Pattern Recognition Vol. 16, No. 5, pp. 447–458, 1983.
Printed in Great Britain.

0031-3203/83 $3.00+ .00
Pergamon Press Ltd.
© 1983 Pattern Recognition Society

PREPROCESSING TECHNIQUES FOR CURSIVE SCRIPT WORD RECOGNITION

M. K. Brown
Bell Telephone Laboratories, 278 Carlton Avenue,
Piscataway, NJ 08854

and

S. Ganapathy*
Ultrasonic Imaging Laboratory, Department of Electrical & Computer Engineering,
Room 2506A E. Engineering Building, University of Michigan,
Ann Arbor, MI 48109, U.S.A.

(*Received* 16 *February* 1982; *in revised form* 13 *October* 1982; *received for publication* 2 *November* 1982)

Abstract—This paper deals with techniques for improving the recognition rate of a cursive script word recognition system. Closed-loop preprocessing techniques have been designed and implemented to achieve this objective on a limited vocabulary but with no restrictions on handwriting style. This paper discusses the details of such a system and its performance on samples from several authors. Results obtained from this study are promising and suggest that closed-loop verification is a potentially more useful technique than previous open-loop processing approaches.

Cursive script Word recognition Feature set Preprocessing Closed-loop Author-independent
Processing time Recognition accuracy

1. INTRODUCTION

Considerable attention has been paid to the cursive script recognition (CSR) problem, starting around 1960 with the work of Frishkopf and Harmon[1] at Bell Telephone Laboratories. Frishkopf tackled the problem of recognizing entire words, i.e. the word was treated as a single complex symbol. Harmon, on the other hand, attacked the problem by segmenting the component characters and performing character by character recognition. In 1962, Earnest[2] reported work on a word recognition system capable of recognizing about 10,000 words with only six features. At the same time Eden[3] described a syntactic recognition technique termed "analysis by synthesis". It consisted of a method of describing the script by a set of stroke primitives, using syntax rules to determine the proper sequences for various words. Serious interest continued until about 1965,[2-4] at which time many of the CSR researchers turned to speech recognition. Many of these researchers were using cursive script as a simpler medium for studying the (phoneme) segmentation problem. An excellent review of the early work in this area was given by Lindgren[4] in 1965.

For the next 15 years interest in CSR was not intense. Sayre[5] reports some very interesting results with "static" recognition of cursive script, i.e. no information concerning the time order of strokes is given to the recognizer. Ehrich and Koehler[6] report some experiments in contextual studies with cursive script in 1975. More recent interest was shown by Farag[7] in 1979 and Hayes[8] in 1980. Further work in the area was done by Brown and Ganapathy[9,10] in 1980, certain aspects of which are presented here. Reviews and references in the field are given by Harmon[11] and Rosenfeld.[12-14]

Cursive script recognition has recently started to receive renewed attention from a number of researchers. Much of the current revival in CSR research that the authors are aware of is being promoted by proprietary concerns. Consequently, little of this activity shows up in the literature. Increasing interest in CSR, as well as speech recognition, is aimed at alleviating many of the problems associated with man–machine communication. Such capability is becoming increasingly attractive because of the development of relatively cheap processing power in the form of microprocessor systems.

Our goal was to develop a recognition system that was author-independent and placed no unreasonable constraints on the handwriting. Preprocessing eliminates much of the initial variability of the handwriting data which causes author-sensitivity problems during recognition. Indeed, a perfect preprocessing system

*To whom correspondence should be addressed at: Room 4E 618, Computer Systems Research Laboratory, Bell Telephone Laboratories, Holmdel, NJ 07733, U.S.A.

Table 1

*1	No. of central threshold crossings
*2	No. of upper crossings
*3	No. of lower crossings
*4	No. of Y maxima
*5	No. of Y minima
*6	No. of cusps below center threshold
*7	No. of cusps above center threshold
8	No. of T's
9	No. of I's
10	No. of X's
11	No. of closures
*12	No. of openings right
*13	No. of openings up
*14	No. of openings left
*15	No. of openings down

would make the characters so uniform that recognition would become trivial.

Section 2 of this paper will discuss an implementation of a CSR system. A fundamental overview is given to aid in understanding the requirements of the preprocessing system, although knowledge of the recognition algorithm is not required to understand the function of the preprocessor. Section 3 will introduce the fundamental processes of open-loop preprocessing and describe the techniques used to normalize cursive script. Section 4 will discuss the closed-loop implementation of the preprocessor. Experimental results and a comparison of performance with open-loop preprocessing are given in Section 5.

2. SYSTEM IMPLEMENTATION

Samples of handwritten script were taken using a Tektronix graphics tablet and Tektronix 4010 graphics terminal coupled to an Amdahl 470 system via telephone lines. The graphics tablet supplies the computer with X and Y coordinates (10 bits per coordinate) in a chronological sequence at 4800 Baud or approximately 96 coordinates per second. Experimentation has shown that a maximum handwriting rate of about four characters per second will not normally be exceeded. At this rate about 24 points per character are taken. This resolution is sufficient for the feature extraction algorithms used here.

The script data is stored in the Amdahl 470 in the form of **X** and **Y** integer vectors. The length (number of dimensions) of the vectors depends upon the amount of script stored. A typical word of four to six letter length may be represented by 200–300 coordinates. A user is prompted for script input and can take as much as about five seconds or as little time as desired to write a cursive script word. The only other restrictions on the user are that the writing be true (not printed characters) and that the script be roughly horizontal (within +/− 30 degrees) and right side up. The data thus collected is referred to as the "pattern vector".

The unnormalized pattern vector represents the input script completely, including all of the variability due to the writer's individuality. The main purpose of preprocessing is to remove as much of this individuality as possible, leaving normalized author-independent script. The resulting data, called the normalized pattern vector, is ready for feature extraction and classification.

In order to gain a better appreciation of the requirements for preprocessing, a description of the feature set is useful. Feature extraction is performed by computer algorithms on the Amdahl 470. Features are detected from the normalized pattern vector directly. Individual routines are used for each different type of feature. The complete feature set consists of 183 features, 15 of which are global (extracted from all parts of the word) and 168 are local measurements on the pattern vector (taken from isolated regions of the word). There are 15 feature types (the 15 global features) and local measurements of these feature types are made so that information about individual script characters within the word is extracted. The global value for each feature type is a combination of the values obtained for the corresponding local measurements. Thus, there is some redundancy in the feature space, but the localized measurements add new information to the space. This technique allows us to avoid the segmentation problem by providing character level information.

The 15 feature types are listed in Table 1. Details of the local features are not critical to the understanding of the preprocessing techniques and will be discussed only briefly. For more details on feature space the reader is referred to the doctoral dissertation by Brown.(10) It should be clear that the features being presented at this point are those used for recognition purposes. Another set of features, to be described later, will be used for the actual preprocessing and will be described in detail.

The first class of feature types listed are the threshold crossings (features 1, 2 and 3). As the names imply, a horizontal threshold is established above, below and through the center of the normalized script. The number of times that the script crosses a threshold becomes the value of that feature. The obvious intent is that upper loops in the script (such as for the letters b, d, f, h, k and l) be detected by the upper threshold and similarly for the lower loops (f, g, j, p, q, y and z) and lower threshold. For this method to work the height of the script, the script location and the orientation must be known. There are two options in the implementation of this method: (a) normalize the script height, location and orientation to pre-establish values or (b) compute the location and orientation of the threshold to fit the script. For several reasons the former approach was taken. The predominant reason for choosing this preprocessing method is that script orientation need only be performed once, whereas in the latter method a transformation would have to be computed for each feature type that is dependent upon these script features. Features listed in Table 1 that are dependent upon script scale (X or Y), location or orientation are indicated with an asterisk (no. 1, 2, 3, 4, 5, 6, 7, 12, 13, 14 and 15).

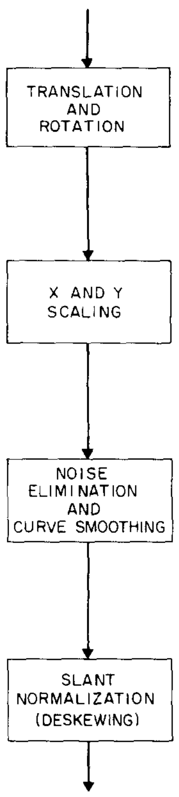
TRANSLATION AND ROTATION
X AND Y SCALING
NOISE ELIMINATION AND CURVE SMOOTHING
SLANT NORMALIZATION (DESKEWING)

Fig. 1. Open-loop preprocessing diagram.

The 168 local features are obtained by measuring the 15 feature types on windowed regions of the word. By a process referred to as "pseudo-segmentation" the script word is divided into overlapping segments upon which feature extraction is performed. These features provide the necessary character level information to yield high recognition accuracy. Further details of the recognition system and the features will be addressed in a separate paper.

3. PREPROCESSING TECHNIQUES FOR CURSIVE SCRIPT

3.1. *Introduction*

As is true with any type of data acquisition system that must interface with real world phenomena, noise errors and data variations will occur on input. The role of the preprocessor is to eliminate the noise as much as possible and to reduce data variation to a minimum. Typical operations performed by the cursive script preprocessor include coordinate translation, rotation and scaling, curve smoothing and character deskewing. Figure 1 shows how a general open-loop preprocessor might handle cursive script.

Open-loop preprocessing refers to a single pass processor which does not verify that the data modification has been performed as anticipated. Experience has shown that unexpected preprocessing errors often occur due to special circumstances in the input data. These errors are sent to the recognition system and recognition accuracy suffers as a result. As is found in control systems, it will be seen later that a closed-loop preprocessing technique provides more accuracy and reliability.

The techniques that are presented in this paper are newly developed. This is partly due to the particular requirements of the available hardware and partly due to the general lack of significant reports in the literature. In the authors' opinion, preprocessing for recognition purposes has been given much less attention than it deserves.

3.2. *Coordinate system*

The hardware used in this investigation has a fixed coordinate system in which the X and Y values may range from 0 to 1023 (10 bits per coordinate). The origin is at the lower left corner of the tablet and terminal screen. The units of X and Y will be referred to as picture elements or "pixels". Because of the aspect ratio of the terminal screen, only 767 pixels in the vertical direction may be displayed. Thus, the operating range of Y has been limited to 0–700 in this implementation. Physically, 100 pixels is equivalent to one inch of pen travel on the graphics tablet.

Cursive script words are normalized during preprocessing so that their baseline is at $Y = 300$ and the strokes begin at $X = 100$. The average height of the main body of the script is adjusted to 700 pixels. The average character spacing is set to 100 pixels. These figures were chosen primarily for readability of the computer graphics display and are not critical to recognition. However, once these display parameters are chosen, the recognition parameters become fixed as well.

The pattern vector, which contains the script image information, is stored as a $3 \times N$ array, where N is the number of sample points taken for the cursive script word. The 3 dimensions of the array are: (a) X coordinate, (b) Y coordinate and (c) visibility. The script image is reconstructed by drawing lines from the previous coordinate to the following coordinate in sequence. The visibility information tells whether the line drawn should be visible (draw) or invisible (move). In this way the script can be accurately reproduced, including lifting of the pen to dot an "i" or cross a "t". The first coordinate of the pattern vector is always a move.

Two-dimensional linear transformations can be performed on this pattern vector by the use of "homogeneous coordinate transformation matrices". This procedure involves augmenting the pattern array with a fourth dimension containing a scale factor, in this case equal to one. Thus, the pattern vector becomes a $4 \times N$ augmented array upon which linear transformations can be performed by post-multiplication of 3×3 matrices. The visibility information is carried along with the coordinates but does not enter into the calculation. The process is illuatrated in Fig. 2. Trans-

$$\left[\begin{array}{ccc|c} X_1' & Y_1' & 1 & I \\ X_2' & Y_2' & 1 & V \\ X_3' & Y_3' & 1 & V \\ & \cdots & & \cdots \\ X_n' & Y_n' & 1 & V \end{array}\right] = \left[\begin{array}{ccc|c} X_1 & Y_1 & 1 & I \\ X_2 & Y_2 & 1 & V \\ X_3 & Y_3 & 1 & V \\ & \cdots & & \cdots \\ X_n & Y_n & 1 & V \end{array}\right] \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ a & b & 1 \end{bmatrix}$$

Fig. 2. Coordinate transformation.

formations can be accumulated from several 3 × 3 matrices into one composite 3 × 3 transformation matrix by multiplying the matrices together in the proper order, thus allowing the several pattern transformations to be processed as one. The reader is referred to any good text on computer graphics for a complete discussion of homogeneous coordinate transformations (e.g. Rogers and Adams[15]).

In order to perform linear processes such as translation, rotation, etc., some measure of the current state of the cursive script is necessary. This orientation, size and shape information is obtained by extracting various features from the pattern vector. This information is analyzed and the results used to calculate entires of the transformation matrices which are in turn used to generate the composite transformation matrix.

The features used for preprocessing analysis are not necessarily the same as those used later for recognition purposes. Special features to be described next are extracted to find the cursive script baseline, vertical extent of the characters, measure the approximate number of characters in a word and determine the predominant character slant. This part of preprocessing is fundamental to the system and is used in both open-loop and closed-loop preprocessing.

3.3. *Translation and rotation*

The first tranformation operations performed on the cursive script are translation and rotation of the

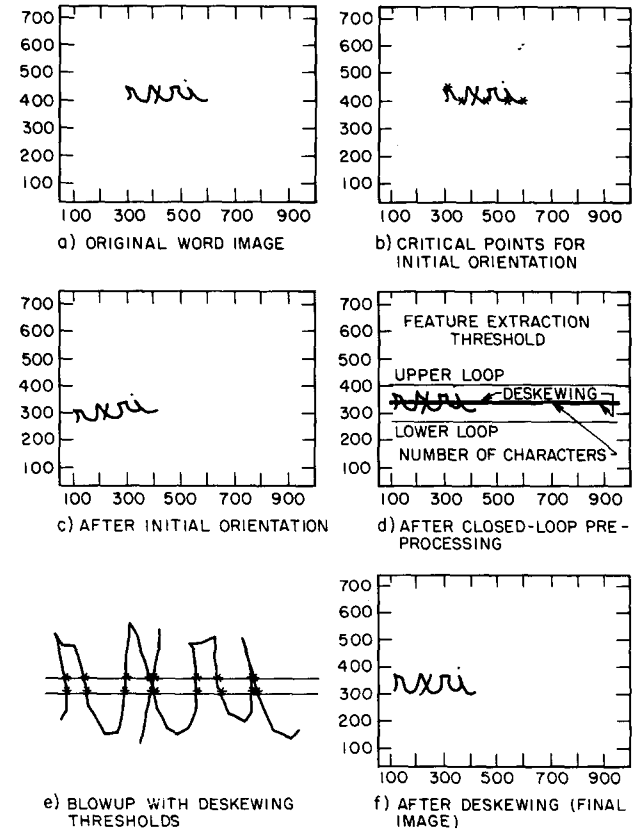

Fig. 3. Word images in preprocessing.

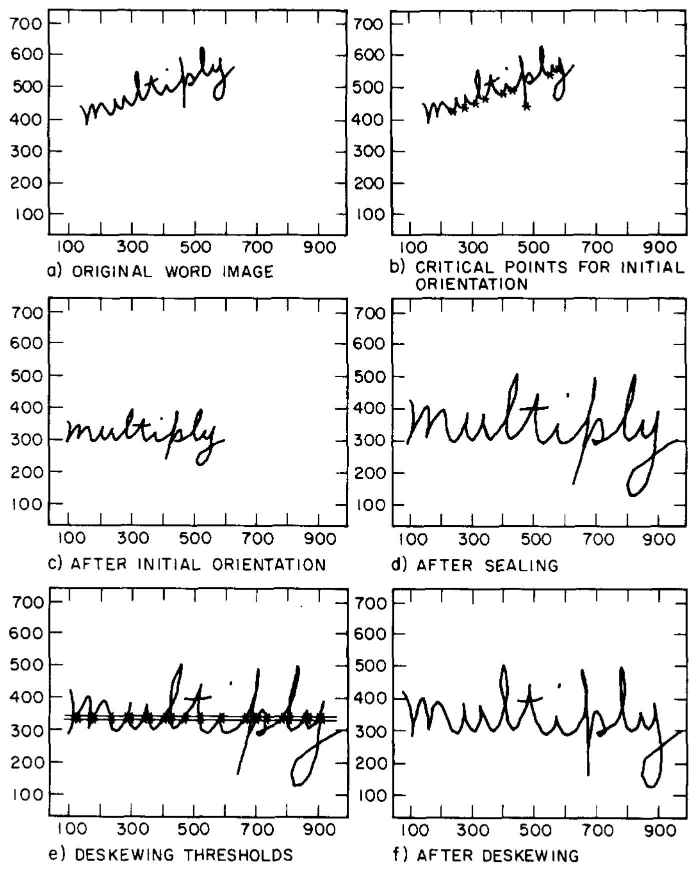

Fig. 4. Word images in preprocessing.

baseline to a fized location. The current location of the baseline is found by first detecting all local Y minima in the script. Figure 3 shows a sequence of word images which result from the various stages of preprocessing that are discussed in Sections 3.3 through 3.6. Figure 4 shows a corresponding set of images for the English word "multiply". Any minima that occur on retrograde strokes, dots and crossbars above the script or at cusp locations are eliminated, since these points are usually not on the baseline of the script. Using the remaining points, which we will call critical points, all possible lines interconnecting any two points are generated and the maximum likelihood slope and associated baseline are determined. Other techniques, such as average slope, were tested and found not to perform as well. Figure 3b illustrates critical points extracted from a four letter word.

Once the baseline slope is determined, the critical point which lies closest to this baseline slope is selected as the center of rotation and is used for Y translation. The word is translated to the origin, rotated until the baseline is horizontal and retranslated back into the display screen space so that the baseline lies at $Y = 300$. These operations can be accumulated into one transformation matrix. This completes the open-loop normalization of the baseline slope.

After rotation is performed, a translation to a normalized X coordinate is computed by determining the global minimum X coordinate for the script and translating the script so that minimum X lies at $X = 100$. This operation is performed after baseline slope adjustment, since extraction of the true beginning of the script becomes a trivial matter. Position and orientation normalization by the open-loop method is now complete.

3.4. *Scaling*

In order to properly scale the script, a measure of the height of the central body of the script must be found. Upper and lower loops and dots and crossbars on t's must be ignored. Since we know that, statistically, the majority of script Y coordinates will lie in the region of the central script body, the relative height of the script may be determined by looking at the probability density of the Y coordinates. This density function should possess low flat tails representing the coordinates of the upper and lower loops, with a rather abrupt plateau in the center representing the central body of script. This density function is determined in histogram form by placing horizontal thresholds over the script at predetermined intervals and counting the number of times the script crosses each threshold. This

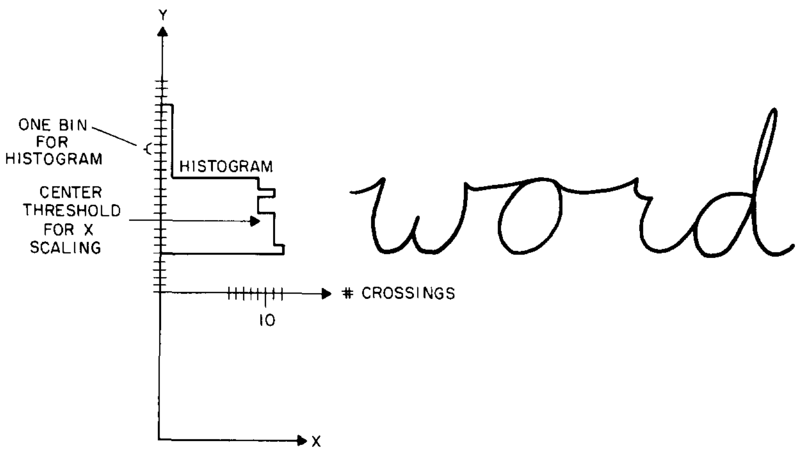

Fig. 5. Preprocess scaling procedure.

operation is depicted in Fig. 5. Note that the threshold and histogram estimate of the probability density thus obtained is not biased by the actual density of the sample points, which is dependent upon the amount of time the writer spent at each point in the script. The central body of the script is scaled in the Y dimension about the baseline so that the plateau of the histogram is 70 pixels wide.

Scaling in the X dimension is dependent upon the number of characters in the word. The number of characters in the word can be estimated by counting the number of penstrokes crossing a horizontal threshold in the center of the script. Using monogram character probabilities of English characters and analyzing the possible variants in the formation of the script characters, we have found that about 2.65 threshold crossings per character will occur in a typical English word. Recall also that in performing the Y scaling, the threshold crossings needed were used to determine the histogram. Hence, the amplitude of the histogram becomes a measure of the number of characters in the word. In this way we have a complete characterization of the X and Y scale of the script in the previously determined histogram. The X dimension scaling is performed about $X = 100$ so that a character is about 100 pixels wide.

3.5. *Curve smoothing*

Curve smoothing eliminates errors due to erratic hand motion during writing of the script and reduces the amount of storage required for the pattern. The basic process being performed is that of a moving linear interpolater with special attention paid to abrupt changes in script slopes. Curve smoothing has been tested at various points in the preprocessing flow, but was ultimately placed after the scaling operation because some of its functions are dependent upon script scale and orientation. The orientation and scaling algorithms were designed to be sufficiently noise insensitive for this reason. Curve smoothing generally reduces the number of sample points needed to represent the script, thus improving efficiency in both the remaining preprocessing steps and in the feature extraction phase during recognition.

The first operation performed by the smoothing algorithm is to identify and remove a leading upstroke if it starts below the lower loop detection threshold, as illustrated in Fig. 6. This is done because such strokes are not typical of a lower loop but generally arise out of stylistic variation.

The second function performed is the elimination of redundant points in the pattern vector. This operation is necessary for the proper functioning of the cusp extraction algorithm to be described later. This is a trivially implemented operation that looks sequentially through the points in the pattern vector and reduces each successive string of identical points to a single point.

The final stage of curve smoothing again scans the

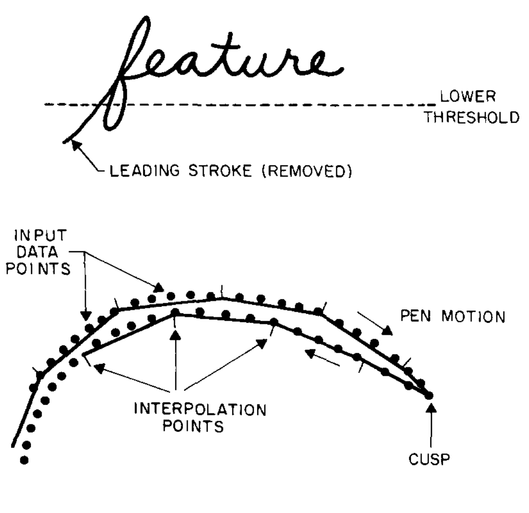

Fig. 6. Curve smoothing.

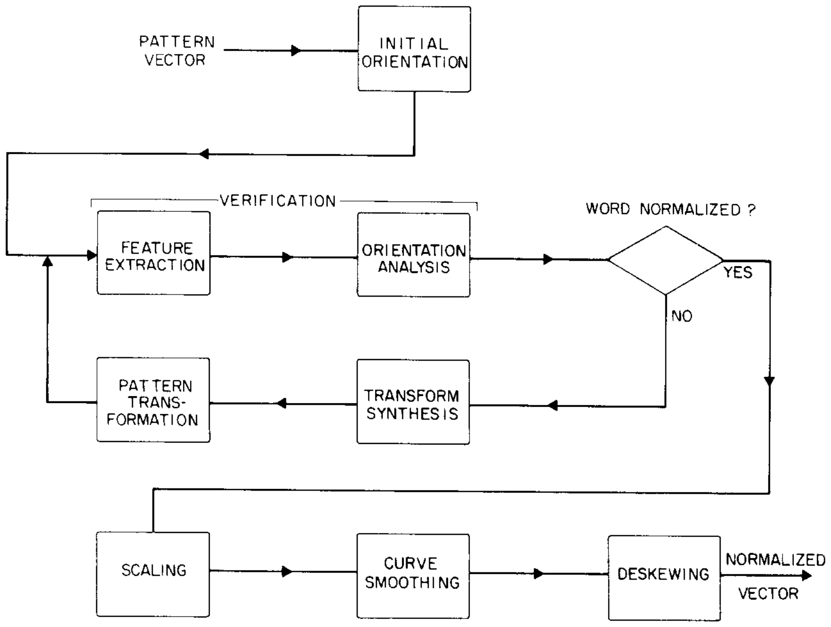

Fig. 7. Closed-loop preprocessing diagram.

pattern vector, determining the distance from each point to the next point in the script. As described earlier, the stroke interconnecting these points may be "visible" or "invisible." If the interconnecting stroke is invisible and its length is less than 10 pixels (which is relatively short), then it is probably due to erratic pen motion which temporarily caused the pen to be lifted. Such strokes are made visible. If the original stroke was visible and if the succeeding point is less than 10 pixels away from the current point, the succeeding point is replaced by a point that is approximately 10 pixels away. The procedure is to move progressively along the script path from a given starting point until a second point is found at a distance of about 10 pixels. Linear interpolation is used to determine the path between actual sample points. The second point thus determined becomes the new starting point and the procedure is repeated. In this way, interpoint stroke lengths become more uniform and a large number of unnecessary points are removed.

The interpolation procedure must be interrupted at stroke end points and at cusps so that these features are not smoothed out. Stroke endpoints are found when an invisible stroke is encountered which is greater in length than 10 pixels. The last visible sample point is retained so that the original end of the stroke is maintained. Dots on the characters i and j which were not previously reduced to single points during redundant point elimination are often reduced to a pair of points (one short visible line) in this manner.

Cusps are detected by looking for abrupt changes in stroke direction. Such abrupt changes are readily identified by determining the dot product of interconnecting stroke segments at each sample point, as illustrated in Fig. 6. Ordinarily, when a cusp has not been detected the direction vectors of the interconnecting stroke segments will point in nearly the same direction. Hence, the dot product will typically be positive until an abrupt change in direction (more than 90 degrees) occurs. Since normalized direction vectors are used, the dot product can be used directly as an indicator of the abrupt angular change. If the dot product becomes less than about 0.6 then a cusp is detected and the sample point is not modified, thus retaining the shape of the cusp. The path scanning and interpolation resumes starting with the succeeding sample point.

3.6. *Deskewing*

The last operation performed is the deskewing process. This operation removes the slant variation typically found in cursive script. A measurement of the script slant is obtained by placing two horizontal thresholds through the center of the script, as shown in Fig. 3e. Crossover points where the script crosses the thresholds are determined and an associated slope (or in order to avoid infinite slopes, the reciprocal slope) is determined from the crossover coordinates. The average of these slopes is used as the measure of script slant.

The script is deskewed, based on this measured slant, by translating sample points in the X dimension only. Sample points at the baseline do not move. Points away from the baseline move horizontally in proportion to their distance from the baseline and the measured amount of original script slant. This is a linear transformation which can be described by a 3×3 matrix. This transformation normalizes the script slant so that the characters appear to be

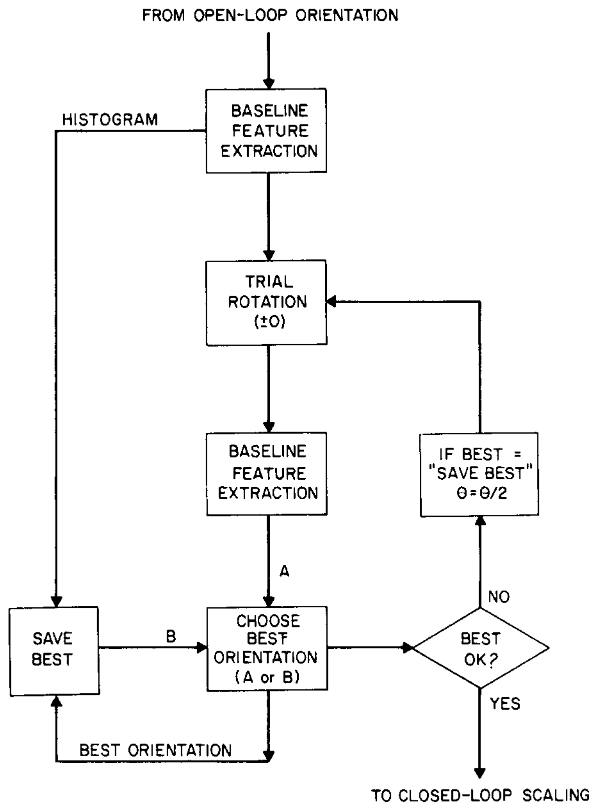
FROM OPEN-LOOP ORIENTATION
HISTOGRAM
BASELINE FEATURE EXTRACTION
TRIAL ROTATION (±Θ)
BASELINE FEATURE EXTRACTION
IF BEST = "SAVE BEST" Θ=Θ/2
A
NO
SAVE BEST
B
CHOOSE BEST ORIENTATION (A or B)
BEST OK?
YES
BEST ORIENTATION
TO CLOSED-LOOP SCALING

Fig. 8. Closed-loop orientation.

vertically oriented.

At this point the open-loop preprocessing is complete. The operations described have been tested on a variety of words of varying lengths, sizes, orientations and slants. In the majority of cases (typically more than 80%) the operations described are capable of properly orienting and scaling the script. The deskewing process is quite robust when words are longer than two characters and seldom causes problems.

Problems do arise, however, in the orientation and scaling of the script. Orientation problems sometimes occur when the critical baseline points selected do not all lie on the true baseline of the word. This is particularly problematic when the number of critical points is low. Such an orientation failure is shown in Fig. 3c. A solution to this problem will be addressed in the next section which describes techniques for performing closed-loop operations.

4. CLOSED-LOOP PREPROCESSING

4.1. *Introduction*

In the same way that a closed-loop control system reduces internal process error by the application of negative feedback, a closed-loop preprocessing system can reduce error by applying verification and information feedback. Since it is usually easy to verify the location and orientation of the word this procedure simplifies the measurement algorithms which are then likely to yield more consistent results. Verification, in this case, involves a position and orientation measurement which indicates, approximately, the amount of positional or rotational error. Since the measurement is approximate, a transformation based upon this measurement may still have some error present. The measurement technique, however, must be such that the amount of measurement error diminishes as the word becomes properly normalized. Although unlikely, it is possible that the original data will not require some of the normalization operations. In this event, since verification is performed before the first application of a transformation (in order to obtain the first transformation estimate), some of the transformation operations can be avoided, thus saving processing time.

The sequence of operations in the closed-loop system is similar to that of the previously described open-loop system. A block diagram of the closed-loop preprocessor is shown in Fig. 7. The diagram shows the verification and transformation processes in expanded form. The scaling and deskewing operations are also of a closed-loop nature but are shown as single blocks for conciseness. In order to minimize the number of operations in closed-loop preprocessing a partial open-loop process is executed before entering the closed-loop preprocessor. This open-loop phase nor-

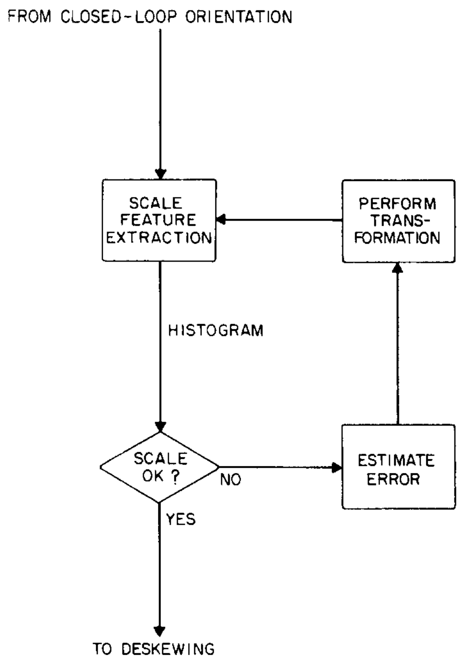

Fig. 9. Closed-loop scaling.

malizes the word position and rotation in a manner identical to open-loop preprocessing. Then, the closed-loop orientation system is entered, followed by a closed-loop scaling and closed-loop deskewing operations. The open-loop curve smoothing operation is performed between scaling and deskewing, as shown in Fig. 7.

4.2. *Closed-loop orientation*

A flow diagram of the closed-loop orientation procedure is shown in Fig. 8. This operation makes use of characteristics of the Y coordinate probability density function which was described earlier in Section 3.4. The objective is to minimize the width of the center plateau of the probability density function. This minimum condition should coincide with an increase in the slopes of the sides of the density function (see Fig. 8b) which indicates that the central body of the script is oriented perpendicularly with respect to the Y axis (i.e. the script is oriented horizontally). This is accomplished by computing the histogram using a set of thresholds as was done previously and then performing a trial rotation and recalculating the histogram. The two histograms thus obtained are compared to determine which has the narrowest plateau. If an improvement has occurred after rotation then the word is rotated again by the same angular change. The process is repeated until no improvement is obtained. The angular change is reduced to one half of its value and the trial rotations are tested again. In this way, by successively reducing the amount of the trial angular change, the word can be properly oriented to within 0.05 radians.

The characteristics of the histogram are tested by identifying the top of the center plateau, removing this part of the function, and integrating the remaining tails of the histogram. Thus, if the sides of the histogram are not steep a relatively large integral value will be obtained. This procedure was developed after finding that simpler techniques, such as placing a dynamic threshold across the histogram, yielded poor results in many cases. The integration procedure was found to be considerably more robust with few failures. At this stage failures that did occur were the result of gross errors in the preliminary orientation. The closed-loop procedure was unable to recover from such catastrophic failures.

Closed-loop scaling

Closed-loop scaling is performed in a manner similar to open-loop scaling except that the histogram is recalculated after each transformation to determine the results of the transformation (see Section 3.3). A flow chart of the process is shown in Fig. 9. In both the closed-loop scaling and deskewing procedures the verification step is performed before the first transformation. Thus, the transformation is performed only if it is necessary. Both the closed-loop scaling and closed-loop deskewing processes take advantage of the fact that measurement errors are minimum when the word is properly normalized. It has been found by experimentation that three loops through the scaling flow is always adequate for properly scaled words (two passes is typical). If more than three passes are needed then it is generally due to an improper word orientation and scaling cannot be done. Under these circumstances the entire process must be aborted and the word rejected.

4.4. *Closed-loop deskewing*

Like the closed-loop scaling operation, closed-loop deskewing is based on the open-loop skewing technique (see Section 3.6). The process flow is identical to that for scaling (Fig. 9) with the word "scale" replaced by "slant." This flow is very similar to that for closed-loop scaling. The procedure is to verify the script slant by the use of two centrally located horizontal thresholds and determine if a transformation is needed. If the transformation is performed then the verification is performed again. As the word becomes less skewed the script slant measurement approaches zero and the measurement error becomes low. The number of deskewing passes is limited to three. Again, failures to deskew in three passes are generally due to orientation errors and cannot be recovered.

5. RESULTS

Preprocessing performance can probably best be measured by its effect on recognition accuracy. Two cursive script testing sets were studied. One set contained words selected by a separate computer study of the recognition feature space. The words of this set were found to be very difficult to distinguish from one another by using the recognition algorithms. This

Table 2. Testing set of computer selected words with recognition error rates

Word	Recognition error Open-loop	Closed-loop
1. rxxi	10%	0%
2. xxxi	9.1%	9.1%
3. rxxr	33.3%	0%
4. exxi	20%	10%
5. xrxi	30%	40%
6. xxxr	0%	0%
7. rxxe	22.2%	30%
8. rxxx	0%	0%
9. cxxi	20%	20%
10. ixxi	20%	0%
11. rxri	22.2%	0%
12. exxr	40%	50%
13. xxri	40%	10%
14. rixx	20%	20%
15. xrxx	50%	30%
16. rexx	60%	10%
17. rxxs	0%	0%
18. rxre	0%	0%
19. exxe	20%	20%
20. exri	30%	10%
21. xxxe	20%	0%

Table 3. Testing set of English words with recognition error rates

Word	Recognition error Open-loop	Closed-loop
1. move	22.2%	10%
2. scale	0%	20%
3. rotate	30%	10%
4. reflect	10%	10%
5. erase	10%	20%
6. draw	40%	20%
7. read	40%	20%
8. write	0%	10%
9. add	10%	30%
10. subtract	30%	0%
11. multiply	0%	0%
12. divide	10%	0%
13. paragraph	10%	10%
14. indent	0%	0%
15. title	0%	0%
16. page	0%	0%
17. quote	0%	10%
18. number	10%	0%
19. footnote	0%	18.2%
20. word	20%	10%
21. image	0%	0%
22. feature	10%	10%

testing set is shown in Table 2 along with corresponding error rates. Ten handwritten samples each of the 21 four letter words (or a total of 210 handwritten words) were used for training and testing.

The philosophy behind the choice of these words arises from a concept of class density in the feature space. The more densely packed the classes are in the feature space then the more difficult it will be to distinguish between them. The computer selection algorithm chose a subset of 21 words from the set of all four letter words which yielded the greatest density measurement using our recognition metric.

The second testing set consisted of the 22 randomly selected English words shown in Table 3. These words were selected for a vocabulary that might be used in an interactive computer graphics system with graphics tablet. The words range in length from 3 to 9 characters and contain most of the letters of the English alphabet.

The recognition system was trained with each of the word sets, first using the open-loop preprocessing method. The total recognition error rates are shown in Table 4. The recognition system was retrained and retested using the same data with the closed-loop preprocessing method. As the results of Table 4 show, a significant improvement in performance was obtained. Note that vocabulary (a), which was chosen for difficult discrimination, yields a much higher error rate than the random vocabulary (b). For vocabulary (a) the error dropped nearly half. An improvement of about 17% was obtained for vocabulary (b). These improvements indicate that a significant number of words that were poorly preprocessed by the open-loop method were properly handled by the closed-loop method.

A closer look at Table 2 shows that of the 21 computer selected words, 10 showed improvement with closed-loop preprocessing, 8 were unchanged and 3 declined in recognition accuracy. For Table 3 the results show that 8 improved, 8 were unchanged and 6 declined. Although the majority of words either improved or remained unchanged, the question of why any words declined in performance is of particular interest.

Visual inspection of the words that failed to improve showed that the majority suffered from incorrect Y scaling. In these cases the scaling algorithm locked on to the wrong part of the scaling histogram and the resulting transformation was unrecoverable. This failure mode could probably be reduced by the use of a

Table 4

Test set	Recognition error Open-loop	Closed-loop	Change
(a) Computer selected	22.1%	12.4%	−43.9%
(b) English words	11.3%	9.4%	−16.8%

finer histogram and averaging along the histogram to obtain a smoother function.

Processing times for the various algorithms were obtained on the Amdahl 470 via operating system software which could report the actual execution times. Preprocessing runs involving several hundred words each were executed in processor times ranging from 84 mS to 187 mS with an average value at about 160 mS per word. The time required to preprocess a word depends upon the length of the word. For example the 84 mS per word execution times were obtained on a vocabulary consisting only of four letter words. Execution times of 160–187 mS were obtained for vocabularies containing 3–9 character long words with an average length of about 5.6 characters per word.

Thus, words can be preprocessed at a rate exceeding 300 words per minute, which is typical of human reading rates. Much of the preprocessing could also be speeded up if special purpose hardware were designed. Furthermore, these operations can be pipelined with the recognition phase so that throughput is maximized.

6. CONCLUSIONS

A new preprocessing technique for cursive script has been presented. Results indicate that a significant improvement in recognition accuracy is obtained using the closed-loop preprocessing method which can handle preprocessing problems that were beyond the capabilities of the open-loop method. Verification of script location, height, width and orientation is easier to implement and more reliable than open-loop or single pass methods which require measurement of these parameters with greater accuracy. These methods, in modified form, can be applied to hand printed text as well as cursive script and can offer significant improvement in performance while relaxing some of the measurement requirements normally associated with handwritten text preprocessing.

SUMMARY

Interest in cursive script processing and recognition has been increasing recently with the advent of relatively inexpensive processing hardware. In our attempts to develop a recognition system that was author-independent and placed no unreasonable constraints on handwriting style we found a need to improve current preprocessing techniques. This paper discusses a new approach to cursive script preprocessing referred to as closed-loop preprocessing.

Handwritten script is digitized by a Tektronics graphics tablet and is stored as a pattern vector. It is first translated and rotated to bring the baseline to a fixed location. Then the text is scaled to fit in a normalized window to enable size-independent recognition. Curve smoothing is done to eliminate errors due to erratic hand motion and to reduce the amount of storage required for the pattern. Finally a deskewing operation is performed to remove the slant variation typically found in cursive script.

These preprocessing techniques normalize script scale, orientation, position and slant. Closed-loop preprocessing uses techniques similar to those of open-loop preprocessing with the addition of information feedback from the results of previous transformations. In order to minimize the number of operations in closed-loop preprocessing a partial open-loop process is executed before entering the closed-loop preprocessor. This paper discusses the implementational aspects of these techniques.

In order to study the performance of these techniques two cursive script testing sets were used. One set consists of 21 four letter words (not English words) chosen because they are difficult to distinguish between by the recognition algorithms in use after preprocessing. The second set consists of 22 randomly selected English words chosen from a vocabulary that might be used in an interactive word processing system. Ten handwritten samples of each word were used in testing, yielding a total of 430 words. This new technique of closed-loop preprocessing was tested and compared against an open-loop version of the preprocessing method to determine the relative performance. The results indicate that closed-loop preprocessing yields a significant improvement in accuracy and consistency of the overall word recognition system. An analysis of the processing time shows that words can be preprocessed at a rate exceeding 300 words per minute, which is typical of human reading speeds. Further, these techniques in modified form can be applied to handprinted text as well and with special purpose hardware these operations can be pipelined with the recognition phase to maximize throughput.

REFERENCES

1. L. S. Frishkopf and L. D. Harmon, Machine reading of cursive script, 4th London Symposium on Information Theory, 1961.
2. L. D. Earnest, Machine recognition of cursive writing. *Information Processing* (1962).
3. M. Eden, Handwriting and pattern recognition, *IRE Trans. Inf. Theory* **IT-8,** 168 (1962).
4. N. Lindgren, Machine recognition of human language—III, *IEEE Spectrum,* 104–116 (May 1965).
5. K. M. Sayre, Machine recognition of handwritten words, *Pattern Recognition* **5,** 213–228 (1973).
6. R. W. Ehrich and K. J. Koehler, Experiments in the contextual recognition of cursive script, *IEEE Trans. Comput.* **C-24,** 182–194 (1975).
7. R. F. H. Farag, Word-level recognition of cursive script, *IEEE Trans. Comput.* **C-28,** 172–175 (1979).
8. K. Hayes, Reading handwritten words using hierarchical relaxation, Ph.D. Thesis, University of Maryland.
9. M. K. Brown and S. Ganapathy, Cursive script recognition, *Proceedings of the 1980 International Conference on Cybernetics and Society*, Boston, MA, pp. 45–51 (1980).
10. M. K. Brown, Cursive script word recognition, SEL Report No. 151 (Ph.D. Thesis) (1981).
11. L. D. Harmon, Automatic recognition of print and script, *IEEE Proc.* **60,** 1165–1176 (1972).

12. A. Rosenfeld, Picture processing: 1974, *Comput. Graphics Image Process.* **4,** 133–155.
13. A. Rosenfeld, Picture processing: 1972, *Comput. Graphics Image Process.* **1,** 394–468.
14. A. Rosenfeld, *Picture Processing by Computer*. Academic Press, New York (1969).
15. D. F. Rogers and J. A. Adams, *Mathematical Elements for Computer Graphics*. McGraw–Hill, New York (1976).

About the Author—Michael K. Brown was born in Highland Park, MI, on 4 January 1951. He received the B.S.E.E. degree in 1973 from the University of Michigan, Ann Arbor, and the M.S. and Ph.D. degrees in 1977 and 1981, respectively, from the University of Michigan, both in Electrical Engineering.

From 1973 to 1976 he was employed with the Burrough's Corp. and was involved in the development of ink jet printer systems. From 1976 to 1980 he continued his work with Burrough's Corp. as a consultant while pursuing his Ph.D. degree at the University of Michigan. His thesis was in the area of image processing and pattern recognition. Since 1980 he has been with the Speech Processing Group at Bell Laboratories, Murray Hill, NJ, where he has been involved in research in speech recognition and synthesis techniques.

About the Author—S. Ganapathy is currently an Assistant Professor of Electrical and Computer Engineering and is the Director of the Ultrasonic Imaging Laboratory at The University of Michigan. At present his research interests are in nondestructive evaluation, artificial intelligence and computer graphics. He has worked previously in the areas of software engineering and combinatorics.

Prof. Ganapathy obtained his B.Tech and M. Tech in Electrical Engineering from the Indian Institute of Technology and obtained a Ph.D. in the area of Artificial Intelligence from the Department of Computer Science at Stanford University in 1976.