Experiments in the recognition of hand-printed text: Part I—Character recognition

by JOHN H. MUNSON

Stanford Research Institute
Menlo Park, California

INTRODUCTION AND BACKGROUND

Among the many subject areas in the field of pattern recognition, the recognition of machine-printed and hand-printed alphanumeric characters has perhaps been the classic example to which people have referred in exemplifying the field. Interest in character recognition has long run high; an extensive literature in hand-printed character recognition alone dates back to at least 1955.[1–36]

In recent years, the recognition of machine printing has become a commercial reality. Following the introduction of the highly controlled E13B magnetic font by the banking industry, several advances in optical character recognition (OCR) capability have been brought to the marketplace. The trend of these advances is toward the acceptance of broader and less controlled classes of input: from single, stylized fonts to multi-font capability; from high-quality copy to ordinary inked-ribbon impressions, and even to multipart carbons of surprisingly poor quality. Still, in contrast to hand printing, the approaches to OCR have been able to rely on the lack of gross spatial distortions in the character images, and to make considerable use of templates.

Progress in the off-line recognition of hand printing has been slower. The problem is intrinsically harder than that of OCR, as reflected in the fact that the human recognition error rate for isloated, hand-printed characters is many times higher than for machine printing. The great spatial variability of hand-printed characters has led many researchers to explore non-template methods for recognition.

Thus, the major effort of many researchers has been the exploration of unique methods of preprocessing, or feature extraction, applied to the hand-printed character images. Dinneen,[1] in one of the earliest papers, investigated local averaging and smoothing operations to improve the quality of the character image. Similar operations have appeared as a part of many other approaches.[4,7] Lewis,[15] Uyehara,[21] Stern and Shen,[23] and Rabinow Electronics[31] have used schemes in which the sequence of intersections of a slit scan with the character image, or the equivalent, gave rise to features for classification. Lewis[15] was one of the relatively few to emphasize the use of multiple-valued rather than binary-valued features, an ingredient we have found important in our own work.

Singer[12] and Minneman[30] employed a circular raster, which can facilitate size normalization and rotation invariance. Unger,[7] Doyle,[9] and Glucksman[27] have emphasized features derived from shape attributes such as lakes, bays, and profiles. The building up of a character representation from component elements matched to the image, such as short line segments or portions of the boundary, has been attempted by Bomba,[4] Grimsdale et al.,[6] Kuhl,[19] and Spinrad.[26] Correlation techniques have been tried by Highleyman[13] and Minneman.[30] Contour-following with a captive flying-spot scan or its simulated equivalent has appeared in the work of Greanias et al.,[20] Bradshaw,[22] and Clemens.[28] The work of Greanias et al.,[20] is especially significant because it led to the method used in the IBM 1287 character reader.

Other workers have placed greater relative emphasis on classification techniques and on the selection of features from a feature set or pool. Chow[16,29] has long worked with statistical classification methods. Bledsoe and Browning[3] and Roberts[8] applied adaptive procedures to features obtained from more or less random connections with the image raster. Uhr and Vossler[11] performed an important pioneering study of a program that "generates, evaluates, and adjusts" its own parameters. Not surprisingly, however, the automatically generated features were confined to simple, local templates.

The recognition of characters printed subject to

specific constraints (such as guide markers appearing in the printing area) has been studied by Dimond,[2] Kamentsky,[14] and Masterson.[18]

It may be said of most of these investigations that they were in the academic, rather than the practical, realm. In general, the methods were never tested against a body of real-world data large enough to give some estimate of their performance in a practical situation. This probably reflects a common emphasis on checking out a preprocessing scheme rather than attacking a particular application problem; it certainly also reflects the labor and equpiment requirements involved in collecting and controlling a significant body of data. An exception to this general statement is the work of Highleyman and Kamentsky in the early 1960's, in which they used data files numbering in the thousands of characters.[13,14] Also, several files each containing many thousands of characters of graded quality were gathered in conjunction with the development of the IBM 1287 character reader and are currently in use at IBM and in our group. Bakis et al.[35] describe these data, on which they and others at IBM have performed extensive experiments.

The use of context to improve recognition performance, which figures prominently in our own work, was discussed briefly by Bledsoe and Browning,[3] but otherwise has received scant attention in the past. Some studies have been carried out under simplifying assumptions such as Markov dependence in digrams and trigrams.

Chodrow et al.[31] surveyed hand-printed character-recognition techniques in 1965 and discussed at some length the procedures of Clemens,[28] Greanias et al.,[20] and Rabinow Electronics. The book *Pattern Recognition* by Uhr[32] reprints a number of the important source papers[3,6,8,11] and contains a well written survey. An early progress report on the work described herein was given by Munson.[36]

Recently, commercial organizations have announced the capability to read off-line hand printing. At the date of this writing (early 1968), one system (the IBM 1287 optical reader) has achieved pilot production operation. The 1287 reader can read the ten numerals and five letters. Another system is announced to have full alphanumeric capability.

A common characteristic of the announced systems is that they are intended to work with hand printing of very high quality, produced by coders who have undergone training in the skill of printing for machine recognition. If individual characters must be recognized with, say, better than 99.9% accuracy in order to yield usable document acceptance rates, this type of training is clearly required. Some experiments that will be described in the next section show that humans cannot recognize isolated characters printed by an untutored population with any rate approaching the required accuracy.

In our work, we have taken the alternative approach: Given text from an untutored coder, in which the individual characters cannot be recognized (by man or machine) with high accuracy, contextual analysis is used to reduce the error rate. Every form of text has its own contextual structure, which is utilized by humans in a complex, largely unconscious process. We have therefore emphasized the following points in our research: the establishment of large hand-printed data files of known quality; the choice of a well defined character alphabet and textual situation (FORTRAN program texts) as a vehicle for study and the reporting of results; the use of multiple approaches to preprocessing; context analysis to improve recognition; and the preservation of non-binary confidence information between the preprocessor and classifier and between the classifier and the context analyzer.

In a companion paper,[37] Duda and Hart describe the use of programmed contextual analysis in the recognition of FORTRAN program texts. The present paper will therefore concern itself only with the problem of recognizing individual characters.

Problem definition

In a recent paper, the author has argued that there is an infinity of character-recognition problems, and that recognition results are meaningless as they are often reported in the literature, without an adequate description of the problem being treated.[38] Accordingly, we shall try to describe the two recognition problems dealt with in this paper thoroughly enough that the reader can form an intuitive opinion of the difficulty of the problems.

We must first distinguish between off-line character recognition from a printed page, and on-line recognition, in which the characters are generated by a light pen, RAND tablet, or similar device.[24,33,34] On-line recognition is much simpler because the data provide a nearly exact trace of the path of the writing instrument and give accurate stroke-position and time-sequence information. Furthermore, an error rate of as much as 5% may be considered acceptable, because each character can be classified, displayed, and corrected immediately by the writer if it is wrong.

The recognition of hand-printed characters should also be distinguished from that of cursive (connected) script.[25] The separation of the printed characters and the fact that each belongs in a well-specified category obviate the "segmentation problem" that makes cursive-script recognition much more difficult.

Within the framework of off-line block hand printing, the difficulty of a particular problem is still affected by many variables: the size of the alphabet; the "standard" forms of the individual characters and the degree of constraint placed on their formation; the size, spacing, and arrangement of text on the page; the writing instrument(s); the number of writers; their training and motivation; and the (fixed and time-varying) characteristics of each individual writer. To illustrate the variability of hand printing, we may cite several instances of human recognition rates on samples of hand printing. Neisser and Weene reported a 4.1% average error rate on characters printed by visitors at the front gate at Lincoln Laboratory.[10] With all subjects voting together, the error rate was 3.2%. We have reported an error rate of 11% on the well-known quantized character set collected by Highleyman, which suffers from crude quantization of the characters.[39] On the multiple-coder data file used in our experiments and described below, the error rate was 4.5%; on the single-coder file, 0.7%. Finally, present commercial systems are intended to operate with character error and reject rates on the order of 0.1% to 0.01%.

The most significant determinants of hand-printing quality are the training and the motivation of the printing population. Our choice in the work described in this paper was to treat data from an essentially untutored, moderately motivated population, represented by computer users who hand-code program texts for keypunching. Such a coder has typically received no instruction in printing, beyond a few rules about slashing or crossing characters to avoid such confusions as I-1, 0-zero, and 2-Z. He does receive feedback of the results from prior keypunching jobs, which motivates him to maintain (perhaps grudgingly) a certain level of legibility. Thus, while this printing is far sloppier than that allowed by presently announced recognition systems, it is more legible than that produced by the general public while, for example, addressing mail.

Two files of data were used in the experiments reported in this paper, a multiple-coder file and a single-coder file. The characters in both files were hand-printed on standard general-purpose coding sheets obtained from the Stanford Research Institute computer center. The cells on these sheets measured 1/4 inch high by 3/16 inch wide, with no extra spacing between cells. A thin-lead mechanical pencil with an HB (soft) lead was used, after brief experimentation indicated that no other conventional writing instrument gave crisper images when viewed through our input system. (A pencil is the preferred instrument because it facilitates erasure.) The coder was free to use whatever character size he found natural.

The 10 numerals, the 26 uppercase letters, and the symbols [= * / + − ., $] comprised the alphabet of 46 characters. This is the basic FORTRAN alphabet, with brackets substituted for parentheses in accordance with the convention associated with our computer system at the time. The blank was not treated as a character category, the recognition of blanks being more a function of a document-scanning subsystem than a pattern-recognition problem. We instructed the coders to print zero with a diagonal slash and Z with a midline slash, and to put crossbars on the letter I. Numeral 1 was to be without serifs; several coders, however, added serifs. Other choices were left to the individual, such as open versus closed 4, the crossbar on J, and the number of verticals in $.

Multiple-coder file

Printed data from 49 individuals were included in the multiple-coder file. Each person was asked to print several 46-character alphabets on a coding sheet (at one sitting), and the first 3 alphabets from each sheet were taken for the file. The data from the first 32 persons (96 alphabets, 4416 characters) were used as training or design data during the experiments, and the data from the remaining 17 persons (51 alphabets, 2346 characters) for test. The coders of the training data were all personnel of the author's laboratory and the computer center at SRI. The coders of the test data were 8 from SRI and 9 from the US Army Electronics Command, Fort Monmouth, N.J. Any cross-country bias in printing styles is probably small compared with individual differences.

Portions of several of the test alphabets are shown in Figure 1. The coders were asked to print naturally, being neither especially casual nor especially meticulous. However, it is obvious that data gathered this way are not candid; they are probably better than data from actual coding sheets prepared for keypunching. Unfortunately, it was not feasible for us to process candid data from a number of people using a variety of coding forms and languages.

Five human subjects were asked to classify the characters in 17 of the test alphabets—one from each coder—viewing the quantized images (see the section on scanning) in isolation and in random order on a cathode-ray tube display. The error rates ranged from 3.0% to 6.4%, with an average of 4.5%. Taking a plurality vote among the five responses, the error rate was 3.2%.

Single-coder file

Experiments were also performed with a single-

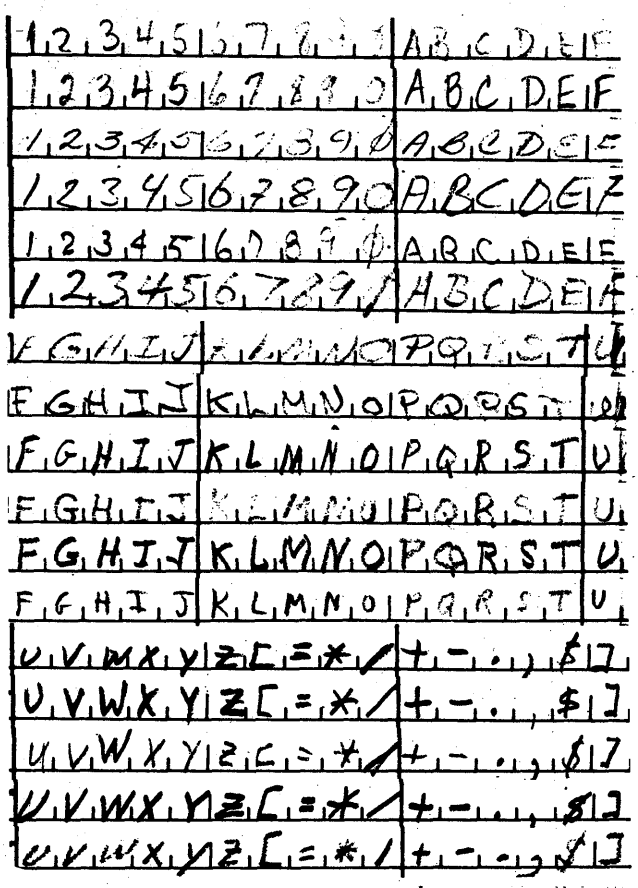

FIGURE 1—Portions of several multiple-coder test alphabets

coder file, in order to investigate the improvement in performance resulting from allowing the recognition system to specialize in the printing of a single individual. This file contained 1727 training characters and 1042 test characters. The training set included 15 alphabets (690 characters) of the type collected for the multiple-coder file. The remaining 1037 training characters were taken from FORTRAN text on coding sheets, as were the 1042 test characters. The 15 alphabets were included in the training set to ensure adequate representation of all the character categories, since their appearance in actual text was haphazard.

The text characters were taken from FORTRAN coding sheets prepared by the author in the course of actual program development, some months before the recognition experiments were performed. The coder corrected major malformations of characters as he noticed them, but avoided printing with unnatural care. Thus, while these data are not candid, it is felt that they closely model a realistic situation that would be obtained if one tried to serve a coder who was making a minimal effort to assist the system.

A sample of the test data is shown in Figure 2. We may describe these characters as being quite legible to

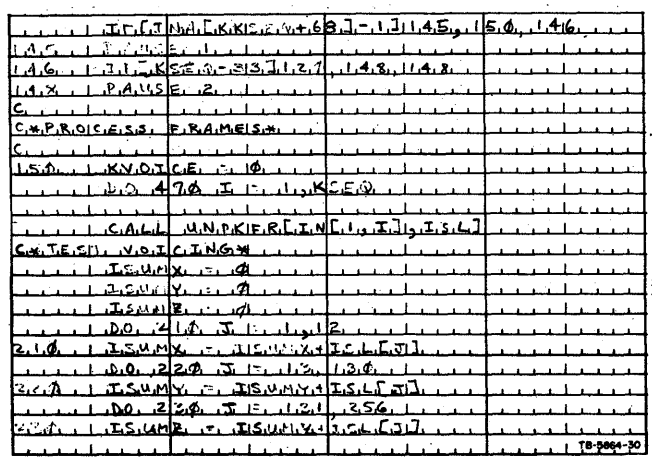

FIGURE 2—A sample of the single-coder test data

humans but not highly regular. Ten human subjects were asked to classify the test characters. The average error rate was 0.7%. Taking a plurality vote among the 10 responses, the error rate was 0.2% (2 errors in 1042 characters).

Scanning

The hand-printed characters were scanned from the source documents (the coding sheets) by a vidicon television camera fitted with a close-up lens and operated under the control of an SDS 910 computer. Each document was mounted in a concave cylindrical holder so that, as the camera panned across the document, the viewing distance and hence the image scale remained constant. The field of view was approximately one inch square. The camera generated a standard closed-circuit television waveform, which was quantized to two levels (black/white) by a Schmidt trigger and sampled in a raster of 120 × 120 points.

The document was illuminated by four floodlights mounted around the TV camera. A colored filter was placed over the camera lens, to suppress the colored coding-sheet guidelines appearing on the document. The guidelines could have been used for locating the characters, but we preferred to strive for a free-field character-locating procedure that could ultimately handle between-the-lines corrections or coding on a blank sheet of paper. Also, without a color-sensitive input system, separating the guidelines from the characters where they crossed or coincided could be a major problem.

The field of view was chosen so that a single character image was usually a little less than 24 points high and about 15 points wide. The computer began the scanning by reading in a 120 × 120 picture containing, in general, several character images. A scanning routine

then proceeded approximately horizontally through the picture, finding and isolating character images. Provisions were included for tracking a line of text, and for accepting multi-part character images such as equals signs and characters with unconnected crossbars. When the scanning routine got to the right of the 120 × 120 picture, it requested the camera to move to the right and input another picture.

As each character was isolated, it was placed in a standard 24 × 24 raster format (Figure 3). No corrections for magnification or rotation were applied. The BCD code of the character was entered manually

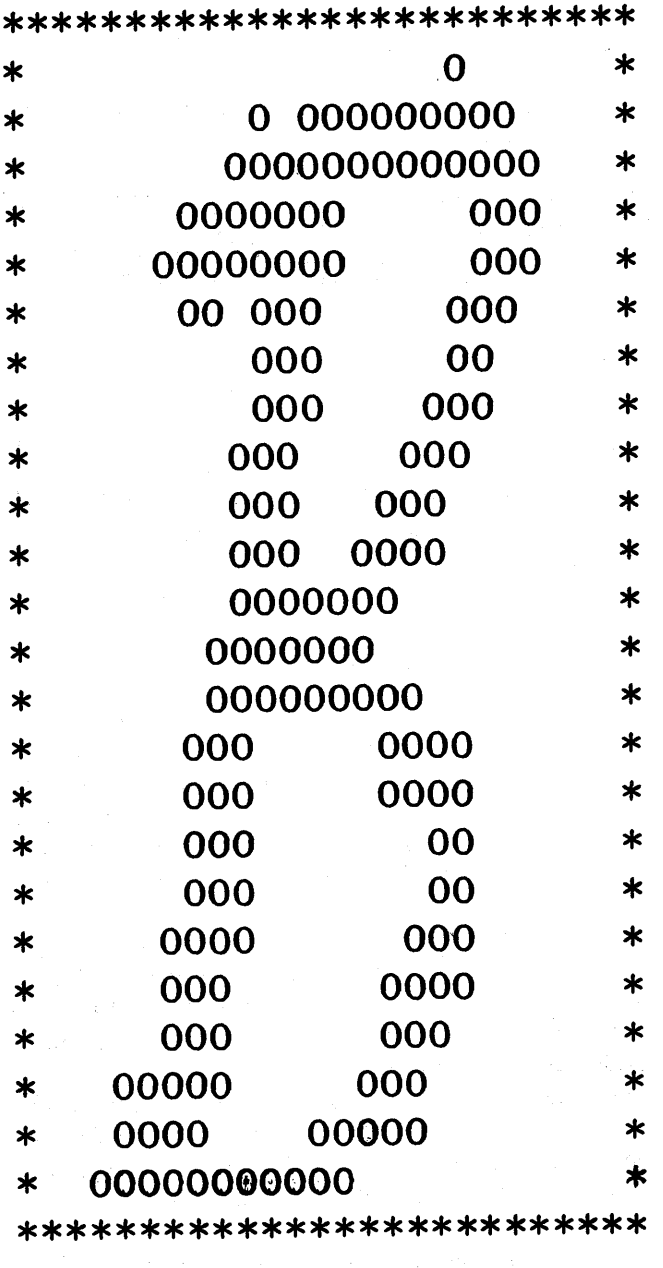

FIGURE 3—A hand-printed character in the standard 24 × 24 format

at the console typewriter and attached to the character record, for subsequent use in the training and testing procedures. The two files (single-coder and multiple-coder) of quantized 24 × 24 black/white character images served as the starting point for all subsequent processing. We hope to make these files available to other researchers through the efforts of the Subcommittee on Reference Data Sets of the Committee on Pattern Recognition of the IEEE Computer Group.

Our scanning setup was "strictly experimental." It was an inexpensive substitute for the sophisticated optical scanner and mechanical transport required for a high-volume production system. Although the scanning routine enabled us to gather the thousands of quantized characters in our data files, it was never capable of running without an attendant to rescue it from its errors. These were due to badly non-uniform sensitivity across the field of view (common in vidicon tubes), which made it impossible to set a single quantization threshold valid throughout the field, and to the lack of precise knowledge of the position of the TV camera. (Incidentally, by solving these problems, it should be possible to create a low-speed, inexpensive automatic scanning system along the lines of the one described above.)

Other files of digitized hand-printed data, supplied through the courtesy of W. Highleyman and researchers at IBM Corporation and Recognition Equipment, Inc., have been processed merely by converting them to our standard 24 × 24 format. In some cases, this has required changing the size of the character raster by copying or deleting rows and columns.

Preprocessing

The term "preprocessing" has acquired a variety of meanings. We use it here to refer to the specific activity of feature extraction: The calculation, from the (quantized) character image, of a set of numerical feature values that form the basis of subsequent pattern classification.

Two preprocessing methods were used in these experiments. The first, embodied in a computer program called PREP, was a simulation of a previously constructed optical preprocessor capable of extracting, in parallel, 1024 optical correlations between a character image and a set of photographic templates, or masks.[40] The second, a program called TOPO, extracted a large number of topological and geometric features of the character image.

The PREP preprocessor

The PREP program performed edge detection on the 24 × 24 quantized images through the use of

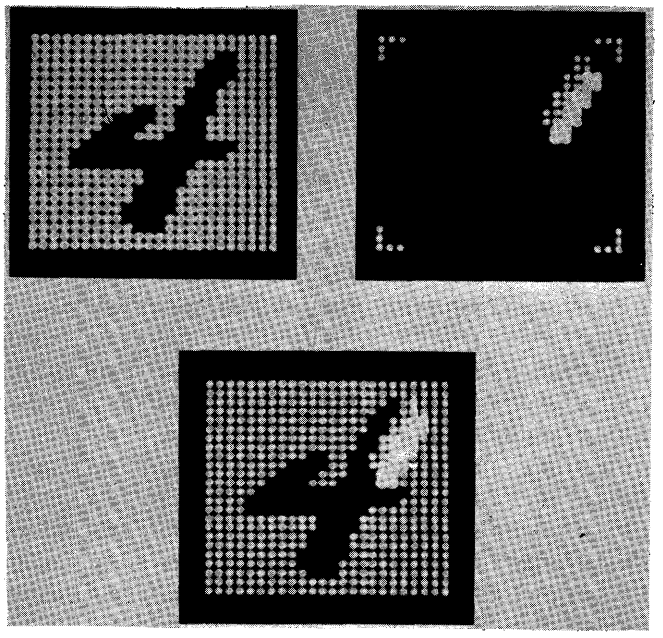

FIGURE 4—Edge-detecting masks in PREP
(a) Quantized character image
(b) An edge mask
(c) Character and mask together

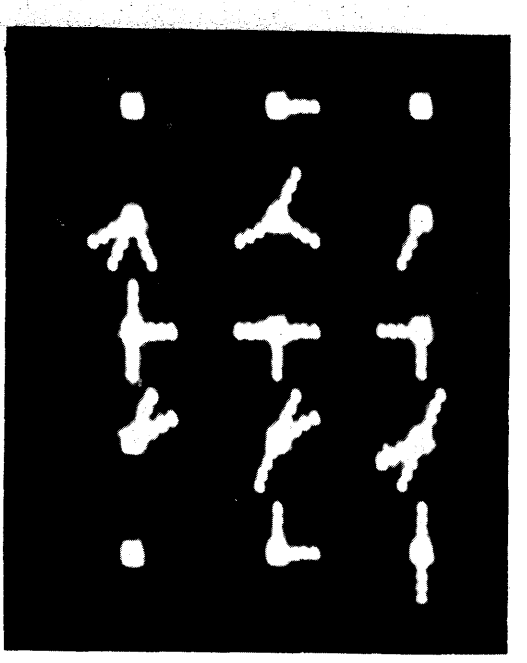

FIGURE 5—Responses of the PREP edge-detecting mask groups to a numeral "2"

edge-detecting mask pairs, or templates. Each mask pair consisted of two 2 × 8 rectangles of points, adjacent to each other along their long edges. One of the masks was given positive weight, the other, negative, and a threshold was set such that if the positive mask encountered six more figure points than the negative one, the binary response of the mask pair was ON (Figure 4).

To provide a limited degree of translation invariance, the responses of five such mask pairs were OR-ed together to give a single binary component of the output feature vector. The five mask pairs in a group had the same orientation and were in the same region of the 24 × 24 field. Nine regions were allotted to each of the four major compass directions, and six regions were allotted to each of the eight secondary directions (at 30° intervals). Thus, the complete feature vector consisted of 84 binary components, and the significance of a typical component was, "An edge oriented north-of-west has been detected in the left central region of the field." Figure 5 shows a computer display in which the lines are normal to edges detected in a sample of the numeral 2. The lines emanate from 15 loci representing the allotted regions.

Each quantized image was presented to the PREP preprocessor nine different times, first in the center of the 24 × 24 field, then in the eight positions formed by translating it vertically and/or horizontally by two units. Thus, for each pattern, a set of nine 84-bit feature vectors was formed. The use of these multiple-view feature vectors to improve classification performance is described below.

The TOPO preprocessor

The TOPO preprocessor was a sizable collection of computer routines assembled to extract topological and geometric features from the character image. In general, these features described the presence, size, location, and orientation of such entities as enclosures and concavities (lakes and bays) and stroke tips in the character.

TOPO began with a single connected character image in the 24 × 24 field. (The equals sign was sought out in advance, and treated as a special case. Other unconnected figures were forcibly joined by growing a bridge between the individual connected regions. If this failed, the lesser region(s) were discarded.) The *perimeter* of the figure was first found (Figure 6). The perimeter was defined as a list of figure points, beginning with the bottommost of the leftmost points of the character figure, found by stepping along the edge of the figure and keeping the figure always at the right hand and the ground (non figure) at the left. The perimeter has the property of including all figure points hand and the ground (non-figure) at the left. The peri-

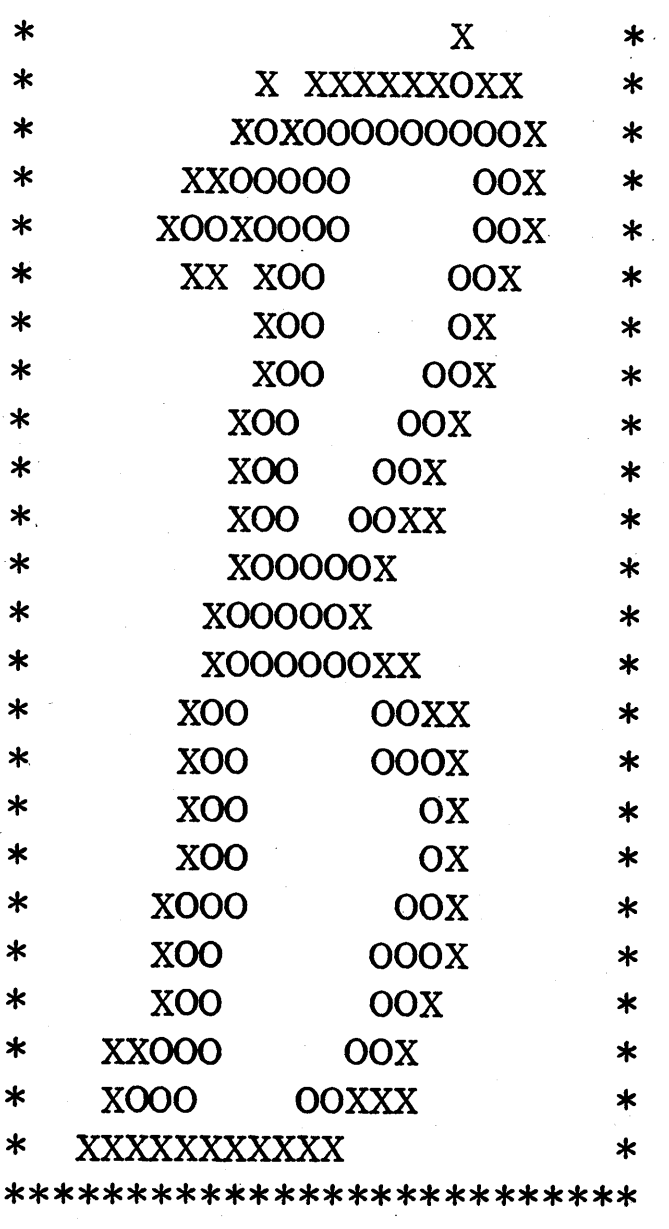

FIGURE 6—Hand-printed character with perimeter points marked X

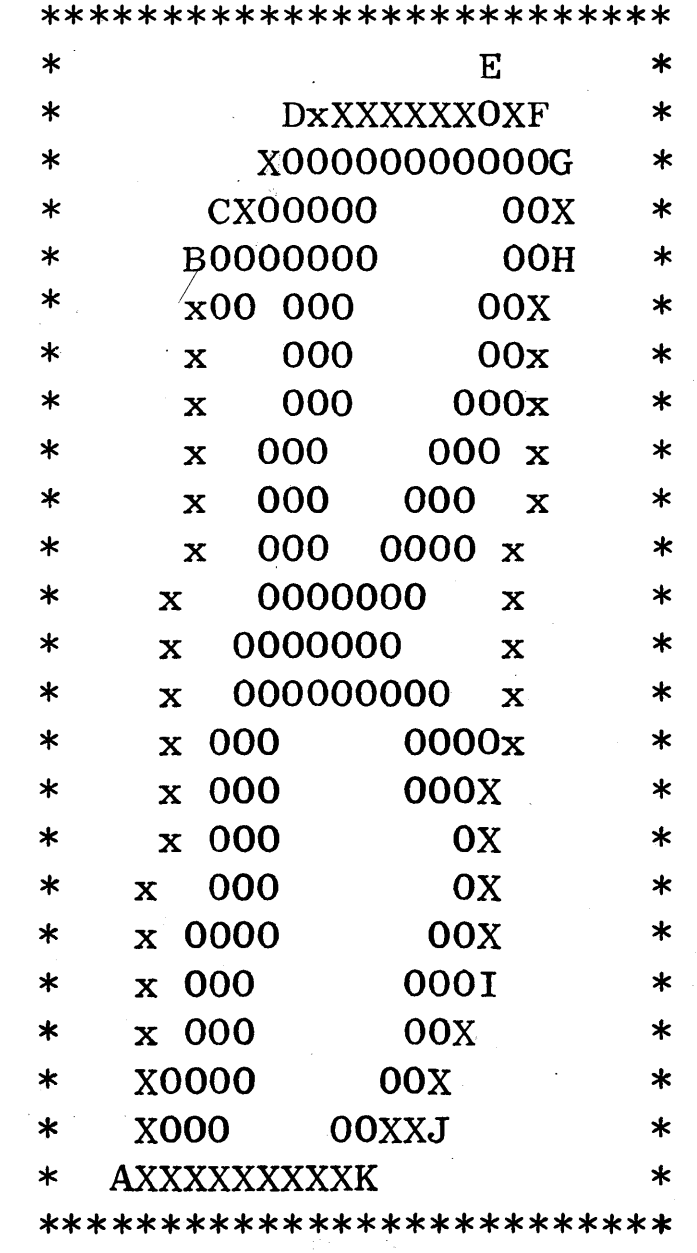

FIGURE 7—Character with convex hull boundary (CHB)

meter has the property of including all figure points that are 8-adjacent (adjacent horizontally, vertically, or diagonally) to ground points outside.

Next, the *convex hull boundary* (CHB) of the figure was found (Figure 7). The CHB of a two-dimensional figure may be thought of as the outline of a rubber band stretched around the figure. In the case of a quantized figure, some arbitrariness is required in the specification of the CHB, because a straight line between two points on the image grid does not generally fall on exact grid locations. We defined the CHB to include all the extremal points of the character image, represented by letters ot herthan "X" or "O" in Figure 7. In between these extremal points, the CHB was to follow as straight a path as possible, but never falling outside of the theoretical straight line connecting the extremal points. Keeping the CHB to the inside reduced the number of small, insignificant concavities found subsequently. To find the extremal points in the CHB, it was only necessary to search among those perimeter points at which the perimeter turned to the right.

After the CHB was obtained, the concavities and enclosures of the character image could be found quite readily using computer routines that simulated Boolean and connectivity operations performed in parallel over

the entire 24 × 24 field. Let the *border* consist of those ground points in the outermost rows and columns of the 24 × 24 field. Let an image be formed consisting of the ground, minus the CHB. The portion of this image that is not connected to the border lies within the CHB and consists of the concavities and enclosures of the character. Those regions that are connected to the border by a path of ground points (including ground points in the CHB) are concavities; those regions that are not are enclosures within the figure. The character in Figure 7 contains two concavities and two enclosures.

Multiple concavities and/or enclosures were extracted all at once in a single 24 × 24 array by the parallel operations. They were then separated (again using the connectivity operations) and sorted by size for subsequent use.

The *spurs* of a character are those strokes that end in an isolated tip. Ideally, the letter X has four spurs, the letter O, none, and the letter S, one spur with the special property of having a tip at each end. The list of perimeter points was used to find the spurs. Consider two pointers moving down the list of perimeter points, with one pointer ahead of the other by, say, 15 places. As the pointers moved, we calculated the Euclidean distance between the two perimeter points indicated by the pointers. Some of these distances are represented by arrows in Figure 8(a). Most of the time this distance would be approximately 15 units. A sudden decrease of the distance between the two points to a minimum that was less than half its usual value indicated that the perimeter had gone around a sharp bend—i.e., had gone around the tip of a spur. The position of the spur tip, indicated by the perimeter point halfway on the list between the two minimum-separation points, was the primary attribute of the spur used for forming features.

Once a spur was found, it could be traced by the "caliper method" [Figure 8(b)]. Imagine that the legs of a pair of calipers are placed at the two minimum-separation points. The calipers are then "slid" along the spur by stepping the legs of the calipers along the perimeter, away from the tip. The calipers are moved as far as they can go without having to be spread by more than, say, seven units. In some cases, such as the numeral "6," the calipers will be obstructed by the body of the figure and must stop. In other cases, such as the letter "S," the legs of the calipers will travel all the way along the figure and meet at the far end, indicating a "single-stroke" figure. The midpoint of the moving calipers traces out the backbone of the spur, and a list of the midpoint positions can be stored to represent the spur (the heavy line in Figure 8b).

Another set of character attributes found in TOPO

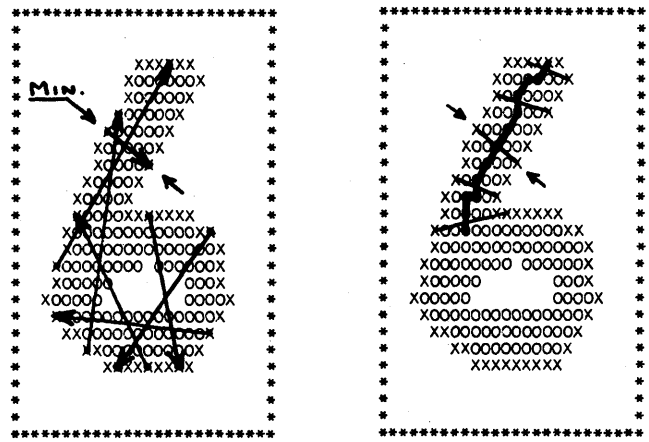

FIGURE 8—Spur-finding
(a) Finding the spur tip
(b) Tracing the spur

and used for feature generation were the *profiles* of the character image. The profiles were four lists, of 24 entries each, specifying the first row (or column) in which a figure point was encountered in each successive column (or row) as seen from the top, bottom, left, and right. The profiles were the basis of a number of specialized feature calculations, designed to discriminate among particular categories, that evaluated such properties as the width of the character at various levels, the number of reversals of direction in a profile, and discontinuities in the profiles.

Numerical feature calculation in TOPO

After the topological and geometric components of the character image—concavities, enclosures, spurs, profiles, etc.—were extracted, it remained to convert them to numerical components of a feature vector suitable for subsequent classification by an adaptive machine. This task was beset with several conceptual and practical difficulties that may not be obvious at first.

In TOPO, the task was carried out in two steps. First, *descriptors* (individual numerical quantities) were derived from the information at hand. Second, *features* in a standard form were calculated from the descriptors.

Each descriptor had to be chosen so that it always represented a unique characteristic of the character image. For example, suppose that one descriptor were to represent the vertical position of the rightmost spur tip. Such a descriptor would help to discriminate, for example, between T and L. But this descriptor would give unpredictable results for characters such as C,E, and [, depending on which spur extended farther to the

right, and would probably be detrimental to the classification of characters in these categories. In addition, there is the problem of vacuous descriptors: What value do we assign to the above descriptor in the case of a letter O?

In TOPO, these problems were countered by a careful choice of the definition of the descriptors. In many cases, it was possible to devise a descriptor that was always well defined. For example, if a spur-descriptor is put in the form, "To what extent is there a spur in the upper right-hand corner," it is defined for any number of spurs and can properly be given its minimum value for a figure with no spurs at all. In addition, this form of definition (unlike the preceding one) has the important property of *continuity*: Deformations of the character image that move the spurs by small amounts always cause small changes in the value of the descriptor. In another paper, the author has argued that the preservation of continuity is important throughout the various stages of the pattern-recognition process.[38]

As the final step in TOPO, the actual features (the numerical components of the feature vector for classification) were calculated from the descriptors. A first requirement on the features was that they be of comparable magnitudes, so that none would dominate the sums formed in the pattern classifier. Thus, the features were all given a standard range of zero to 100. (Note that these features were multiple-valued, whereas those from the PREP preprocessor were binary.)

A second, heuristic requirement on the features was that they emphasize the significant differences among character classes. In a two-category classification problem, it is feasible to analyze the discriminating power of a feature statistically (or even by inspection), and to adjust the transformation from descriptor to feature so as to maximize this power. In our 46-category problem, we could only guess at reasonable transformations. In any case, one should not expect the feature to be a simple linear function of a descriptor.

The derivation of features in TOPO may be indicated by an example. Consider a descriptor, MCONC(up), which is a measure of the presence of an upward-facing concavity in the character. For a flat-topped or round-topped character, such as T or O, MCONC(up) should have the value zero. For a character such as U or V, MCONC(up) should have a value of eight or greater. For a Y or an open-topped 4, however, we should only expect values of five or greater. Owing to the linear nature of the dot-product units used in the pattern classifier, it is impossible for a single feature proportional to MCONC(up) to discriminate between Y and T, for example, without treating U as a "super-Y." We actually require *two* features—one that "switches" in the range 0 to 5 and one that does so at a higher range.

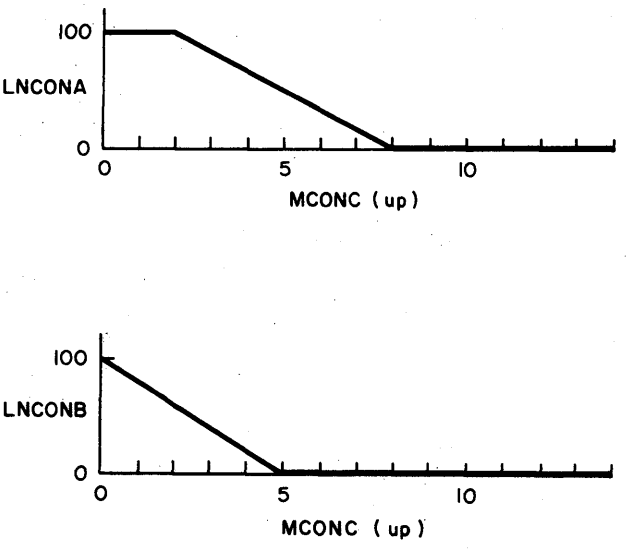

FIGURE 9—Two transformations that derive features from a concavity descriptor

The two transformations that derived features from MCONC(up) in TOPO are shown in Figure 9. There were two such features corresponding to each spur descriptor and concavity descriptor in TOPO. In all, TOPO produced 68 features: 16 for the spurs, 16 for the concavities, 8 for the enclosures, 6 for overall character size and shape, and 22 resulting from special calculations about the width of the character at various levels, discontinuities in the profiles, etc. Each feature was calculated from a numerical descriptor by a transformation arrived at by inspection.

It should be evident from the foregoing description that the development of TOPO was a cut-and-try affair. The extraction of topological entities and the generation of descriptors and features were continued only as far as patience permitted. For example, a feature to look for structure within an enclosure and help discriminate between O and Q was never implemented. It is the author's opinion that the generation and selection of features for pattern classification, especially in the multi-category case, is the greatest problem area in pattern recognition at the present.[38]

Classification

An adaptive pattern classifier, or learning machine, was used to classify the characters on the basis of the feature vectors generated by a preprocessor, either PREP or TOPO. The learning machine was of the piecewise linear (PWL) type, described by Nilsson.[41]

The learning machine for these experiments was implemented by a computer program called CALM (Collected Algorithms for Learning Machines),[42] running on the SDS 910 computer, which simulated the action of the MINOS II hardware learning machine constructed earlier during this project.[40,43,44]

Briefly, a learning machine embodies a set of Dot Product Units (DPU's) that form the *dot product* (also called the inner product or vector product) between the incoming pattern, or feature vector, and a set of stored *weights*. The j^{th} DPU of the machine forms the dot product

$$S_j = X \cdot W_j = \sum x_i w_{ij}$$

between the pattern vector X and the weight vector W_j associated with the j^{th} DPU. In a PWL learning machine, a small number of DPU's are assigned to each of the 46 character categories. The largest dot product formed among the DPU's assigned to a category is taken as the *response* for that category.

The category responses may be utilized in two ways. If it is desired to explicitly categorize a character, the character is assigned to the category with the largest response. A *testing margin* or *dead zone* may be employed, so that any character for which the largest response does not exceed the second largest by the margin is classed as a reject. In the performance results listed below, the reject margin is not used. The performance scores are thus of the simplest possible type: percentage of successful classifications with no rejects allowed (response ties are broken arbitrarily).

Alternatively, if the goal is not to achieve a succinct performance measure but rather to use the character-classification information for contextual analysis, the responses may be used to obtain *confidence information*. The simplest confidence measure is the set of 46 responses from the learning machine, with a higher response indicating a higher confidence that the character belonged to the category in question.

To adapt a learning machine, a *training pattern* is presented, and the responses to that pattern are obtained. If the response in the true category of the training pattern does not exceed the largest response among the other categories by a value called the *training margin*, the DPU yielding the response in the true category is marked to be *incremented*, and that yielding the competing response is marked to be *decremented*. This is done by setting

$$a_{true} = 1 \quad ; \quad a_{competing} = -1$$

in the *adapt vector* A, and setting all the other components of A to zero. Adaptation of the weights is then performed according to the *fixed-increment error correction rule*:

$$W_j \leftarrow W_j + a_j \cdot D \cdot X, \quad \text{for all j.}$$

In other words, the pattern vector is added to or subtracted from the j^{th} weight vector, depending on a_j. D is an overall multiplying factor called the *adapt step size*, usually set to a small integer throughout a block of training. (Other methods of determining the responses and A and D lead to learning machines other than the PWL machine.)[41,42]

The adaptation causes the subsequent dot product between the pattern vector and the weight vector to be changed by an amount

$$\bar{\Delta}S_j = X \cdot (a_j \cdot D \cdot X) = a_j D |X|^2.$$

Since X^2 and D are always positive, the sign of a_j automatically determines whether the response (i.e., the dot product) of the j^{th} DPU with the pattern X is enhanced or reduced. Through this means, appropriate DPU's can be made to respond to certain patterns and ultimately to classes of patterns.

To perform a learning-machine experiment, the adaptive weights w_{ij} are initialized, usually to zero. The training patterns are then presented sequentially. The responses to each training pattern are formed, and if the classification is incorrect the machine is trained. One pass through the training patterns is called an *iteration*. Typically, repeated iterations through the training set are performed until the classification performance on the training patterns ceases to improve. At that time, the test patterns may be presented, and the classification performance on them recorded. This performance is generally taken as the measure of success of the learning machine on the task represented by the training and test patterns.

In dealing with the nine-view sets of feature vectors produced by the PREP preprocessor, the running procedure was modified slightly (Figure 10). During training, one of the nine feature vectors representing a training pattern was selected quasi-randomly at each iteration. Thus, it took nine iterations for the machine to encounter each view of each pattern. The use of multiple views had the effect of "broadening" the training experience of the learning machine. During "nine-view testing," all nine views of each test pattern were presented and the nine responses in each category were added together to form cumulative responses that were used as the basis for classification. It will be seen that the redundancy achieved by accumulating the nine responses led to a significant improvement in performance. This technique has also been used successfully

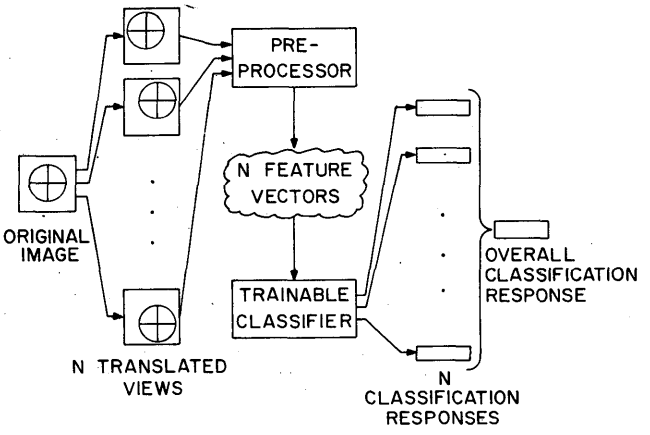

FIGURE 10—Multiple-view testing procedure

by Darling and Joseph in the processing of satellite photographs.[45]

Experimental results

A series of experiments were performed on the single-coder and multiple-coder data files, using the preprocessors and learning machine described above. Experiments were run under four conditions.

In Condition 1, the characters were preprocessed by PREP, but only the one feature vector representing the central view of each pattern was used for training and testing the learning machine. Only the single-coder file was run under Condition 1.

In Condition 2, the characters were preprocessed by PREP in all nine views, and nine-view training and testing were performed as described above. A PWL learning machine with two DPU's per category was used in Conditions 1 and 2.

In Condition 3, the characters were preprocessed by TOPO, and the single feature vectors produced by TOPO were used for training and testing. Owing to computer restrictions, a learning machine with only one DPU per category was used. This is generally called a linear rather than a PWL learning machine, after the form of the discriminant functions in feature space.[41]

In Condition 4, the responses of the learning machines in Conditions 2 and 3 for each test pattern were added together and taken as a new basis for classification. This procedure was a way of harnessing the preprocessor-classifier systems of Conditions 2 and 3 "in tandem" in order to improve classification performance, in a manner analogous to the nine-view testing of the PREP feature vectors.

The results of the experiments are presented in Table I. The results show a significant improvement in performance for the case of nine-view training and testing over single-view training and testing, and further improvement with the combined system of Condition 4. The most important results can be summarized as follows:

Using the combined system, a correct character-classification rate of 97% (with no rejects) was obtained on independent test of relatively unconstrained hand printing in the 46-character FORTRAN alphabet, when the learning machine was allowed to specialize on data from a single coder. When the learning machine was trained on the printing of 32 coders and tested on the printing of 17 others, the correct classification rate was only 85%. These rates are for the isolated characters, without context.

Condition	Preprocessor	Number of Iterations	Final Classification Scores	
			Training Patterns	Test Patterns
Single-Coder File				
1	PREP, 1 view	10	99%	88%
2	PREP, 9 views	27	89%*	96%
3	TOPO	10	94%	91%
4	Combined	--	---	97%
Multiple-Coder File				
2	PREP, 9 views	18	65%*	78%
3	TOPO	4	84%	77%
4	Combined	--	---	85%

*Single-view classification scores

TABLE I—Experimental results on two files of hand-printed alphanumeric characters

A well known set of quantized hand-printed character images (letters and numerals only) collected by Highleyman were also processed under Condition 2, yielding a test classification score of 68%. Previously reported classification methods, not employing preprocessing, had achieved scores of 58% or less. These characters are of very poor quality, being only 86% to 89% classifiable by humans. These results are described in Ref. 39.

A large number of preliminary and auxiliary experiments, not described in this paper, were performed. In particular, during the development of the TOPO preprocessor, an attempt was made to use the features produced by TOPO in a binary decision-tree classifier. The results of this effort were very poor, because it was impossible to find features reliable enough to serve for dichotomization of the character classes. For example,

the presence of an enclosure was a useless feature, because quantization noise introduced some spurious enclosures, and other expected ones were lost because they were filled in or not completely formed. It thus appears to us that, for patterns with the variability of hand printing, an approach that considers all the features in parallel is a necessity.

The development of the TOPO preprocessor, the exploration of variations of the PREP preprocessor, and the running of classification experiments with different learning-machine configurations and different data files were all severely restricted by system limitations of two types. First, it was awkward to handle the necessary large data files on our computer, which had paper tape input/output and one magnetic tape unit. Second, all of the operations of scanning, preprocessing, and classification were performed in serial or at most in 24-bit parallel by simulations in the computer (an SDS 910, with 12K words of 24-bit memory and an 8-μsec cycle time). It sometimes took days to accomplish the experimental runs. However, the flexibility, guaranteed reproducibility, and operational advantages of computer simulation compensated for the drawbacks. With a view toward practical systems, it may be noted that the operations of PREP and the learning machine, and even much of TOPO, are extremely well suited for implementation in parallel hardware.

CONCLUSION

This paper has been concerned solely with the classification of individual hand-printed characters, in isolation. To this end, performance scores based on the positive classification of each character have been presented, as the simplest and most understandable measure of performance. The end goal, however, is text recognition, not character recognition *per se*. The results presented here, including the scores for classification by humans, indicate that context analysis will be a necessary adjunct to character classification for the recognition of fairly unconstrained, untutored hand-printed text. The companion paper by Duda and Hart[37] describes our effort in context analysis. To assist context analysis, the classifier puts out not explicit classifications but lists of alternate categories and their confidences for each character. This may be viewed as a way of retaining additional valuable information generated by the classifier, under a continuous transformation (from feature space, through the learning machine, to the space of category confidence values). This preservation of alternative information is a vital aspect of context-aided text recognition, not to our knowledge previously discussed in the character-recognition literature.

In experimenting with a large body of actual data from an untutored hand-coding population, we are attempting to set a benchmark in an area not covered by the experiments in the former literature, nor by the present commercial developments. At the current levels of available computing power and cost, contextual analysis appears economically infeasible, at least for syntactically rich texts such as FORTRAN. However, if the progress of OCR is a guide, we may expect a pressure for the extension of recognition systems for hand printing to accept input from broader and less highly trained classes of coders. The character-recognition methods described in this paper and the context-analysis procedures described by Duda and Hart may then point toward systems of future importance.

ACKNOWLEDGMENTS

The author wishes to express his gratitude to the many people in the Artificial Intelligence Group at Stanford Research Institute whose work during the past several years has paved the way for this research, especially Dr. Richard O. Duda and Dr. Peter E. Hart.

Support for this work has come from the United States Army Electronics Command, Fort Monmouth, New Jersey, under Contract DA 28-043 AMC-01901(E).

REFERENCES

1 G P DINNEEN
Programming pattern recognition
Proc 1955 WJCC pp 94–100

2 T L DIMOND
Devices for reading handwritten characters
Proc 1957 EJCC pp 232–237

3 W W BLEDSOE I BROWNING
Pattern recognition and reading by machine
Proc 1959 EJCC pp 225–232 Also in 32

4 J S BOMBA
Alpha-numeric character recognition using local operations
Proc 1959 EJCC pp 218–224

5 L A KAMENTSKY
Pattern and character recognition systems—picture processing by nets of neuron-like elements
Proc 1959 WJCC pp 304–309

6 R L GRIMSDALE F H SUMMER C J TUNIS T KILBURN
A system for the atuomatic recognition of patterns
Proc IEE Vol 106B No 26 pp 210–221 March 1959 Also in 32

7 S H UNGER
Pattern detection and recognition
Proc IRE pp 1737–1752 October 1959

8 L G ROBERTS
Pattern recognition with an adaptive network
IRE 1960 International Convention record pp 66–70 Also in 32

9 W DOYLE
Recognition of sloppy hand-printed characters
Proc 1960 WJCC pp 133–142

10 U NEISSER P WEENE
A note on human recognition of hand-printed characters
Information and Control Vol 3 pp 191–196 1960
11 L UHR C VOSSLER
A pattern-recognition program that generates evaluates and adjusts its own operators
Proc 1961 WJCC pp 555–569 Also in *Computers and Thought* Feigenbaum and Feldman Eds pp 251–268 McGraw-Hill Book Co New York N Y 1963 and in 32
12 J R SINGER
A self-organizing recognition system
Proc 1961 WJCC pp 545–554
13 W H HIGHLEYMAN
An analog method for character recognition
IRE Trans on Electronic Computers Vol EC–10 pp 502–512 September 1961
14 L A KAMENTSKY
The simulation of three machines which read rows of handwritten arabic numbers
IRE Trans on Electronic Computers Vol EC–10 No 3 pp 489–501 September 1961
15 P M LEWIS II
The characteristic selection problem in recognition systems
IRE Trans on Information Theory Vol IT–8 pp 171–178 February 1962
16 C K CHOW
A recognition method using neighbor dependence
IRE Trans on Electronic Computers Vol EC–11 No 5 pp 683–690 October 1962
17 W H HIGHLEYMAN
Linear decision functions with application to pattern recognition
Proc IRE Vol 50 No 6 pp 1501–1514 June 1962
18 J L MASTERSON R S HIRSCH
Machine recognition of constrained handwritten arabic numbers
IRE Trans on Human Factors in Electronics Vol HF–3 p 62 September 1962
19 F KUHL
Classification and recognition of hand-printed characters
IEEE 1963 International Convention Record Part 4 pp 75–93 March 1963
20 E C GREANIAS et al
The recognition of handwritten numerals by contour analysis
IBM Journal Vol 7 No 1 pp 14–21 January 1963
21 G U UYEHARA
A stream-following technique for use in character recognition
IEEE 1963 International Convention Record Part 4 pp 64–74 March 1963
22 J A BRADSHAW
Letter recognition using a captive scan
IEEE Trans on Electronic Computers Vol EC–12 No 1 p 26 February 1963
23 D M STERN D W C SHEN
Character recognition by context-dependent transformations
Proc IEE Vol 111 No 11 pp 1923–1931 November 1964
24 W TEITELMAN
Real-time recognition of hand-drawn characters
Proc 1964 FJCC pp 559–575
25 N LINDGREN
Machine recognition of human language part III-cursive script recognition
IEEE Spectrum pp 104–116 May 1965
26 R J SPINRAD
Machine recognition of hand printing
Information and Control Vol 8 pp 124–142 1965
27 H A GLUCKSMAN
A parapropagation pattern classifier
IEEE Trans on Electronic Computers pp 434–443 June 1965
28 J K CLEMENS
Optical character recognition for reading machine applications
Mass Institute of Technology PhD Thesis 1965
29 C K CHOW C N LIU
An approach to structure adaptation in pattern recognition
IEEE Trans on Systems Science and Cybernetics Vol SSC–2 No 2 pp 73–80 December 1966
30 M J MINNEMAN
Handwritten character recognition employing topology cross correlation and decision theory
IEEE Trans on Systems Science and Cybernetics Vol SSC–2 No 2 pp 86–96 (December 1966
31 M M CHODROW W A BIVONA G M WALSH
A study of handprinted character recognition techniques
Technical Report No RADC–TR–65–444 Contract No AF 30 (602)–3721 Information Dynamics Corp Reading Mass February 1966
32 L UHR
Pattern recognition
John Wiley & Sons New York N Y 1966
33 G GRONER
Real-time recognition of hand-printed text
Proc 1966 FJCC pp 591–601
34 J G SIMEK C J TUNIS
Handprinting input device for computer systems
IEEE Spectrum pp 72–81 July 1976
35 R BAKIS et al
An experimental study of machine recognition of hand-printed numerals
Research Report RC–1778 IBM Research Center Yorktown Heights N Y February 1967
36 J H MUNSON
The recognition of hand-printed text
In Pattern Recognition L Kanal Ed Thompson Book Co Washington DC 1968
37 R O DUDA P E HART
Experiments in the recognition of hand-printed text
Part II—context analysis
in this volume
38 J H MUNSON
Some views on pattern-recognition methodology
To be published in the Proc International Conference on Methodologies of Pattern Recognition University of Hawaii Honolulu 1968
39 J H MUNSON et al
Experiments with Highleyman's data
IEEE Trans on Computer Vol C–14 No 4 pp 399–401 April 1968
40 A E BRAIN J H MUNSON
Graphical-data-processing research study and experimental investigation
Final Report Contract DA 36–039 AMC–03247(E) SRI Project ESU 4565 Stanford Research Institute Menlo Park Calif April 1966 Astia Document No AD 632 563
41 N J NILSSON
Learning machines
McGraw-Hill Book Co New York N Y 1965
42 J H MUNSON
User's manual for the CALM learning-machine simulation program
Technical Report Contract AF 30 (602)–4216 SRI Project ESU 6021 Stanford Research Institute Menlo Park California

November 1966 Astia Document No AD 648 959

43 J H MUNSON et al
A pattern-recognition facility with a computer controlled learning machine
Proc IFIP Congress 65 Vol II pp 360–361 May 1965

44 *Graphical-data-processing research study and experimental investigation*
Quarterly Technical Reports ECOM–01901–23 thru ECOM–01901–30 SRI Project ESU 5864 Standard Research Institute Menlo Park California July 1966 thru May 1968

45 E M DARLING JR G D JOSEPH
Pattern recognition from sattellite altitudes
IEEE Trans on Systems Science and Cybernetics Vol SSC–4 No 1 pp 38–47 March 1968