Algorithms for Recognizing Contour-Traced Handprinted Characters  
GODFRIED T. TOUSSAINT AND ROBERT W. DONALDSON

Abstract—A contour-tracing technique originally devised by Clemens and Mason was modified and used with several different classifiers to recognize upper case handprinted characters. Recognition accuracies obtained compare favorably with other published results, particularly when additional simple tests are performed to differentiate commonly confused characters. One suboptimum classifier, in addition to yielding near optimum performance when tested on training data, uses much less statistical information than the optimum Bayes classifier and is significantly better than the optimum classifier when training and test data are limited.

Index Terms—Character recognition, classification algorithms, contextual constraints, contour analysis, decision trees, feature extraction, handprinted characters, limited-data experiments, machine learning, suboptimum classification.

1. Introduction

Feature extraction and classification are common to all pattern recognition schemes. Let the features from pattern $C_i$ be described by a vector $X = (x_1, x_2, \ldots, x_D)$. If $P(C_i)$ and $P(X|C_i)$ denote, respectively, the probability of $C_i$ and the probability of $X$ conditioned on $C_i$, then the probability of misclassification is minimized by choosing $i$ to maximize any monotone increasing function of

$$R_i = P(X|C_i)P(C_i) = P(x_1, x_2, \ldots, x_D|C_i)P(C_i).$$ (1)

In most character recognition schemes the dimensionality $D$ of $X$ is large, typically $D > 50$. Even if the components of $X$ are binary, learning the statistics of $2^D$ different probabilities $P(X|C_i)$ for each $i$ requires many training samples and much storage capacity. When the $D$ components $x_k$ of $X$ are statistically independent, however,

$$P(X|C_i) = \prod_{k=1}^{D} P(x_k|C_i),$$ (2)

in which case it is necessary to learn and store only the $D$ terms $P(x_k|C_i)$. Two possibilities now arise. a) If the components $x_k$ are not strongly interdependent, and if there are insufficient samples to train $P(X|C_i)$ sufficiently well, it may be best to assume (2), or to use some other suboptimum classifier which can be trained on fewer samples than are needed to train $P(X|C_i)$.
b) Even when $P(X|C_i)$ is known exactly, a suboptimum classifier which requires less storage capacity than the optimum classifier may be desirable. Unused storage can be occupied by contextual constraints and sophisticated decoding algorithms, both of which can effect considerable improvement in recognition accuracy [3]–[6]. Thus, feature extraction schemes which retain those features essential for recognition while keeping $D$ small are attractive, because the relative independence of the $x_i$’s encourages the use of (2).

The feature extraction algorithm used in our work is based on two observations. First, reasonably legible Roman characters are recognizable solely on the basis of their external contour, as can be demonstrated by printing upper and lower case alphabets and darkening all areas completely surrounded by black. Second, confusion between characters is not random, but highly structured. Accordingly, a character’s outside contour was used to generate a binary vector $X$ which was then classified either as a single character, or as one of a group of characters not easily distinguished solely on the basis of $X$. In this latter case, simple class-dependent tests on the unknown character effected final recognition. The transformation of character contours into binary vectors is similar to one used by Clemens and Mason [7]–[10] for recognizing typewritten characters. In addition to being easy to implement, the transformation yields a feature vector whose dimensionality is, on the average, considerably smaller than that of most other recognition schemes.

Our results, based on a character set of reasonably good quality, compare favorably with those obtained by others (Knoll [11] presents a summary of the results of others). Application of (2) to the recognition of 14 upper
case alphabetic characters from seven persons yielded 90 percent accuracy when the same data were used to train and test the classifier; when ten alphabets were used for training and the remaining four for testing, the recognition accuracy was 73 percent. Both results, which apply to statistically independent equiprobable characters from seven individuals, improve significantly if simple additional tests are performed to distinguish between easily confused characters, if the characters occur with first-order probabilities of English text, or if the classifier is trained on characters from one individual and later tested on (different) characters from the same individual. Results for several suboptimum classification algorithms appear in Section V.

II. DESCRIPTION OF THE DATA

A total of twenty upper case alphabets was obtained from seven different persons, all of whom were required to print on good quality graph paper in squares 0.40 inch high by 0.50 inch wide. A Paper Mate thin felt-tip pen was used. Each person was asked to leave the outside contour of each character unbroken. All seven persons printed the alphabet once from A to Z, as shown in Fig. 1, and once from Z to A. Two persons printed three additional alphabets by copying three randomized lower case alphabets. An enlarger-projector of magnification ten was used to project each character onto an area divided into 2500 0.1-inch squares. If more than 50 percent of any square’s area was black, the square was blackened and assigned to value 1. Otherwise the square was left white and assigned a value 0. Each spatially quantized character was then encoded onto IBM punch cards by experienced keypunch operators.

The legibility of the data was estimated by asking ten graduate students to identify all 520 spatially quantized characters. Each student responded verbally to characters viewed one at a time at close range for as long as desired. All viewing was done at one session which lasted approximately 12 minutes per student. The recognition accuracy averaged over the first two alphabets of all seven persons and over all ten viewers was 99.7 percent. Averaging over all ten viewers and over the five alphabets from persons 1 and 2 yielded 99.8 and 100 percent, respectively. Machine recognition errors in excess of a few percent would, therefore, result from weaknesses in either feature extraction or classification, or both.

III. TRANSFORMATION OF CHARACTER CONTOURS INTO VECTORS

Fig. 2 illustrates the algorithm for scanning the outside contour of a character. The scanner was simulated using an IBM 7044 computer, into which punched cards containing the data had been read. The scanning spot moves, point by point, from the bottom to the top of the leftmost column, and successively repeats this procedure on the column immediately to the right of the column previously scanned, until the first black point is found. Upon locating this point, the scanner enters the contour mode, in which the scanning spot moves right after encountering a white point and left after encountering a black point. The contour mode terminates when the scanning spot completes its trace around the outside of a character and returns to its starting point.

After being scanned the character is divided into either four or six equal-sized rectangles whose size depends on the height and width of the letter (see Fig. 3). A y threshold equal to one half each rectangle’s height and an x threshold equal to one half each rectangle’s width is defined. Whenever the x coordinate of the scanning spot reaches a local extremum and moves in the opposite direction to a point one threshold away from the resulting extremum, the resulting point is designated as either an \(x_{\text{max}}\) or \(x_{\text{min}}\). After an \(x_{\text{max}}(x_{\text{min}})\) has occurred, no additional \(x_{\text{max}}\)’s (\(x_{\text{min}}\)’s) are recorded until after an \(x_{\text{min}}(x_{\text{max}})\) has occurred. Analogous comments apply to the y coordinate of the scanning spot. The starting point of the contour mode is regarded as an \(x_{\text{min}}\). The code word for a character consists of a 1 followed by binary digits whose order coincides with the order in which extrema occur during contour tracing;
1 denotes max’s and min’s in x, while 0 denotes max’s and min’s in y. The rectangles are designated by binary numbers, and the ordering of these numbers in accordance with the rectangles in which extrema fall in event sequence constitutes the COORD word. The feature vector consists of the CODE word followed by the COORD word. Fig. 3 shows CODE and COORD words for “C.” Further details relating to CODE and COORD word generation appear elsewhere [7]–[10].

We note here that the rectangles in Fig. 3 are numbered in such a way that rectangles far away from one another differ in more binary digits than do rectangles close to each other.

IV. Classification Algorithms

In classification algorithm S, \( P(X \mid C_i) \) was estimated from data, and unknown characters were then classified by choosing \( i \) to maximize \( R_i \) in (1). In algorithm \( T \), which follows from (1) and (2) when these equations are suitably modified to account for the fact that the length \( D \) of the feature vector is variable, \( i \) was chosen to maximize

\[
T_i = \sum_{k=1}^{D} \log_e P(x_k \mid C_i, D) + \log_e P(D \mid C_i)
+ \log_e P(C_i). \tag{3}
\]

In algorithms \( U \) and \( V \), which are minimum decoding distance algorithms, \( i \) was chosen to minimize

\[
U_i = \sum_{k=1}^{D} \left[ x_k - P(x_k = 1 \mid C_i, D) \right] - \log_e P(C_i) \tag{4}
\]

\[
V_i = \sum_{k=1}^{D} \left[ x_k - P(x_k = 1 \mid C_i, D) \right]^* - \log_e P(C_i). \tag{5}
\]

Probability \( P(x_k = 1 \mid C_i, D) \) is equal to the mean of \( x_k \) conditioned on \( C_i \) and \( D \). When all characters are equiprobable and when the \( x_k \)’s are binary, \( U_i \) is identical to \( \hat{T}_i = \sum_{k=1}^{D} P(x_k \mid C_i, D) \) which is obtained by omitting the last two terms from (3) and taking the antilogarithm of all remaining terms. It follows that \( U \) will never yield a lower error probability than \( T \) when the characters are equiprobable and binary vector components \( x_k \) are statistically independent. Algorithms \( U \) and \( V \) are optimum under certain conditions; the interested reader is referred elsewhere for details [12], [13].

Let \( M \) be the maximum value of \( D \). In algorithms \( W \) and \( Y \), vectors of length \( D < M \) were made equal to \( M \) by setting \( x_k = 0 \) for \( D < k < M \). Index \( i \) was then chosen to maximize \( W_i \) and minimize \( Y_i \), where

\[
W_i = \sum_{k=1}^{M} \log_e P(x_k \mid C_i) + \log_e P(C_i)
\]

\[
Y_i = \sum_{k=1}^{M} \left[ x_k - P(x_k = 1 \mid C_i) \right] - \log_e P(C_i). \tag{7}
\]

Probabilities \( P(X \mid C_i) \), \( P(x_k \mid C_i) \), and \( P(x_k \mid C_i, D) \) were learned by determining the relative number of times a vector \( X \) or component \( x_k \) occurred, given the event \( C_i \) or the joint event \( C_i, D \). When an experimentally determined probability \( P = 0 \), we set \( P = 1/(n+2) \) [14], where \( n \) is the number of times \( C_i \) or \( C_i, D \) occurred during training.

V. Results

Results obtained using the different recognition algorithms appear in Fig. 4. The caption in Fig. 4 explains the way in which the tree is used to determine the error probability of any given recognition scheme. When the test data were different from the training data the population (POP) results were obtained using the first two alphabets from five persons for training, and the first two alphabets from each of the two remaining persons for testing. The results were averaged over seven trials. In the \( i \)th trial, data from persons \( i \) and \( i+1 \) (\( i = 1, 2, \ldots, 6 \)) were used for testing. In the seventh trial, data from persons 7 and 1 were used for testing. For tests on the data from one individual, four alphabets were used for training and the remaining one for testing. The results are averaged over five trials in which a different alphabet served as test data for each trial. These methods of training and testing are not unlike the \( U \) method recommended elsewhere [2]. Although the \( U \) method would probably predict better performance than does ours, the amount of computation needed for training would be prohibitive in our case. When the same alphabets were used for both training and testing the first two alphabets from each person were used in the POP tests. In tests on alphabets from persons 1 and 2, all five alphabets were used for both training and testing.

Over POP’s 14 alphabets the average vector length
was 17 and 25 for the four- and six-part area divisions, respectively.

The actual performance obtained when large disjoint training and test sets are used would likely lie somewhere between that shown for the TS and TR cases in Fig. 4. Results obtained when the same characters were used to both train and test the classifier are included for this reason, and because direct comparison of our TS results with those of other workers is made difficult by the fact that results depend on the actual partitioning of the characters into training and test sets when both sets are small [2], [15].

The procedure used to obtain finite estimates of $P(X | C_i)$ and $P(C_i | X, D)$ makes rejection of any character unnecessary; for example, when $P(X | C_i)P(C_i) = 1/n + 2$ for all $i$, the optimum decision is to select any one of the 26 characters as being correct. The numbers in parentheses in Fig. 4 merely indicate the probability that the optimum choice is any one of the 26 characters; in many recognition schemes, however, the unknown character would be rejected in such cases. Thus, rejects are indicated for algorithm $S$ whenever a feature vector different from any encountered during training occurs during testing, and for $T$, $U$, and $V$ whenever the length of a test vector is different from any lengths encountered during training.

Some scans used to differentiate easily confused characters appear in Fig. 5(a). Use of these scans and the decision trees in Fig. 5(b), when training characters were initially classified using algorithm $U$, reduced error probability for POP from 24 to 10 percent, as Fig. 4 shows. The ABD decision tree reduced the error for person 2 from 8.5 to 0.8 percent (see Fig. 5(c)).
tests similar to those indicated in Fig. 5 yielded the other numbers in square brackets in Fig. 4, although fewer scans, decision trees, and branches per tree were needed in these other cases, which are not discussed in detail here.

VI. DISCUSSION

Although algorithm $S$ yielded the lowest error probability when the training and test data were identical, the difference between $S$ and $T$ is small. In fact, when characters occurred with first-order probabilities of English text and when $T$ made use of this contextual data, better results were obtained than when algorithm $S$ was used on equiprobable characters, even though $P(X|C_i)$, needed for algorithm $S$, requires considerably more storage capacity than does $P(x_k|D, C_i)P(D|C_i)$·$P(C_i)$ for $T$. When storage capacity is limited, a trade-off between contextual constraints and measurement statistics is clearly indicated; suboptimum classifiers which use fewer measurement statistics and more contextual data may well be superior to optimum classifiers which make little or no use of existing contextual constraints.

Algorithms $T$, $U$, and $V$ all performed much better than $S$ when training and test data were disjoint, which indicates that in such circumstances suboptimum classifiers having relatively few parameters may be significantly better than optimum classifiers requiring extensive training. This conclusion is not unlike one reached by Hughes [16], which was that when a pattern recognition problem is selected at random from all possible problems, the optimum number of measurement states decreases with the amount of the training data.

In nearly all cases $T$ is better than both $U$ and $V$, probably because $T$ uses $P(D|C_i)$ while $U$ and $V$ do not. Differences between, $T$, $U$, and $V$ are generally less pronounced when characters occur with English probabilities than when the characters are equiprobable.

Comparison of the results for $T$ versus $W$ and $U$ versus $Y$ shows that making all vectors equal in length by adding zeros makes recognition more difficult.

For algorithms $T$ and $U$ recognition accuracy improved as the number of scans and decision trees increased, as one would expect. Although tests to resolve ambiguities were performed only for those cases indicated by square brackets in Fig. 4, it appears that schemes which first classify characters into one of several classes and then perform further class-dependent tests may be significantly better than schemes which recognize all characters solely on the basis of one measurement procedure common to all characters.

Considerable improvement results from using the six-part rather than the four-part area division when recognition accuracy is reasonably high. Clemens [7] used the binary vector resulting from the four-part division, along with a coarse description of the character's height-to-width ratio to recognize 260 upper case characters from ten different type fonts. Algorithm $S$ was used. Even when the same data was used for both training and testing, 15 percent error resulted. Important extrema on many of the upper case characters occur in squares 100 and 101 in Fig. 3, and these extrema are more accurately located by a six-part than by a four-part coord word. In applying Clemens' technique to the recognition of upper case handprinted characters, Munson [5] obtained 42 percent error of which 19 percent was reject when 2340 characters from seven writers were used for training and an (apparently) comparable number of samples from different individuals was used for testing. Perhaps Munson's results would improve considerably if, in addition to using a six-part area, either a larger number of training samples were used or the experiment was repeated using algorithm $T$ rather than $S$.

In addition to yielding a feature vector of relatively low dimensionality, our recognition scheme is insensitive to character size, position, and line thickness, and relatively insensitive to variations in character style. Touching characters are troublesome, although characters which touch each other regularly can be treated as single characters. The real difficulty is with broken characters, such as $R$ in Fig. 6. One corrective measure is to close small breaks by blackening all white squares adjacent to a black square. A second alternative is to move a bar of length $b$ along the outside contour, keeping one end of the bar against the contour and maintaining an angle of $90^\circ$ between the bar and the line tangent to the contour at the bar-contour point of contact. Whenever the other end of the bar encounters black, the line defined by that portion of the bar between the two black points is made part of the character, as in Fig. 6. The bar must be long enough to close most breaks but short enough that intentional breaks remain.

The problem of broken characters can be partially

---

4 By using $x$ and $y$ thresholds equal to 3/16 letter height, the error was reduced to 3 percent.

5 These results are for $x$ and $y$ thresholds of 1/10 a character's width and height, respectively.
solved by simply requiring that people avoid printing them. This constraint apparently caused our printers no real difficulty, although other reports [17] indicate that people do not always make unbroken characters even when they are required to do so.

Final conclusions should wait for experiments on several large data sets. It appears that algorithms $T$, $U$, and $V$ operating on binary vectors obtained from contours of characters quantized as in Fig. 3 yield recognition accuracies much better than those obtainable using a partially trained optimum classifier, and that if six-part area division is used, reasonably good recognition accuracies can be achieved, particularly if simple tests are used to differentiate between commonly confused characters and if contextual constraints are incorporated. These conclusions support those of Bakis et al. [18] that curve-following features extract the significant information from handprinting.

REFERENCES


A Sonic Pen: A Digital Stylus System

A. E. BRENNER AND P. de BRUYNE

Abstract—An inexpensive computer graphical input device giving Cartesian coordinates with better than 1-mm resolution is described. The technique works equally well for two-or three-dimensional input variants of the device.

Index Terms—Digital stylus, graphical input, sonic pen, three-di-

mensional input, two-dimensional input.

INTRODUCTION

Computer graphic displays often require interaction with a human operator which cannot be conveniently handled by a keyboard. The display program may then be written to allow the use of a light pen as an operator control. Although the hardware involved is very simple, the pen searching and tracking technique which is required complicates the computer program and consumes appreciable computer time. When time-sharing computer terminals are used, the available computer time is often insufficient for a live cycling display, and use is made of a storage tube or local display memory. Use of a light pen under these conditions becomes impractical, and a RAND [1] tablet is a natural device to perform such input functions in that it is an active device generating a computer interrupt signal, either periodically or on demand. The time required to store data following each interrupt is very small; hence efficient use may be made of the computer time. The sonic pen system [2] to be described performs functionally as a RAND tablet or the Sylvania data tablet [3], although operationally it is a much simpler device. It has the added advantages of being relatively inexpensive and having the capability of digitizing in three dimensions. Finally, the scheme directly generalizes to a three-dimensional graphical input device. In this sense, although it is similar to the Lincoln Laboratory Lincoln wand [4] in that it uses the propagation time of a sound wave as a measure of distance, the two devices differ substantially.

SONIC PEN PRINCIPLE

The basic sonic pen digitizer shown in Fig. 1 consists of an orthogonal pair of plane microphones positioned at two edges of the digitized area and a pen or puck capable of generating a short sonic pulse. In use the sonic pen generates a spherical sonic wavefront by causing a small sparc to jump across a small gap in the tip of the pen. When the wavefront first reaches the sensing microphone planes, an output is obtained with a rise time of 1 to 2 μs. The distance from the pen to each microphone is measured by counting clock pulses during the transit time of the sound front from the spark gap to the microphone. The limiting accuracy is

Manuscript received August 21, 1969; revised December 14, 1969. This work was performed at Harvard University, Cambridge, Mass., and was supported in part by AEC Contract AT(30-1)-2752.

A. E. Brenner is with the Department of Physics, Harvard University, Cambridge, Mass.

P. de Bruyne is with R.C.A. Laboratories, Burlington, Mass.