution in the case of β is more peaked. In this case, one of the codes $c_i = c_l$ is much more probable than the others. Restated, the more peaked the distribution of $P\{c_i|\mathbf{x}\}$ the more is known about S. If we can find a computable and good measure of the peakedness of $P\{c_i|\mathbf{x}\}$ and average this measure over all of the parameter values for a given parameterization, we can produce a measure of the value of that parameterization in separating codes.

An information measure will be used here to describe the spread of $P\{c_i|\mathbf{x}\}$. It is in effect a measure on how much we know about the c_i's and is a maximum if the c_i's are completely determined. Conversely, Shannon's[20] information measure describes how much knowledge is contributed by each c_i and is a minimum if the c_i's are determined *a priori*. The measure to be used here is similar to one used by Lewis[21] to describe approximations to probability distributions and to one used by Lindley[22] to compare experiments to determine the value of a continuous parameter.

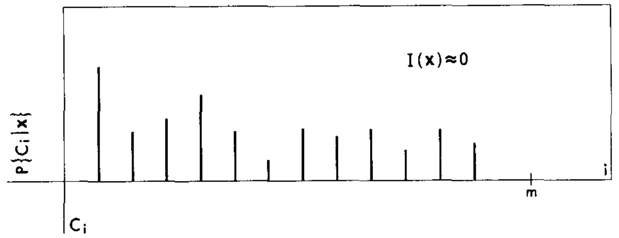


Figure A-1 **Distribution of code probabilities for a specific measurement $\bar{x}$, case α.**

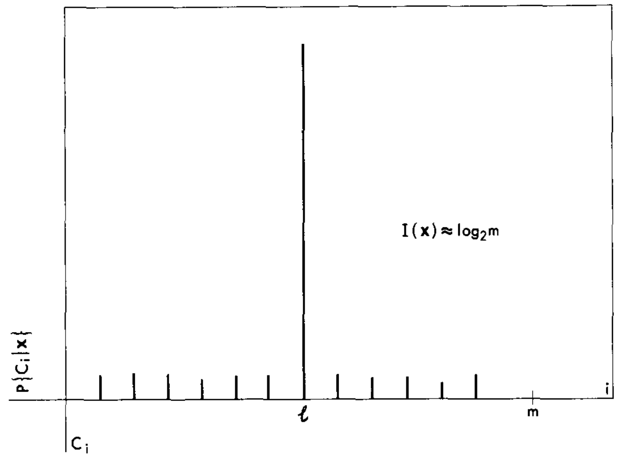


Figure A-2 **Distribution of code probabilities for a specific measurement $\bar{x}$, Case β.**

First, consider the following measure which is a function of $\mathbf{x}$:

$$I(\mathbf{x}) = \sum_{i=1}^{m} P\{c_i|\mathbf{x}\}\log_2 P\{c_i|\mathbf{x}\} - P\{c_i\}\log_2 P\{c_i\} .$$

Shannon[20] has shown that the function $\sum_{i=1}^{m} P_i \log_2 P_i$ is a maximum for all of the P_i equal, zero only if one of the P_i's $= 1$ and the others are zero, and monotonic between these limits. Thus if the probabilities $P\{c_i|\mathbf{x}\}$ equal their *a priori* probabilities $P\{c_i\}$, $I(\mathbf{x}) = 0$. For the perfect parameterization, $P\{c_i|\mathbf{x}\} = 1$ for $i = 1$, $P\{c_i|\mathbf{x}\} = 0$ for $i \neq 1$ and, if $P\{c_i\} = 1/m$, $I(\mathbf{x}) = \log_2 m$. (We can specify one of m states with $\log_2 m$ bits.) Since $\sum_i P_i \log P_i$ is monotonic, $I(\mathbf{x})$ gives a measure of the spread of $P\{c_i|\mathbf{x}\}$ when using a particular value of $\mathbf{x}$.

The probabilities of the states of the parameter values may be described by a distribution $P\{\mathbf{x}\}$ where $\sum_{\text{all } \mathbf{x}} P\{\mathbf{x}\} = 1$. The average information I of a parameterization is then the expected value of $I(\mathbf{x})$ for the parameterization or

$$I = \sum_{\text{all } \mathbf{x}} P\{\mathbf{x}\} I\{\mathbf{x}\} .$$

If the *a priori* probabilities of each character c_i are equal, $P\{c_i\} = 1/m$, then

$$I = \log_2 m + \sum_{\mathbf{x}} P\{\mathbf{x}\} \sum_{i=1}^{m} P\{c_i|\mathbf{x}\}\log_2 P\{c_i|\mathbf{x}\} .$$

References and footnotes

1. M. E. Stevens, *Automatic Character Recognition, A State-of-the-Art Report,* U.S. National Bureau of Standards, Technical Note No. 112, May 1961.
2. M. Minsky, "A Selected Descriptor-Indexed Bibliography to the Literature on Artificial Intelligence," *IRE Transactions on Human Factors in Electronics,* **HFE-2**, 39-55 (March 1961).
3. A. Newell, J. C. Shaw, and H. A. Simon, "Empirical Explorations of the Logic Theory Machine," *Proceedings WJCC*, pp. 218-239, Feb. 26, 1957.
4. H. Gelernter, "Realization of a Geometry Theorem Proving Machine," *Information Processing*, Unesco (Paris) 1960, pp. 273-282.
5. R. E. Bonner, "A 'Logical Pattern' Recognition Program," *IBM Journal* **6**, 353-360 (1962).
6. P. M. Lewis, "The Characteristic Selection Problem in Recognition Systems," *IRE Transactions on Information Theory*, **IT-8**, 171-178 (1962).
7. W. H. Highleyman, "An Analog Method for Character Recognition," *IRE Transactions on Electronic Computers*, EC-**10**, 502-512 (1961).
8. L. P. Horwitz and G. L. Shelton, Jr., "Pattern Recognition Using Autocorrelation," *Proceedings of the IRE* **49**, 175-185 (1961).
9. O. G. Selfridge, "Pattern Recognition and Modern Computers," *Proceedings WJCC*, pp. 91-93, 1955.
10. W. W. Bledsoe and I. Browning, "Pattern Recognition and Reading by Machine," *Proceedings EJCC*, pp. 225-233, 1959.
11. E. C. Greanias, "Some Important Factors in the Practical Utilization of Optical Character Readers," *Proceedings for the Symposium on Optical Character Recognition*, Spartan Books, August 1962.
12. R. J. Evey, "Use of a Computer to Design Character Recognition Logic," *Proceedings EJCC*, pp. 205-211, 1959.
13. D. N. Freeman, "Computer Synthesis of Character-Recognition Systems," *IRE Transactions on Electronic Computers*," **EC-10**, 735-747 (1961).
14. J. J. Leimer, "Design Factors in the Development of an Optical Character Recognition Machine," *IRE Transactions on Information Theory* **IT-8**, 167-170 (1962).
15. Registration invariant logic has been described by many authors. See Refs. 6, 8, 9, 11, 12, 13 and 14.
16. M. C. Andrews, "Multifont Print Recognition," *Proceedings for the Symposium on Optical Character Recognition*, Spartan Books, August 1962.
17. M. L. Minsky, "Steps Toward Artificial Intelligence," *Proc. IRE* **49**, No. 9, 8-30 (1961).
18. P. M. Woodward, *Probability and Information Theory with Applications to Radar*, Pergamon Press, 1953.
19. A similar measure has been applied to character recognition measurements by P. M. Lewis (Ref. 6).
20. C. E. Shannon, *The Mathematical Theory of Communication*, University of Illinois Press, 1949.
21. P. M. Lewis, "Approximating Probability Distributions to Reduce Storage Requirements," *Information and Control* **2**, 214-225 (1959).
22. D. V. Lindley, "On a Measure of the Information Provided by an Experiment," *Ann. Math. Stat.* **27**, 986-1005 (1956).
23. F. Rosenblatt, "Perceptron Simulation Experiments," *Proceedings of the IRE* **48**, 301-309 (1960). See also G. Palmieri and R. Sanna, "Automatic Probabilistic Programmer/Analyzer for Pattern Recognition, "*Estratto Rivista Methodos* **12**, #18, 1 (1960).
24. L. Uhr and C. Vossler, "A Pattern Recognition Program that Generates, Evaluates, and Adjusts its own Operators." *Proceedings WJCC*, pp. 555-561, 1961.

Received July 30, 1962