R. L. Hoffman
J. W. McCullough

Segmentation Methods for Recognition of Machine-printed Characters

Abstract: This paper reports an investigation of some methods for isolating, or segmenting, characters during the reading of machine-printed text by optical character recognition systems. Two new segmentation algorithms using feature extraction techniques are presented; both are intended for use in the recognition of machine-printed lines of 10-, 11- and 12-pitch serif-type multifont characters. One of the methods, called quasi-topological segmentation, bases the decision to "section" a character on a combination of feature-extraction and character-width measurements. The other method, topological segmentation, involves feature extraction alone.

The algorithms have been tested with an evaluation method that is independent of any particular recognition system. Test results are based on application of the algorithm to upper-case alphanumeric characters gathered from print sources that represent the existing world of machine printing. The topological approach demonstrated better performance on the test data than did the quasi-topological approach.

Introduction

When character recognition systems are structured to recognize one character at a time, some means must be provided to divide the incoming data stream into segments that define the beginning and end of each character. Writing about this aspect of pattern recognition in his review article, G. Nagy [1] stated that "object isolation is all too often ignored in laboratory studies. Yet touching characters are responsible for the majority of errors in the automatic reading of both machine-printed and hand-printed text. ..."

The importance of the touching-character problem in the design of practical character recognition machines motivated the laboratory study reported in this paper. We present two new algorithms for separating upper-case serif characters, develop a general philosophy for evaluating the effectiveness of segmentation algorithms, and evaluate the performance of our algorithms when they are applied to 10-, 11- and 12-pitch alphanumeric characters.

The segmentation algorithms were developed specifically for potential use with recognition systems that use a raster-type scanner to produce an analog video signal that is digitized before presentation of the data to the recognition logic. The raster is assumed to move from right to left across a line of printed characters and to make approximately 20 vertical scans per character. This approach to recognition technology is the one most commonly used in IBM's current optical character recognition machines. A paper on the IBM 1975 Optical Page Reader [2] gives one example of how the approach has been implemented.

Other approaches to recognition technology may not require that decisions be made to identify the beginning and end of characters. Nevertheless, the performance of any recognition system is affected by the presence of touching characters and the design of recognition algorithms must take the problem into account (see Clayden, Clowes and Parks [3]).

Simple character recognition systems of the type we are concerned with perform segmentation by requiring that bit patterns of characters be separated by scans containing no "black" bits. However, this method is rarely adequate to separate characters printed in the common business-machine and typewriter fonts. These fonts, after all, were not designed with machine recognition in mind; but they are nevertheless the fonts it is most desirable for a machine to be able to recognize. In the 12-pitch, serif-type fonts examined for the present study, up to 35 percent of the segments occurred not at blank scans, but within touching character pairs.

Definitions

What we have so far referred to as "segmentation" is a process that is executed in three distinct, sequentially applied steps:

1) identifying start-of-character features,
2) identifying end-of-character features (sectioning), and
3) deciding which scan is to represent the dividing line between two characters (segmenting).

From now on, the term "segmentation" will be used only to refer to the entire process. The term "segmenting" is used in reference to the third part alone. It is the second step, which we call "sectioning," that we concentrate on in this paper. The first step is touched upon only lightly.

Many segmenting algorithms developed by IBM and others use either the analog video signal or the digitized video bit pattern as the basis for correlation measurements that are thought to be reliable indicators of the character end. These methods determine which scan should be the dividing line by comparing the correlation measurements of several adjacent scans. Scan correlation techniques are obvious outgrowths of searching for blank scans between characters and can be economically implemented in hardware. However, their performance may be unacceptable if they are "free running" across characters; i.e., if they are permitted to make a segment after any scan at which the decision criteria are satisfied. If the bit patterns of adjacent characters touch each other to the extent commonly found in serif-type fonts, then a free-running correlation scheme that is strong enough to make valid separations between "hard-touching" characters is sure to make invalid separations within characters.

Whenever one chooses to use a non-free-running segmenting scheme, he must also provide a sectioning function which, by our definition, generates a signal that permits the segmenting algorithm to begin. The sectioning function, in effect, makes the judgment that the end-of-character region has been reached and that it is now appropriate to test for the particular scan that will be identified as the end-of-character.

One method of sectioning is based on an a priori pitch (character width) measurement. Here the segmenting algorithm is permitted to operate whenever a specified number of scans has occurred since the start-of-character scan. However, pitch measurement can locate the end-of-character region only approximately since the ratio between character widths in a fixed-pitch font can be as much as 2 to 1.

Another class of sectioning techniques is based on pattern feature extraction. The features chosen might be very simple or they might be complex enough for use in character recognition itself. Two important constraints are encountered when the latter type of feature extraction is used for sectioning. For one, the features must be "tuned" to characters from specific fixed-pitch fonts if high recognition performance is to be achieved.

For another, this approach to sectioning makes it economically unfeasible to use any approach to recognition other than feature extraction. Both constraints are incompatible with our desire to provide a segmentation algorithm that is useful over a range of pitch sizes and is independent of machine logic.

In this study we investigated several feature extraction schemes. We chose a middle course and looked for medium-power feature extraction techniques more applicable to sectioning than to recognition. One approach, referred to as quasi-topological, makes use of character width measurements (in number of scans) as well as pattern geometry. The other approach, termed topological, depends only on the geometry of the character under consideration. We undertook the investigation of both approaches because it appeared that each would have its own area of special usefulness. We thought that the quasi-topological algorithm could be adjusted for better performance on a single type font (12-pitch serif, in particular), while the topological algorithm could provide good performance over a wide range of character styles and widths (possibly even including fonts with proportionally spaced characters).

All of the work reported here is based on a right-to-left direction of scanning. This basic design decision was made because strong end-of-character indicators, such as full-height vertical strokes and closures, occur more frequently on the left side than on the right side. This fact led us to expect that the performance of segmenting algorithms would be enhanced by right-to-left scanning. On the other hand, we realized that it is possible for the weaker right-side indicators to degrade performance of the start-of-character algorithms. Consideration of performance trade-offs between right-to-left and left-to-right scanning, however, are beyond the scope of this paper.

Segmentation based on quasi-topological sectioning

Three feature-extraction schemes were proposed as candidates for quasi-topological sectioning: 1) a vertical scan black-bit peak-contour histogram pattern, 2) a horizontal stroke pattern and 3) a character envelope contour pattern.

- *Black-bit histogram*

It was postulated that the histogram obtained by counting the number of black bits in each vertical scan through a character could be used for sectioning characters from fixed-pitch type fonts. A peak occurs in the black-bit histogram for each vertical stroke of the character (over-

hanging serifs, as in a T, are considered vertical strokes). Characters from serif-type fonts have one, two or three peaks, as demonstrated by I, 8 and T, respectively. For fixed-pitch fonts it was postulated that the spacing between peaks and the contour of the black-bit histogram could be used as sectioning criteria.

A computer program to detect peaks in the black-bit histogram was written and tested. Simple filtering of the digitized video pattern was effective in reducing the number of missing or false peaks caused by noisy characters, i.e., patterns with white spots in predominantly black areas and vice versa. However, many unresolvable conflicts were encountered in formulating logic to use peak-count and contour information for sectioning. For example, peaks missed because of overlapping characters such as LT and WA led to inappropriate sectioning action.

The primary problem was to determine whether the character being scanned had one, two or three peaks. Peak spacing and slope of the line connecting two peaks were found to be unreliable indicators of the true peak count even for fixed-pitch characters. Unless the maximum character width (in number of scans) is known, one cannot determine whether sectioning should be done after observing the first peak because a single-peak character has been encountered or whether sectioning should be held off until two (or three) peaks have been observed. The black bit histogram method was discarded as a candidate for use in a segmentation algorithm.

• *Horizontal stroke pattern sectioning*

A sectioning algorithm was developed based on recognition of patterns in the sequence of stroke-count values computed as the character is traversed horizontally. The stroke count is the number of times a given vertical scan encounters part of the character. Examples of stroke-count sequences for several characters are shown in Fig. 1. The characters I, S, T and A produce patterns in their stroke-count sequences that are symmetric about the center line of the character, while characters like L and 7 have unsymmetric patterns. The character T, for example, shows the pattern (12121), which is symmetric about the center vertical stroke. A vertical stroke, by definition generates only one horizontal stroke in the scans that it occupies.

Stroke counting, when applied to the entire alphabet of upper-case alphanumeric characters, clusters patterns into a few general classes. Serif characters like M, N, T, U, W and Y that are symmetric about the center line exhibit a (12121) pattern, while D, V and 7 exhibit a (121) pattern. In all, 11 classes or subclasses were identified during experiments on serif characters.

Sectioning decisions are made by finding the closest match between the exhibited stroke-count pattern and one of a set of stored, reference stroke-count patterns.

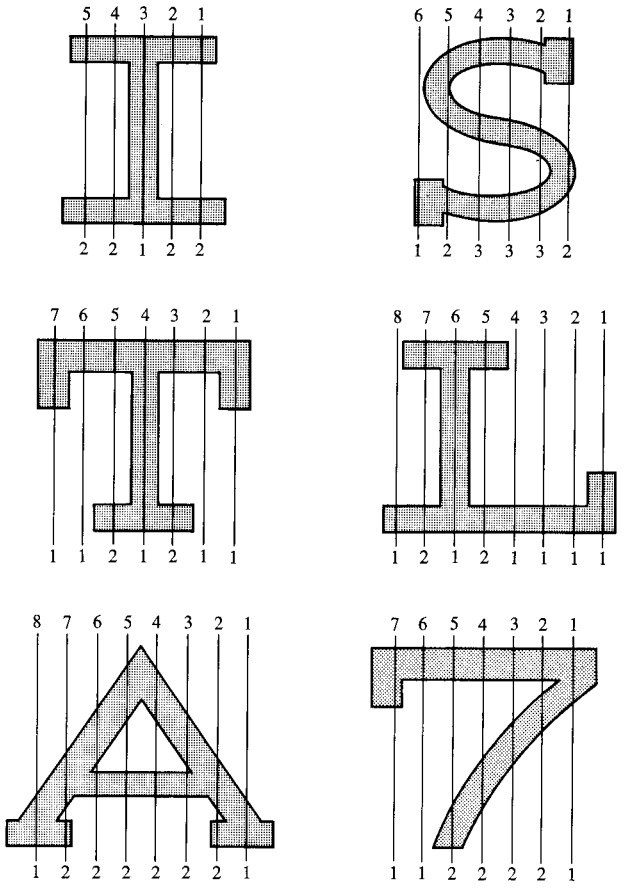


Figure 1 Horizontal stroke patterns. The numbers above the character identify the scan number and those below, the stroke count.

The references were determined a priori and each was "tuned" to data from the particular type font under consideration. These references yield decisions that are statistical rather than deterministic since there is no unique correspondence between stroke-count patterns and references. Matching of stroke-count patterns to references is attempted after each scan.

During each scan, certain measurements are extracted and stored in a table. The measurements used are the current stroke count and the number of scans for which that count has existed, the previous stroke count and the number of scans for which it existed, and the second previous stroke count and the number of scans for which it existed. For an example see Fig. 2.

Matching of data in the six-item table to the stored references is performed by the following procedure:

1) The group of references assigned to the stroke count of the current scan is retrieved from storage. For a current stroke count of 1, the group contains seven references and for a current stroke count of 2, the group contains four

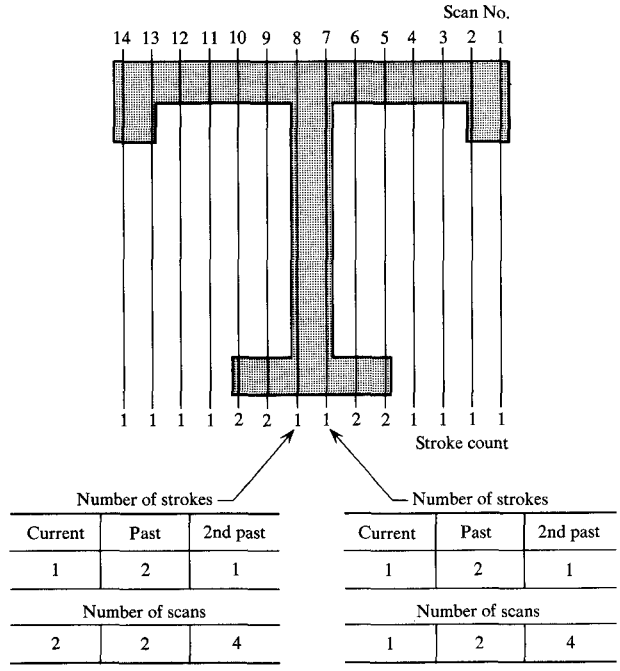
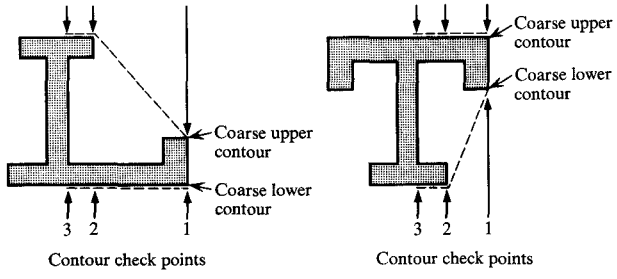


Figure 2 Feature extraction from horizontal stroke patterns.

Figure 3 Pattern contour measurements.



references. Accessing the references is not undertaken if the count is 3 since no "ideal" character exists that has such a count in its end-of-character region. Whenever the stroke count is 0, a sectioning decision is made independently of the references.

2) The references that have stroke counts matching the current, past and second-past values of the table are identified.

3) The number of scans for which each stroke count has existed is compared with these references to determine whether the number falls within the range prescribed for one of them. If a match occurs, a decision to section is made.

Early in the scanning of a character, patterns containing fewer than three sequences of non-zero stroke counts are likely to be encountered. Nevertheless, tests for sectioning are still made after some predetermined number of scans because some characters actually do have fewer than three stroke-count sequences. For instance, a dash (—) has but one sequence of one-stroke scans.

An important disadvantage of the above procedure is that it does not appear capable of producing reliable results when characters of variable width are presented to the scanner. What happens is that a reference which validly indicates the end-of-character region for some characters may also cause an invalid sectioning at the center region of other characters. A primary example of such a conflict is seen in the stroke-count patterns for L and T. The first three sequences of stroke counts are 121 for both characters. However, the second one-count of this pattern occurs in the long vertical stroke at the end of the L and in the long stroke at the *center* of the T.

If the sectioning algorithm were restricted for use with only a single character font, it would be easy to design a reference pattern to discriminate between the L and the T: the first one-count would always exist for many more scans through the L than through the T. But if one also wishes to distinguish a narrow L from a wide T, sectioning on this basis will not be successful.

In our experiments the L, T problem, as well as others of similar nature, could not be solved by "tuning" the references. To resolve these conflicts a technique involving measurements of character "envelope" outlines was combined with the stroke-count pattern method of sectioning. At each of the first three scans in which the stroke count changed value, the number of black bits was counted and the vertical positions of the top and bottom bits were noted. This made it possible to detect the envelope outlines of the right sides of characters and thus to distinguish many of those which had previously been unresolvable. Figure 3 shows the envelope contours of the L and the T. A sectioning algorithm based entirely on the contour characteristics was not considered, although there may be some merit in doing so.

• *The quasi-topological segmentation algorithm*

Figure 4 shows the flow diagram of a segmentation algorithm which incorporates a combination of the stroke-count and contour measurement methods of sectioning just described. For experimental purposes, the algorithm operates on input data stored on a magnetic tape that contains the digitized video signals obtained by scanning the lines of a printed document.

The procedure used to detect the start of a character is a simple scheme that tends to reduce the effects of "ragged" line edges. Two successive scans that contain at least one pair of horizontally adjacent black bits are sought out. When such scans are found, the number C of horizontally adjacent black bit pairs is counted. Also,

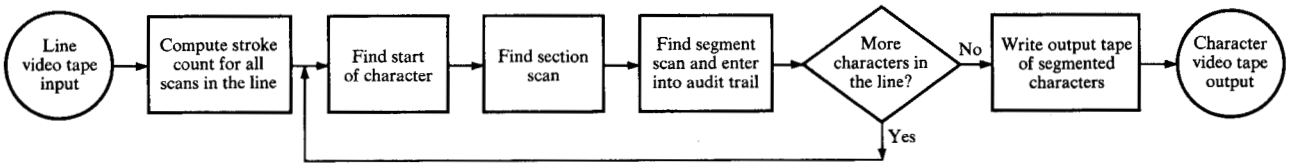


Figure 4 Flow chart for quasi-topological segmentation algorithm.

$\bar{C}$, the number of horizontally adjacent pairs in which one bit is black and the other is white is computed. If $C \geq \bar{C}$, the first of the two scans starts the character. If $C < \bar{C}$, and is also less than some specified number, the second scan starts the character. Under any other conditions, the first scan starts the character.

After the character-start decision, a further test is performed to determine whether or not the pattern encountered should really be considered to be a character. The pattern is said to be a character if a sufficient number of horizontally adjacent bit pairs continues to be both black for some prescribed number of successive scans. If this condition is not satisfied (because of too many "noise" bits or a ragged leading edge), the character-start procedure begins again.

Once the character-start decision has been made, the sectioning procedure begins. To obtain reasonable reliability and to cover a sufficiently large portion of a character while storing just three stroke-count sequences, it became necessary to do some filtering of the measurements made during sectioning. In the vertical (scan) direction the stroke count is increased only if two or more adjacent black bits are surrounded by white bits. A maximum count of three strokes is permitted per scan since, ideally, none of the characters in the type fonts studied has more than three horizontal strokes in any one scan. Filtering in the horizontal direction is used to smooth the variation in stroke count from scan to scan.

When the sectioning procedure has resulted in a decision that the end-of-character region has been reached, a sequence of logic tests is made to determine exactly where to segment the character. A given scan is said to represent the character end if any of the following conditions is satisfied:

1) The scan column contains fewer than some specified number of black bits horizontally adjacent to black bits in the previous column. This may indicate that a discontinuity between characters has been reached.
2) The scan column contains a specified number of white bits horizontally adjacent to black bits in the previous column. This may indicate that the end of a serif or some other character feature has been reached.
3) The next scan column contains a slightly larger number of black bits than the given column. This may indicate that a diagonal stroke at the right side of a new character has been reached.
4) The next scan column contains a significantly higher number of vertically adjacent black bits than the given column. This may indicate that some feature at the leading edge of a new character has been found.

Segmentation based on topological sectioning

The sectioning procedure used in the "topological" segmentation algorithm is based on the detection and measurement of the leading and trailing edges of character strokes as the bit pattern moves through a scan-storage shift register. At each scan three parameters are computed from logical tests of the bit pattern. Each parameter is assigned a weighting factor and all are combined in a "threshold" function that is sensitive to end-of-character (i.e., left-hand side) features when scanning from right to left.

The approach to sectioning discussed here is derived from the observation that in the bit pattern of any isolated character the sum of the vertical lengths (in bits) of all leading edges must be equal to the sum of the vertical lengths of all trailing edges. To use this observation for detecting the end-of-character region, one can examine the character bit pattern column-by-column. During examination of the right-most columns there is an accumulation of leading-edge bits and as the pattern is traversed toward the left, the number of trailing-edge bits begins to build up. It can be specified that when the accumulation of trailing-edge bits has become sufficiently near the total of leading-edge bits, the character should be sectioned. Here we define a leading-edge bit as a black bit with a horizontally adjacent white bit in the previous scan, and a trailing-edge bit as a white bit with a horizontally adjacent black bit in the previous scan.

Notice that in every column (scan) of a bit pattern that begins with a blank scan (i.e., $L_0 = 0$) the following relation holds:

$$\sum_{j=0}^{n} P_j - \sum_{j=0}^{n} N_j - L_n = 0, \tag{1}$$

where P_j is the number of leading-edge bits in scan j, N_j is the number of trailing-edge bits in scan j, and L_n is the number of black bits in scan n, the scan currently being examined. When a scan is reached in which $L_n = 0$,

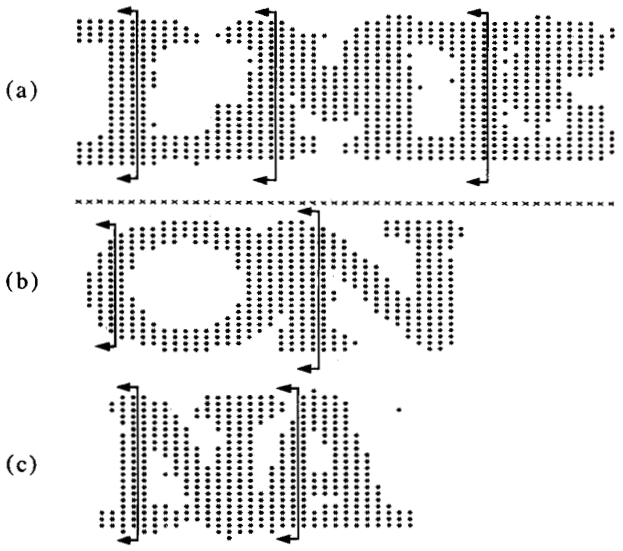


Figure 5 Examples of touching characters showing scans where sectioning is desirable.

all leading and trailing edges have been accounted for and a "natural" segmenting point has appeared.

The relationship of Eq. (1) can be adapted for use in the sectioning of touching, noisy characters. In the procedure used in the "topological" segmentation algorithm the parameters are combined in an equation of the form

$$T_n = \sum_{j=0}^{n} U_j P_j - \sum_{j=0}^{n} V_j N_j + Y_n P_n - W_n L_n - X_n. \tag{2}$$

Whenever a scan is reached in which $T_n \leq 0$, sectioning occurs.

The design problem is to choose suitable definitions for P, L and N and appropriate values for the coefficients U, V, W, X and Y so that sectioning will occur in the desired region of the character. The general considerations for choosing the terms and coefficients of Eq. (2) can be illustrated with some observations about character geometry in the regions where sectioning should occur. A few examples of touching bit patterns as well as the first scan where sectioning should be allowed are shown in Fig. 5. For characters that terminate in left-side vertical strokes, as does the E in line (a) of Fig. 5, the threshold function can be satisfied after the left vertical is detected if the coefficient V is increased in this area. Since the terminating feature of the character N in line (b) is largely obscured in the section area, a portion of the connecting bits must, in effect, be subtracted by the $W_n L_n$ and X_n terms as the section area is entered. The character A in line (c) illustrates the occurrence of decreasing bit density in the section area.

In general, the characteristics of the section area of characters can be summarized as follows:

- The parameter N has a relatively large value as left-side trailing edges are encountered. This suggests that its coefficient V should be increased near the desired section area.
- It is unlikely that the value of the parameter P will get much larger at the left-hand side of a character. Thus the term $Y_n P_n$ inhibits the satisfaction of the threshold condition if new leading-edge bits are detected in what would otherwise be the section area.
- The bit-density parameter L tends to decrease in value as the section area is encountered. However, the bit density in the first scan of the section area is likely to be almost as large as the maximum density of the character. These considerations lead to the following definition for the bit-density parameter: a bit in scan n is counted if it is black and is horizontally adjacent to the middle bit of three vertically adjacent black bits in scan $n - 1$. The bit density L_n is the number of such bits counted in scan n.

To reduce the effects of single-bit wide noise on the measurement results, the definitions of leading- and trailing-edge bits are restated: A bit in scan n is counted as a leading-edge bit if it is one of three horizontally adjacent bits in scans n, $n - 1$ and $n - 2$ that are respectively, black, black and white. The parameter P_n is the sum of leading-edge bits detected in scan n. Similarly, a bit in scan n is counted as a trailing-edge bit if it is one of three horizontally adjacent bits in scans n, $n - 1$ and $n - 2$ that are respectively, white, black and black. The parameter N_n is the sum of trailing-edge bits in scan n.

The coefficients U and Y of the P parameters in Eq. (2) were designed to be constant for all scans and their specific values resulted from experiments on the character sets that were candidates for sectioning. The other coefficients V, W and X have values that vary as the bit pattern is traversed. They each have the form

$$\begin{aligned} V_n &= \gamma_V + \delta_V F_n + \epsilon_V \bar{F}_n, \\ W_n &= \gamma_W + \epsilon_W \bar{F}_n, \\ X_n &= \delta_X F_n \\ U_n &= \gamma_U \\ Y_n &= \gamma_Y, \end{aligned} \tag{3}$$

where the γ's, δ's and ϵ's are positive constants determined experimentally for a given character set, and F_n and $\bar{F}_n$ are computed from a Connective Feedback Request $(\mathrm{CFR})_n$ function, which is determined at the end of each scan. By definition,

$$F_n = \begin{cases} 0 & \text{when } (\mathrm{CFR})_n < \lambda \\ (\mathrm{CFR})_n - \lambda & \text{when } (\mathrm{CFR})_n \geq \lambda, \end{cases} \tag{4a}$$

$$\bar{F}_n = \begin{cases} \bar{F}_{n-1} \text{ when } (\mathrm{CFR})_n \geq 1 \\ \quad \text{and} \quad (\mathrm{CFR})_n = (\mathrm{CFR})_{n-1} \\ \bar{F}_{n-1} \text{ when } (\mathrm{CFR})_n = (\mathrm{CFR})_{n-1} + 1 \end{cases} \tag{4b}$$

and

$$(\mathrm{CFR})_n = \begin{cases} (\mathrm{CFR})_{n-1} + 1 \text{ when } G_n - \eta F_n > \theta \\ (\mathrm{CFR})_{n-1} \text{ when } G_n - \eta F_n \leq \theta, \end{cases} \tag{5}$$

where

$$G_n = \sum_{i=0}^{n} (U_i P_i - V_i N_i)$$

and λ, η and θ are positive constants experimentally determined from the geometry of the characters to be sectioned. The starting values F_0, $\bar{F}_0$ and $(\mathrm{CFR})_0$ are all zero.

Equation (5) indicates that in the early scans the CFR count will increase by one as soon as G_n becomes greater than θ. It takes λ increases of CFR before F can have a nonzero value. Then F increases by one at every scan in which CFR increases. The $\bar{F}$ count increases by one at every scan in which CFR is not increased, provided that CFR has increased at least once for that character.

The value of G_n remains large as the character is traversed if the bit pattern is "heavy" or if it is composed largely of horizontal strokes, as in the character S (where many leading-edge bits and few trailing-edge bits are encountered). This condition indicates a large likelihood of connectivity between adjacent bit patterns and, consequently, a high probability of a difficult section. In this case the "heavy" feedback term F becomes large, causing the coefficient V, and to a lesser extent X, to increase with the number of scans and, hence, to force the threshold function T_n negative. On the other hand, low connectivity between adjacent bit patterns, such as occurs when characters are lightly printed or are composed of diagonal or vertical left-hand side strokes, causes $\bar{F}$ to increase. The dependence of the coefficients W and V on $\bar{F}$ produces proper sectioning in these characters by forcing T_n negative in the desired region.

The flow diagram in Fig. 6 illustrates the sequence of computations in the sectioning procedure as the bit pattern is examined scan-by-scan. First, the measurement coefficients are set to their initial values. Then the logical measurements are made, the P, N and L parameter values are computed and the CFR function is updated at each scan. Finally, the threshold function T_n is computed and tested to determine whether the requirements for sectioning have been met. If not, the next scan is examined. If so, the segmenting procedure is enabled.

The segmenting procedure makes use of two algorithms applied in parallel. One is called the "density increase" algorithm and the other is called the "terminating line-element" algorithm. The character is segmented if the decision criterion for either algorithm is satisfied.

In the first algorithm a bit-density function is computed for the section scan. Then another bit-density function is computed at each succeeding scan until a scan is found in which the difference between the two density functions exceeds a certain number ϕ. Taking the section scan as $j = 0$, the condition is expressed as

$$\sum_{j=1}^{n} P'_j - N'_n > \phi, \tag{6}$$

where n is the current scan.

A bit contributes to the P'_j count if it is black and horizontally adjacent to a white bit in scan $j - 1$, or if it is horizontally adjacent to bits in scans $j - 1$ and $j - 2$ that are, respectively, black and white. A bit in scan n contributes to the N'_n count if it is white and horizontally adjacent to a black bit in scan $n - 1$. It can be seen in Eq. (6) that when N'_n becomes sufficiently small, as will happen with touching characters when a transition occurs from a left-side trailing edge of the current character to a right-side leading edge of the next character, the current character will be segmented.

The second algorithm searches the scans, including and following the section scan, in a vertical direction for bit configurations indicative of terminating features. It does this by testing the bits of a 3×3 matrix. The first measurement M_1 looks for a 9-bit square containing all white bits. When this is found, a second measurement M_2 seeks a configuration of black and white bits as shown in Fig. 7. If M_2 is satisfied then M_3 checks to see if the M_2 configuration is followed by another 9-bit square of all white bits. When a scan is found in which all three measurements are satisfied in sequence, a decision to segment the character is made.

Figure 8 shows an example in which segmenting decisions were made by the two algorithms. The performance of the entire topological segmentation algorithm is discussed in the section on "Test results."

Segmentation testing philosophy and procedure

The performance of a segmentation algorithm on touching characters can be measured by the percent of "correctly" segmented characters. It may seem simple to measure segmentation error rates, but in theory and in practice it is not. Basically, no one universal definition of "correct" segmentation is available, as compared to the reject-substitution figure of merit used for evaluating recognition algorithms. One point of view is that segmentation is an integral factor in recognition and that recognition figure of merit [2] is a useful criterion for segmentation. We did not choose this approach because the results are constrained and influenced by the particular recognition

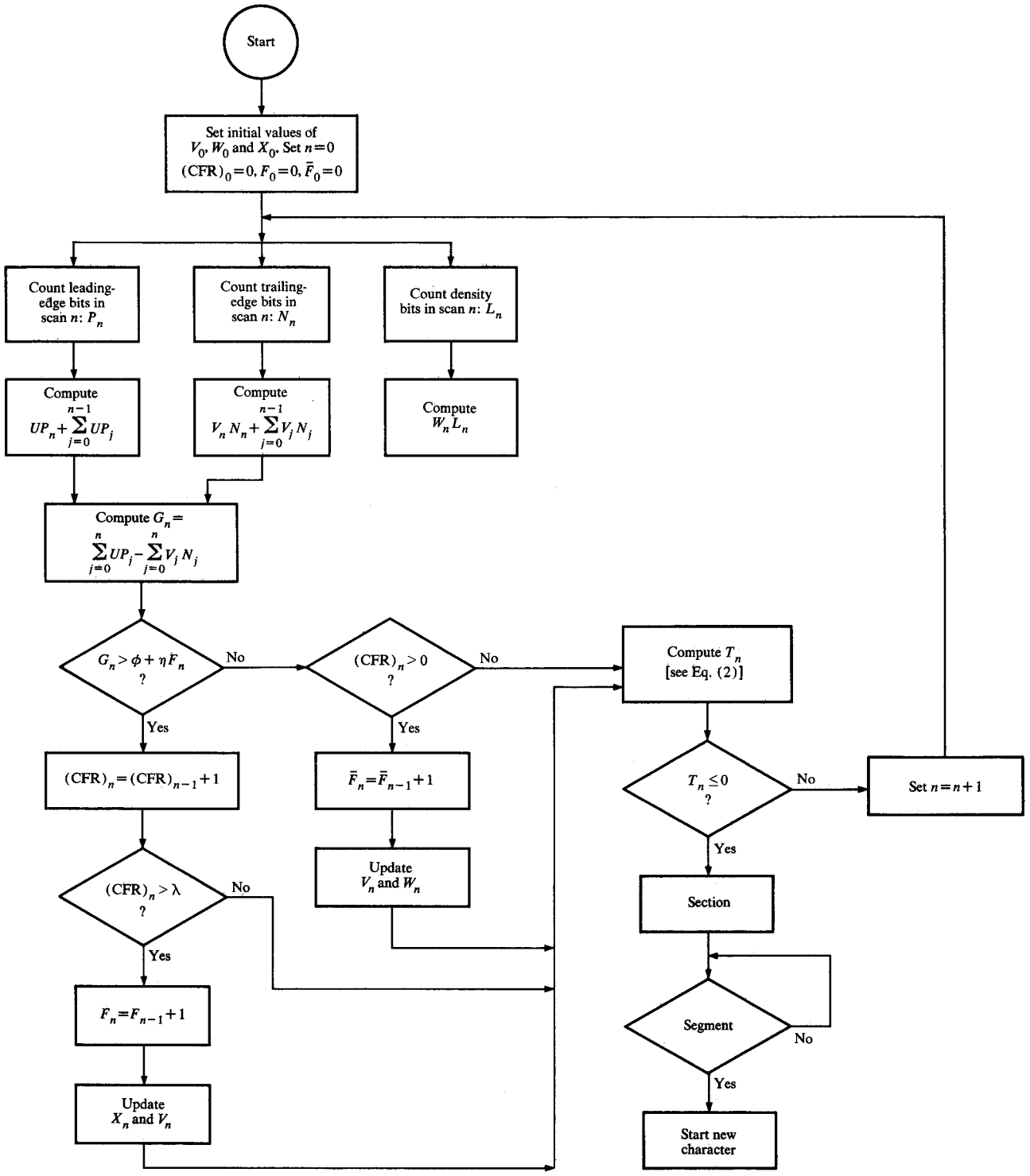


Figure 6 Flow chart of sectioning procedure used in the topological segmentation algorithm.

system employed. To explain further, consider that segmentation "errors" in general appear to the recognition system as characters shifted horizontally by some number of scans from a yet to be defined "ideal" registration. Recognition systems vary in their ability to correctly classify misregistered (shifted) characters. Template matching recognition schemes work poorly if at all, on misregistered patterns, while adaptive techniques can "learn" to correctly classify shifted patterns. The point is that the figure of merit is useful for evaluating how

well a segmentation algorithm and a recognition system work together, but figure of merit tells little about what may happen when that same segmentation algorithm is used with another recognition system.

The basic premise of the segmentation analysis procedures developed for this study is that for every touching character pair there is an "ideal" segment scan—ideal in the sense that the information lost by both characters is minimized if segmenting occurs at that scan. To find the ideal segment scans, a powerful pattern recognition system was used—the human observer. The ideal scan was selected as the point of minimum video density between characters, the scan where new features were encountered, or the scan that minimized the total feature loss.

Rules were established for counting segmentation errors. After observing many touching serif characters at a resolution of approximately 17 to 20 scans per character, we concluded that:

Segments at 0 and ± 1 scan from the ideal are correct, segments at ± 2 and ± 3 scans from the ideal may be correct, and segments at more than ± 3 scans from the ideal are errors.

At this resolution a recognition system of reasonable sophistication should tolerate ± 1 scan errors since rarely is a new feature added or a feature lost by deviating from the ideal by one scan. Deviations of more than 1 scan may, however, produce errors. Deviations of ± 2 or ± 3 scans are more likely to produce recognition errors. Deviations of four or more scans usually lead to recognition errors irrespective of the recognition system since important features of the character are lost. In the statistics that follow, a segment is considered correct if it deviates 0 or ± 1 scan from a manually chosen ideal segment scan, and probably is an error otherwise.

This method of analysis, named scan deviation segmentation analysis, can be automated for any one data set once ideal segment scans have been encoded. Editing these scans onto line video data by human observation is a time-consuming task. However, when ideal segment scans are available, the error rate for any segmentation algorithm may be computed without human intervention. An analysis program that simultaneously compared the scan deviations of each of two algorithms with respect to the ideal segments was developed. Pertinent summary statistics such as number of isolated characters (nontouching on both left and right) and number of touching characters were also generated.

The program also developed a matrix showing the number of segments per block of touching characters for each algorithm. When the number of segments made by a test algorithm differed from the ideal number, the block of characters was judged unsuitable for reliable automated analysis and scan difference statistics were not generated. Generally, not all characters in the block are completely lost if an algorithm puts in the wrong number of segments. Therefore, automatic printout of character blocks with too many or too few segments was included in the program. These printouts were later inspected to determine which, if any, character-pair segments were in error by less than 15 scans and thus should be included in the scan deviation statistics. The automated scan difference segment error analysis techniques were found useful for obtaining performance trends and for sorting out high performance algorithms. However, the statistics generated by counting segment errors are not directly comparable to other analysis techniques, such as counting the number of mis-segmented characters. Character error rate, a statistic more closely related to recognition figure of merit, is computed by considering the error in the two segments that isolate a character from its neighbors, not the error in each individual segment. For example, a character is put into the $+2$ scan error bin if either the segment on the left

M_1

n	$n-1$	$n-2$
0	0	0
0	0	0
0	0	0

M_2

n	$n-1$	$n-2$
0	–	–
0	X	X
0	–	X

or

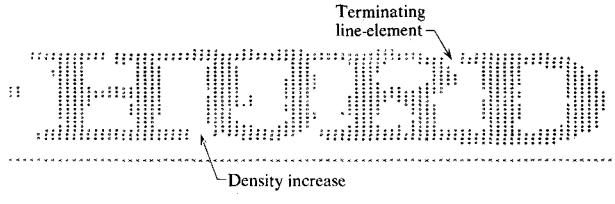
n	$n-1$	$n-2$
0	–	X
0	X	X
0	–	–

M_3

n	$n-1$	$n-2$
0	0	0
0	0	0
0	0	0

Figure 7 Measurements used in the "terminating line-element" segmenting algorithm. White bits are represented by 0's, black bits by X's and "don't care" bits by —'s. n is the current scan.

Figure 8 Sample of segments determined by the "terminating line-element" and "density increase" segmenting algorithms.



or the segment on the right or both are in error by +2 scans, and the other segment right or left, respectively, is in error by 2 or fewer scans. At the conclusion of the study, character error rates were generated for selected algorithms to establish a positive correlation between the two measurement techniques.

Test data set

During the development phase of this study, segmentation algorithms were tested and their parameters were optimized on a tape of approximately 18,000 characters. Most of these characters were from 12-pitch type sources. About 4500 segments (25%), involving about 40% of the characters, occurred at touching character pairs. The entire tape was coded with ideal segment information. For conclusive testing of the segmentation algorithms that evolved, a larger collection of characters was gathered. An IBM page scanner system scanned nearly 700,000 serif characters as obtained from field-generated documents printed on a wide range of 10-, 11- and 12-pitch printing devices. In this data set some 22% of the segments (more than 150,000) occurred at touching character pairs. To reduce the manual effort required to edit ideal segments onto this data, a statistically representative sample was chosen. The basic concern in sampling the 700,000-character test set was how large a sample to select in order to obtain reliable projections of error rates and to detect actual differences between segmentation algorithms. The following calculations describe how reasonable sample sizes were selected.

Assume that the performance of a segmentation algorithm can be described by a binomial distribution [4]. That is, the algorithm makes either a correct segment or an incorrect segment and these are the only possible outcomes. Also, assume that individual segments are independent events and the sample chosen is representative of the world.

If the probability that the segmentation algorithm makes an error is p, and we take a sample of segments of size n, we can show that the observed error rate p, and the sample variance σ^2, are related to n by

$$n = \hat{p}(1 - \hat{p})/\sigma^2.$$

From this sample, we obtain $\hat{p}$, an estimate of the actual error rate p. The error in this estimate is important. A more useful statistic for describing p is the "confidence interval." Since large sample sizes are involved, the normal approximation of the binomial confidence interval can be used. For example, the 95% confidence interval is

$$P(\hat{p} - 1.965\sigma > p > \hat{p} + 1.965\sigma) = 0.95.$$

In other words, the probability is 0.95 that the actual error rate p is greater than $\hat{p} - 1.965\sigma$ and less than $\hat{p} + 1.965\sigma$.

The above expressions may be used to relate an appropriate sample size to a desired confidence interval, the approximate error rates, and the allowable error range. For example, given a sample of 80,000 segments of which 5% are incorrect, and using the above relations, one can state that the error rate of that algorithm on the entire test set will be within the range 5% ± 0.15%, with 95% confidence.

For our tests we concluded that a randomly selected sample of 80,000 segments would give adequate error rate resolution and yet represent a feasible coding effort. About 22%, or some 16,000 segments, would occur at touching character pairs and thus require human observation. Should the actual error rate for algorithms operating on this sample be higher than expected, or should two algorithms have nearly equal sample error rates (which could preclude accurate relative ranking on the entire data set), an additional sample could be drawn, coded and tested. This was not necessary.

Test Results

Scan deviation statistics obtained by applying the topological and the quasi-topological algorithms to the sample set of 10-, 11- and 12-pitch serif patterns are summarized in Table 1. Detailed statistics by pitch and print quality are presented in Table 2. As described in the "Segmentation testing philosophy and procedures" section, three classes of segments are recognized. Those segments which differ by 0 and ±1 scans from the ideal are called correct, those differing from ±4 to ±15 scans from the ideal are definite errors and the segments at ±2 or ±3 scans from the ideal may be in error.

Table 1 illustrates that relatively small variations between algorithms in the 0, ±1 category indicate important characteristics of algorithm performance. This category contains almost all of the non-touching characters (rarely did any of the algorithms mis-segment a non-touching character) and some of the touching characters. Since the ratio of non-touching to touching characters was high in this data set (about 5 : 1), small changes in the percentage of correct segments imply larger changes in the percentage of correctly segmented touching characters, as Table 1 shows.

Table 2 presents segmentation scan deviation information in greater detail. The effects of pitch and quality are illustrated for each algorithm.

Discussion of results

- *Quasi-topological system*

Sectioning errors of the quasi-topological algorithm using stroke count and outline contour have two major sources. First, many errors are related to variations in width or pitch. The stroke pattern section algorithm uses a

Table 1 Summary of segment scan deviation data.

Segmentation algorithm	*Percent of segments differing from the ideal by*			*Percent of touching pairs within*	
	0, ±1	±2, ±3	±4 *to* ±15	0, ±1	0 *to* ±3
Topological	96.9	1.7	1.4	84.8	93.3
Quasi-topological	95.4	2.4	2.2	77.0	89.0

Table 2 Segmentation scan deviation performance comparison.

Pitch	*Character shade*	*Topological*			*Quasi-topological*		
		0, ±1	±2, 3	±4 *to* ±15	0, ±1	±2, 3	±4 *to* ±15
10	Dark	97.2%	1.6%	1.2%	96.9%	1.7%	1.4%
10	Light	98.0	1.1	0.9	97.5	1.1	1.4
10	Good	98.3	1.1	0.6	97.6	1.1	1.3
Total		98.0	1.2	0.8	97.5	1.2	1.3
11	Light	98.5	1.0	0.5	96.9	1.2	1.9
11	Good	98.9	0.6	0.5	97.5	1.3	1.2
Total		98.7	0.8	0.5	97.2	1.3	1.5
12	Dark	94.3	2.8	2.9	90.1	4.4	5.4
12	Light	97.2	1.8	1.0	95.5	2.9	1.6
12	Good	96.2	2.1	1.7	93.8	3.2	3.0
Total		96.0	2.2	1.8	94.2	3.3	2.5
Total		96.9	1.7	1.4	95.4	2.4	2.2

statistical masking system for finding the left character features and, like other masking schemes, the degree of match to a mask set depends in part on character width. The mask set was purposely tuned to 12-pitch characters and, as might be expected, on 10-pitch characters (which are generally wider) many blocks of touching characters were mis-segmented with too many segments. Design calculations indicated that it would not be reasonable to adjust parameters to tune the mask set to a compromise value and successfully handle both 10- and 12-pitch characters. Thus, for this section algorithm to be effective, an a priori character pitch (width) measurement should be available to adjust the mask width limits. A rather coarse measurement of pitch could be used. For example, the average width of previous non-touching characters might be sufficient. No experiments were conducted to substantiate this theory.

The second source of sectioning errors was responsible for many failures in 12-pitch data. On 12-pitch characters, where low error rates were expected, but were not obtained, many character blocks were mis-segmented with too few characters. Upon inspection, much of the 12-pitch data from the test set was found to be dark with heavy line strokes. There was a strong tendency for these characters to be filled with noise. Quite obviously, the stroke count unique to a given character will be obscured in this type of printing. As a result, sectioning action became highly unreliable. At the opposite extreme, a few light characters with thin or broken line elements were found. These characters, which lead to character blocks with too

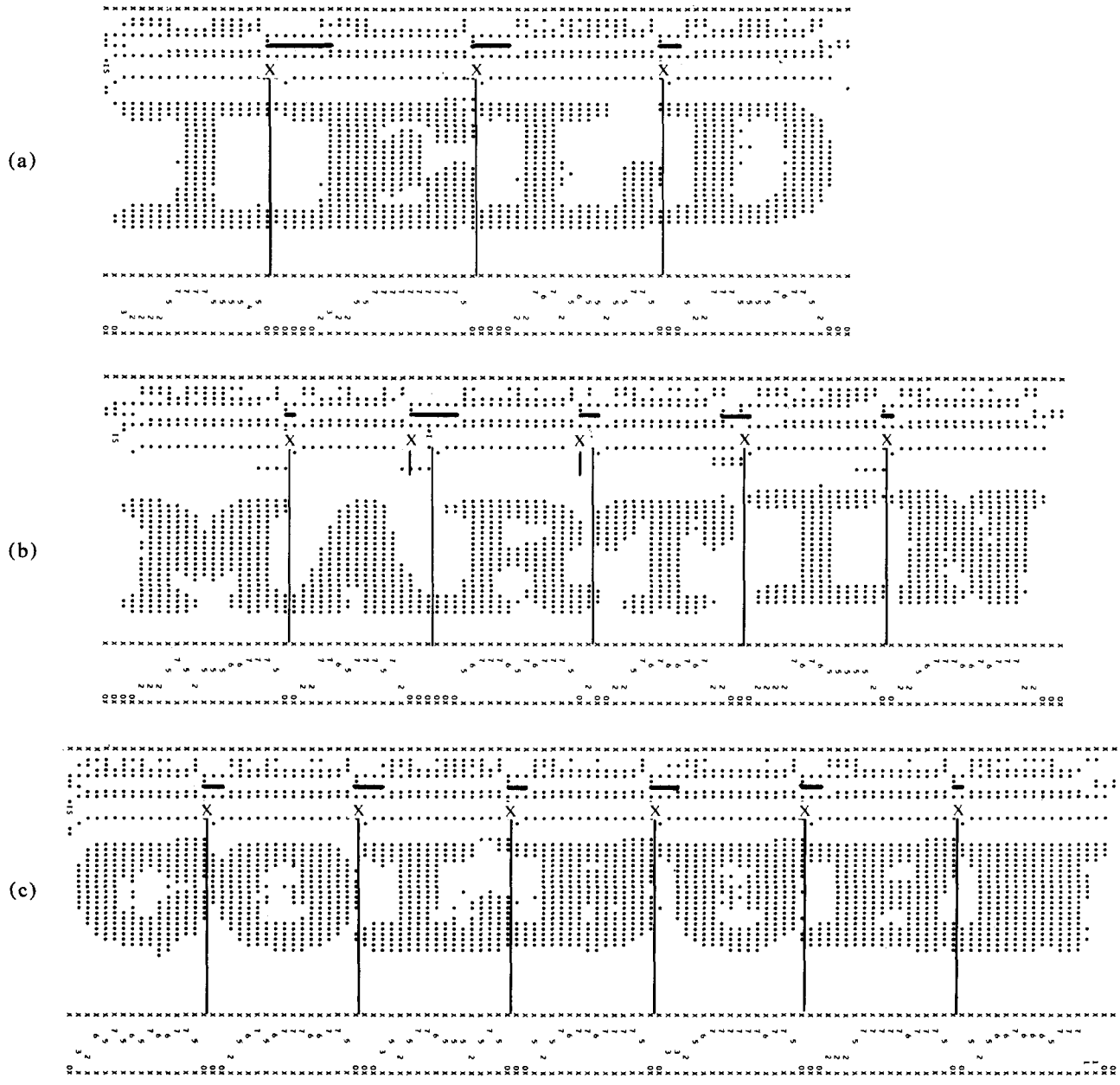


Figure 9 Examples of good segmentation on 10- and 12-pitch character sets.

many segments, may not produce as many errors since light characters are less likely to be linked together in long strings.

This experiment leads us to believe that in machine printing, the problem of video quality spread would preclude developing a high performance sectioning algorithm based on patterns in the horizontal stroke count derived from digitized video. The stroke pattern can be successfully computed only for printing of reasonable quality. The possibility of an analog video implementation exists but this same limitation is likely to appear.

• *Topological system*

Figure 9 shows a few examples of the topological segmentation system performance. These examples also illustrate the quality range of the data for which it was designed and on which it was evaluated. The vertical bars show the best human estimate of the optimum segment points. The horizontal bar in the retrace area above the characters indicates where the section algorithm was active. The X denotes the segment scan, that is, the first scan of the next pattern. Below each line in Fig. 9 is the running computation of the section threshold function for each scan.

In general, the topological sectioning algorithm demonstrated that its performance capability was substantially independent of pitch over the range of 10-, 11- and 12-pitch data on which it was evaluated. This was evidenced by its ability to correctly segment touching characters ranging from 12 to 13 scans wide up to 24 scans wide. Error causes were attributed primarily to characters with heavy noise fill-in, those which exhibited minimal terminating features, and cases where there was a significant feature overlap between adjacent characters.

The general form of the algorithm lends itself well to the broad range of optical character recognition applications. If the average character width is fixed or can be premeasured, the measurement coefficient feedback control can depend more heavily on width constraints. On the other hand, for proportional space and variable pitch applications, where a very large width variation must be tolerated, the coefficient feedback parameters may be updated by topology change, rather than by scan as has been described.

The discussion so far describes one form that the general algorithm might take for a particular application. However, the algorithm can be modified depending on the source data to which it is applied. For example, if the data are of particularly low quality with a high incidence of noise bits, the P and N measurements may be extended over more adjacent scans which, in effect, averages their incidence over a localized area. The form of the L measurement may be expanded over a larger vertical or horizontal area to account for extreme variations in line width or particular character geometries.

A different form of measurement coefficient control may be introduced by using the convolution property of the geometry of many characters. For example, characters like M and O are characterized by multiple line elements when the character is cut by a horizontal plane. The coefficients of the resulting N and P measurements for subsequent line edges following the first line edge can be increased or decreased, respectively, to support the section threshold function independent of character width for proportionally spaced data. Furthermore, for this application the connective feedback function might be updated at specific geometry changes rather than by scan to further reduce the character width dependence. Also, for application to predetermined-pitch fonts, constraints can be introduced more heavily by scan count than in the example shown.

Conclusions

Several results can now be established. First, extremes in video quality degrade the performance of the segmentation algorithms studied. While there are problems peculiar to individual algorithms, extremely light video with thin, broken lines generally leads to early segments and heavy dark video leads to late segments. Often the digitized video has little information for determining a segment point between two dark, hard-touching characters.

Second, from a performance standpoint alone, the results of this investigation demonstrated that topological segmentation was the preferred choice in the uncontrolled printing environment used for evaluation.

Third, a perfect segmenting-sectioning algorithm has not been found. What is not portrayed by our figures is how the segmentation errors are distributed among the character start, sectioning, and segmenting functions of each algorithm. For the algorithms developed during this study (topological and quasi-topological), human observation of segmentation errors revealed that the major source of error was due to improper sectioning. Segmentation errors chargeable to improper character start conditions were essentially nonexistent. Errors due to improper action by the segmenting function were present but were outnumbered by sectioning errors. In general, if sectioning activated segmenting in the correct area of the character, any error that resulted was relatively small in magnitude. To summarize, the majority of incorrect segments were caused by improper sectioning.

Acknowledgements

The authors acknowledge the design and programming contributions of A. Cutaia and D. W. Piller of IBM Rochester, Minn., and the personnel of the Recognition Systems Development Department at Rochester for providing consultation, documents, and the video collection facilities that made this study possible.

References

1. G. Nagy, "State of the Art in Pattern Recognition," *Proc. IEEE* **56,** 836 (1968).
2. R. B. Hennis, "The IBM 1975 Optical Page Reader, Part I: System Design," *IBM J. Res. Develop.* **12,** 346 (1968).
3. D. O. Claydon, M. B. Clowes and J. R. Parks, "Letter Recognition and the Segmentation of Running Text," *Inform. and Control* **9,** 246 (1966).
4. M. A. Mood and F. A. Graybill, *Introduction to the Theory of Statistics,* McGraw-Hill Book Co. Inc., New York, 1963.

Received August 8, 1969
Revised, June 24, 1970 and November 11, 1970

The authors are located at the IBM General Systems Division Laboratory in Rochester, Minnesota.