J. M. White
G. D. Rohrer

Image Thresholding for Optical Character Recognition and Other Applications Requiring Character Image Extraction

Two new, cost-effective thresholding algorithms for use in extracting binary images of characters from machine- or hand-printed documents are described. The creation of a binary representation from an analog image requires such algorithms to determine whether a point is converted into a binary one because it falls within a character stroke or a binary zero because it does not. This thresholding is a critical step in Optical Character Recognition (OCR). It is also essential for other Character Image Extraction (CIE) applications, such as the processing of machine-printed or handwritten characters from carbon copy forms or bank checks, where smudges and scenic backgrounds, for example, may have to be suppressed. The first algorithm, a nonlinear, adaptive procedure, is implemented with a minimum of hardware and is intended for many CIE applications. The second is a more aggressive approach directed toward specialized, high-volume applications which justify extra complexity.

Introduction

One of the most significant problems in Optical Character Recognition (OCR) is the conversion of nonideal analog images into ideal binary images [1]. The original documents which are scanned for characters are often dirty, multicolored, and produced by a variety of pens, markers, pencils, or printer mechanisms. Characters are often smeared or smudged, and are sometimes written with either very light strokes that are difficult to detect or very heavy strokes that tend to broaden and run together when imaged. The scanning hardware, due to technology and cost limitations, may have nonuniform illumination over the scan field, sensitivity and dark current variations from element to element in the sensing array, and nonideal resolution characteristics from the lens and from crosstalk in the array.

A similar but more inclusive thresholding problem may be called Character Image Extraction (CIE), which describes the suppression of unwanted background patterns so that only printed or handwritten characters may be captured as electronic images. As with OCR, this process involves converting nonideal analog images of characters into ideal ones, but the binary images may be compressed for storage or distribution, sorted, or used in computerized printing. The CIE process is different from digital facsimile, where a pseudo-gray-scale reproduction of the image is desired [2]. CIE output is binary, as opposed to multi-level gray scale, and consists of black picture elements (pixels) where character strokes are written and white pixels elsewhere. Pictorial content and "noise" which do not conform to the criteria for character strokes are eliminated. These images can be compressed more efficiently than digital facsimile and, therefore, are used for electronic distribution, sorting, and computerized printing as well as for OCR.

To overcome the difficulties of character extraction, the designer of a thresholding algorithm or circuit must use as much *a priori* information about the character images as is practical for the cost range of the equipment being designed. For example, the width of a typical character stroke is about 0.2 mm, with some of the widest strokes up to about 1 mm. (This stroke width encompasses most characters found in CIE applications.) The overall size of a character ranges from about 2.5 mm wide by 4.2 mm high (typewriter output) to 6 mm by 9 mm for hand-printed characters. Except for

dots on the "i" and "j" and for punctuation marks, characters are made of strokes which are ideally long, but narrow, connected groups of black pixels. The thresholding circuits should delete background levels which are changing over regions larger than the character size. Maximum detectability should occur for dimensions appropriate to the character stroke. Ideally, this last maximization process should not be done independently for each dimension in the two-dimensional space. This double maximization would emphasize dots and dot-like noise more than lines, which have high spatial frequency components in only one direction.

The decreasing cost of digital image capture and processing hardware, especially CCD-scanned (Charge Coupled Device) photodiode arrays and memory chips, has made it possible to consider approaches to thresholding that have not been practical before. Interline pauses required by some systems make digital thresholding more favorable than strictly analog thresholding. The former can be controlled by the system clock and time-independent digital memory, whereas analog time constants are fixed and require uninterrupted operation.

State-of-the-art document scanning systems provide discretely sampled output on a rectilinear grid. One typical grid is 240 pixels/inch (approximately 0.1 mm/pixel) both horizontally and vertically. This provides an average of at least two samples, or pixels, per stroke width—a condition which guarantees that at least one sample will fall totally within the stroke. This also produces 5.4 million pixels for a typical 8.5 by 11-inch sheet of paper or 1.0 million pixels for a typical 2.75 by 6-inch bank check. Processing this amount of data at high speeds requires special real-time processing algorithms in order to minimize hardware costs. The approaches we have developed and tested, and which form the basis of this paper, are thus significantly different from typical low-speed, iterative digital processing of photographic or satellite data [3].

The first approach is a dynamic threshold algorithm [4]. The black/white decision is determined by a threshold level which is continually changed as the scanned gray-scale data stream changes. The basic threshold calculation may be represented (in each dimension) as a first-order difference equation with a nonlinear coefficient. The nonlinear term was heuristically determined, but was modeled after the response of a resistor-capacitor-diode circuit. Not only does this algorithm result in near-optimal image fidelity, it also can be built with a very small amount of logic and memory hardware and can be programmed with very few lines of code if a microprocessor implementation is preferred. This ensures low cost as well as high-speed performance. Although not described in detail in this paper, a modification of the dynamic threshold algorithm was created for calculating the threshold across a segmented scan field (with eight parallel outputs from the scanning array). This modification was simulated in software, not in real-time hardware, with successful results in image quality.

The second algorithm is a unique combination of simple algorithms that more fully utilizes the linear width and connectedness of character strokes. Some of the simplicity of the first thresholding algorithm is sacrificed, but typically more of the pixels in the background region are made white. This results in a more idealized output, and the improved output is more noticeable when the scanned gray-scale data are distorted or have dark background regions. Unlike other multi-operational algorithms that can achieve this idealized output [5], the black/white decision in the second algorithm is determined in only one processing pass of the scanned data. This greatly reduces the complexity of the implementation and allows the algorithm to be economically feasible for high-speed imaging.

This algorithm is a label and search process. Before the final black/white decision is made, the pixels lying near an edge (sharp change in gray-scale data) are labeled. Pixels located on the dark side of an edge are distinguished from those on the light side. The light and dark sides of an edge are identified by a sum of the differences calculation that is an approximation of the Laplacian operator (two-dimensional second derivative). The final black/white decision is based on a search operation of the labeled image. Unlike the results with our Dynamic Threshold and many other algorithms [6], the sharp edge of a large background pattern is eliminated by this algorithm since the other edge does not appear within the specified distance. Pixels within a character stroke are made black because associated edges can be found. The high spatial frequency associated with the edge is not sufficient in itself to cause a black/white transition in the output.

Dynamic Threshold Algorithm

• *Functional description*

The main objective of the Dynamic Threshold Algorithm is to set a threshold for the binary (1 for black, 0 for white) decision about a given pixel. The approach conceptually is to compare the gray value of the pixel with some average of gray values in some approximately character-size neighborhood about the pixel. If the pixel is significantly darker than the neighboring pixels, it is called black. Two difficulties arise with the obvious approach of uniformly averaging the gray values in a circular neighborhood. The first problem is one of cost: Storing many lines of gray-level pixel data becomes prohibitive. An averaging approach must be developed which does not require referral back and forth to other lines of scan data. The second problem is one of performance: If the contrast ratio of the character to the background is

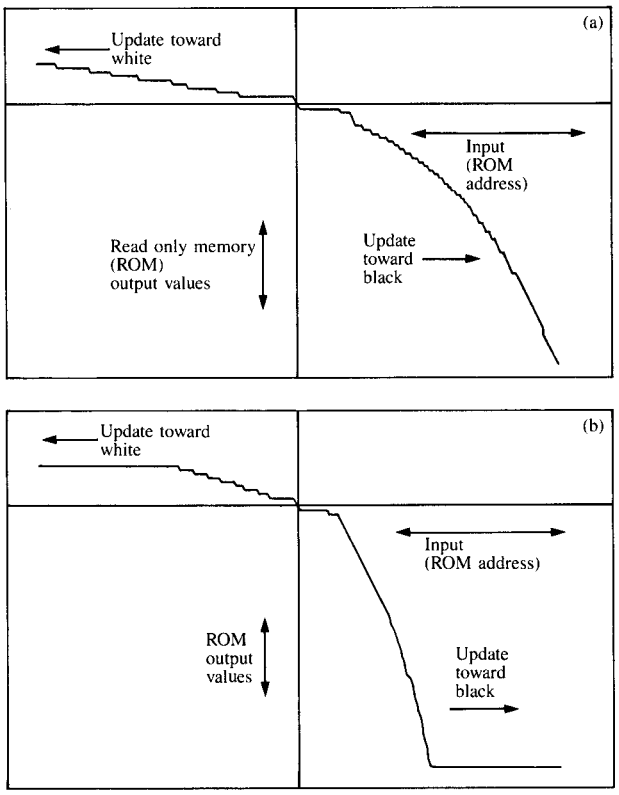

Figure 1 Update functions f (upper curve) and g (lower curve) provide the rate of response of the running average to changes in the gray values input to the Dynamic Threshold Algorithm hardware. Abscissas vary from -127 to $+128$, representing the sign and the seven most significant bits of the difference. Ordinate values vary from 10 to -63.

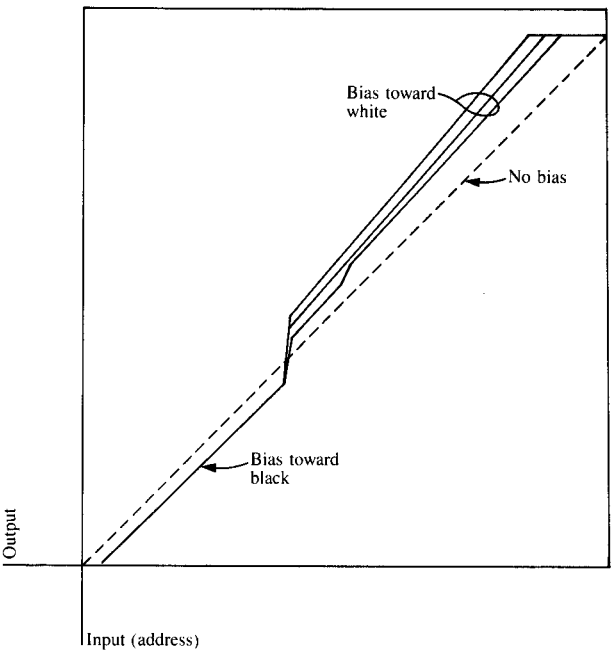

Figure 2 Bias functions are used to offset the decision level and to eliminate noisy backgrounds. Since the input is 6 bits (values 0 to 63) and the output is 8 bits (0 to 255), a multiplication of approximately four is included in the table, which scales the output to the range covered by the average.

high, then the threshold level should be increased by an additional amount to reduce "noise" from dirt, smudges, background printing, etc. Furthermore, the "average" must adapt quickly after leaving a very dark character so that a following lighter character will not be eliminated.

The solution to the storage requirement problem for averaging is to use a "running" average instead of a true average. To calculate a running average, $y(n)$, from a stream of sampled, digitized, raster-scanned gray values, $u(n)$, a fraction, f, of the input gray value is added to a complementary amount of the previous average, $y(n-1)$:

$$y(n) = f \cdot u(n) + (1 - f) \cdot y(n-1). \quad (1)$$

This equation can be expanded to show explicitly that $y(n)$ is a nonuniformly weighted average of the current and *past* pixels:

$$y(n) = \sum_{i=0}^{\infty} f \cdot (1-f)^i \cdot u(n-i). \quad (2)$$

(Note that, for negative arguments, u is taken to be the value set in the history buffer when the thresholding system is initialized.) However, to implement the running average, it is instructive to rewrite Eq. (1) as

$$y(n) = y(n-1) + f \cdot [u(n) - y(n-1)]. \quad (3)$$

Thus, the average can be updated by adding to the previous value a fraction of the difference between the current gray pixel value and the prior average value. When implemented in hardware, this latter expression for the running average requires very few components.

Replacing the constant value f in Eq. (3) with a function f gives a more versatile, nonlinear equation:

$$y(n) = y(n-1) + f[u(n) - y(n-1)]. \quad (4)$$

Certain restrictions should be applied to the function f. It should equal zero only when its argument is zero, and otherwise it should have a value between zero and the value of its argument. This will guarantee that the average never exceeds the range of the input gray pixel values and that the average will converge to the input value for uniformly gray areas.

The nonlinearity permitted by the use of a function instead of a multiplicative constant provides a solution to the second problem described above; i.e., the average can be made to

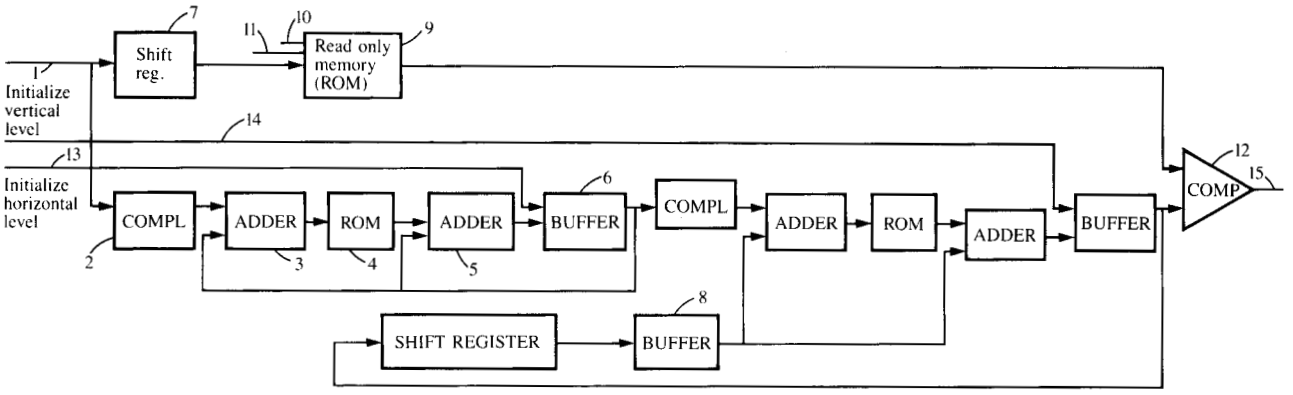

Figure 3 Components used to implement the Dynamic Threshold Algorithm. Inputs 13 and 14 are used to initialize the horizontal and vertical averages at the beginning of each line and column, respectively. Input 1 is 6-bit gray scale data from the A/D converter connected to the scanning array. Output 15 is the binary result of the thresholding.

adjust rapidly to large, high-contrast signals, and to have a tendency to follow the black peaks of the character stroke pixels. The peak-following characteristic is similar to that of a rectification circuit, and in fact, the function f illustrated in Fig. 1(a) is similar to the current-voltage characteristics of a leaky diode. Other thresholding implementations [7] have actually used diodes in a peak-following scheme, but the analog approach does not allow fine tuning of the adaptation rate and is not clock-controlled as is the digital approach.

Equation (4) only gives one-dimensional averaging. Images are two-dimensional, and experience has shown that two-dimensional averaging greatly enhances performance. To achieve a two-dimensional running average, vertical averages, $z(n)$, are stored for each column of the image, and as that column is reached, the vertical average is updated by the horizontal average value $y(n)$:

$$z(n) = z(n - \ell) + g[y(n) - z(n - \ell)], \tag{5}$$

where ℓ is the number of pixels in a scan line. Thus, we have a vertical average of the horizontal average. The update function, g, shown in Fig. 1(b), operates more rapidly than the first-stage, horizontal update for two reasons: The first stage usually eliminates the extreme values; also, there is little "look-ahead," if any, in the vertical direction.

Since the running average is one-sided in that it only averages over past pixels, it is desirable to store some number N of scanned pixel values and use this delayed value, $u(n - N)$, in comparison with the dynamic threshold or average $z(n)$. We found that $N = 8$ was a good choice for data scanned at 240 pixels/inch. This number is four times the number of pixels in the narrowest of "typical" strokes. A minimal additional improvement was observed when we also added an entire line to the delay; i.e., $N = 8 + \ell$.

One other feature is required to complete the Dynamic Threshold Algorithm, and that is a bias between the gray value of the pixel being compared, $u(n - N)$, and the two-dimensional average, $z(n)$. Without bias, the threshold decision would be determined by noise fluctuations in uniform areas. Additional amounts of bias are required to guarantee the suppression of residual images of the specially colored boxes used in many OCR forms. Color filtering of the optical image eliminates most of the contrast from these boxes, but due to the variety of inks which are used and the tolerances which are specified to accommodate these inks, biases of five to twenty percent may be required for some applications.

The bias may be a function of the history, $z(n)$, of the localized pixel, $u(n)$, or of both. In our implementation, the biasing function, h, was based only on the localized pixel. That is, $h[u(n - N)]$ was compared with $z(n)$. Figure 2 illustrates typical biasing functions. The dashed line indicates the unbiased condition in which $h[u(n)]$ is equivalent to $u(n)$. For those cases indicating a bias toward white, the output decision will be white unless the pixel is definitely darker than the neighborhood. Conversely, if the pixel is dark on an absolute scale, then the output decision will be black unless the pixel is relatively lighter than its neighborhood by the indicated amount. Various bias curves are indicated and were selectable for the various color drop-out modes of operation.

Implementation

The low cost of implementation makes the Dynamic Threshold Algorithm potentially useful in many OCR applications. Figure 3 is a block diagram of the implementation in dedicated digital hardware. A flow diagram for a simple microprocessor software implementation would be very simi-

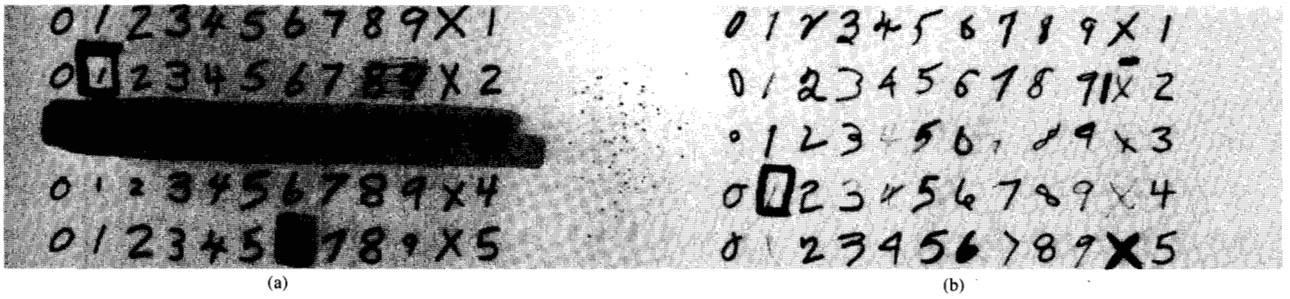

Figure 4 Pseudo-gray reproduction of stress test document data as scanned.

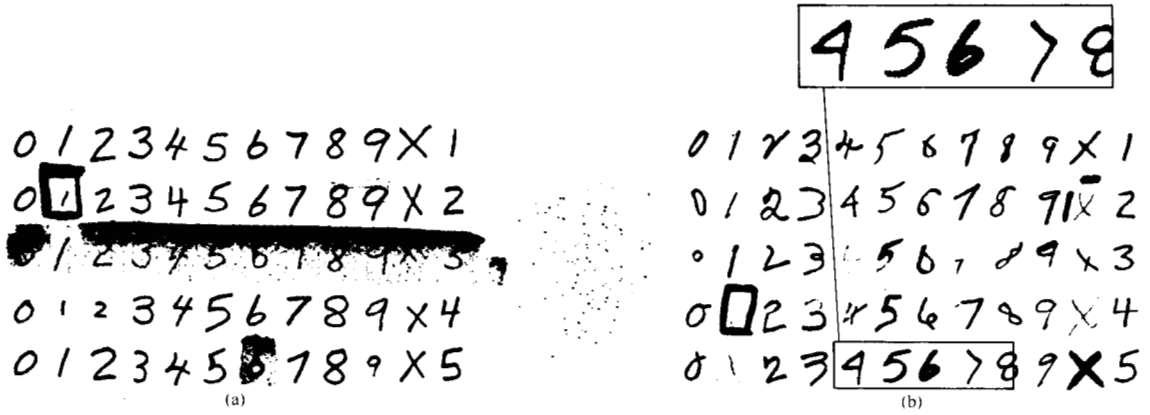

Figure 5 Result of using the Dynamic Threshold Algorithm on data of Fig. 4. Insert is an enlargement of lower right numbers. Each pixel corresponds to 1/240 inch in the original document.

lar. Several nonobvious manipulations of the data stream were used to keep the number of parts to a minimum without sacrificing quality.

The data from the analog-to-digital converter (ADC) was 6 bits per pixel, representing gray levels of 0 to 63. The 6-bit-wide data path is indicated by 1 in Fig. 3. The subtraction of $u(n)$ from $y(n-1)$ was performed by complementing $u(n)$ and adding it to $y(n)$, as indicated in blocks 2 and 3. The horizontal average $y(n)$ was an 8-bit value, ranging from 0 to 255. The six bits from block 2 were added in block 3 as the most significant bits—in effect multiplying $u(n)$ by 4. This permitted the updating of history values by 1 part in 256, instead of 1 part in 64, so that very slow rates of updating could be realized.

The carry bit from the adder (3 in Fig. 3) along with the seven most significant bits were used to address ROM 4 which stored the function f. This incremental result from the ROM was added to the buffered history value in block 5, and the new value $y(n)$ was passed to buffer 6.

A similar arrangement is used to perform the vertical averaging. The only difference is that a shift register 7 and buffer 8 are used to store the line of history values and provide the delay ℓ. The resulting value $z(n)$ is applied to one side of an 8-bit comparator. The upper data path consists of the shift register 7, which provides the delay N between the average term and pixel under comparison, and the bias table, ROM 9, which took a 6-bit input and provided a biased, 8-bit output so that the upper channel would in effect also have the 4 times multiplication. Since the ROM had a 9-bit input, the three extra inputs (two are shown as 10 and 11) could be used to select eight different bias tables.

- *Results*

The Dynamic Threshold Algorithm has been tested with a wide variety of documents. OCR statistics on thousands of

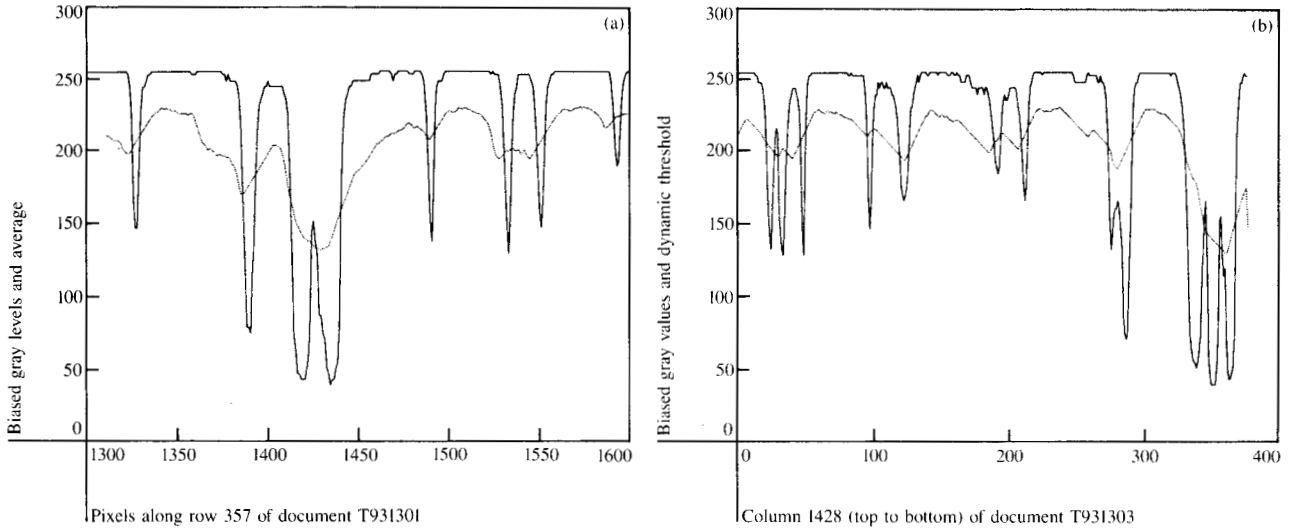

Figure 6 Biased gray levels (solid curve) and two-dimensional running average (dotted curve) for the horizontal line (a) and the vertical column (b) intersecting in the lower right "6" of Figs. 4 and 5. Intersection coordinates are 1428 and 357. Output of threshold is white when the biased gray level is larger and is black when the average is larger.

characters scanned from "live application" documents and thresholded using the Dynamic Threshold Algorithm hardware compared favorably with results using prior thresholding methods. For purposes of illustration, we have selected three document samples which exemplify worst case problems (Figs. 4 and 5), a typical application (Fig. 8), and an idealized image test pattern (Fig. 9).

Figure 4 is a computer/photocomposer pseudo-halftone reproduction of the actual gray-scale data scanned from a stress test document. The reading of all characters on this document far exceeds the capabilities of most OCR machines (some characters are too light, others are smudged, erased, or marked over); in a number of instances the minimal-hardware, Dynamic Threshold Algorithm fails. However, it is instructive to see what the problems are and how closely we border on a "failure."

Figure 5 shows the output of the Dynamic Threshold Algorithm operating on the data of Fig. 4. The "6" in the bottom row of Fig. 5(b) is an example of a common problem—the center of a loop is almost filled in but needs to be open for best operation of the recognition logic. The gray value in the center of the loop is actually darker than many of the other character strokes, as can be seen in Fig. 6, which shows the bias gray levels and average output values as they would appear at the input of the comparator (12 in Fig. 3). The opening of the 6 is located approximately at the intersection of column 1428 and row 357. (The document is 2048 by 384 pixels). The threshold dynamically shifts to capture the opening of the 6 as desired.

If a dark area is roughly the size of a character, then it will tend to remain dark. Even with biasing, which strongly emphasizes contrast since these levels straddle the break in the bias curve (Fig. 2), the center left row of characters [Fig. 4(a)] is not machine-readable. Figure 7(a) indicates that there is not enough time for the vertical average to be adjusted to such a dark background during the processing of one character line, but if the problem line is replicated [as in Fig. 7(b)], then the likelihood of detecting and reading the characters is enhanced for the second line of characters in the dark background.

Figure 8 is a composite of a form using a green "drop-out" background with boxes to constrain the location of characters to be recognized and of segments of thresholded output. The bias function of the Dynamic Threshold Algorithm does ignore the residual contrast of the boxes against the white paper. The area about the scanned characters is then free of any "noise" which would reduce the effectiveness of the recognition process.

Figure 9 is a portion of the IEEE Facsimile Test Chart, which has been scanned and thresholded using the Dynamic Thresholding Algorithm. In addition to demonstrating the high resolution of the 240 pixel/inch system, the portion of the face shows how the bias toward black (lower left part of the curves in Fig. 2) retains the overall pictorial content of the large dark areas. Were the bias toward white for all gray values, then *any* large uniform area would appear as white with only transition areas having black output pixels. This is, of course, an option of the user and is controlled by the bias

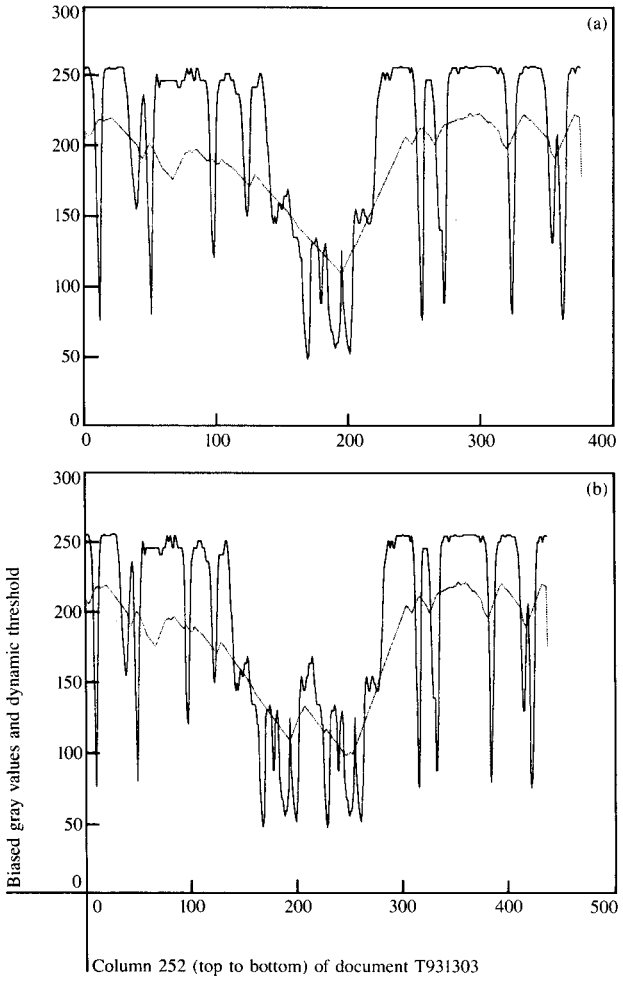

Figure 7 Biased gray levels (solid curve) and two-dimensional average (dotted curve) along column 252 of Figs. 4 and 5. Excessive overall darkening of a character-sized area results in a loss of information (a), but replicating the dark line (b) shows that the threshold tends to adapt as desired when the dark area is larger than a single character dimension.

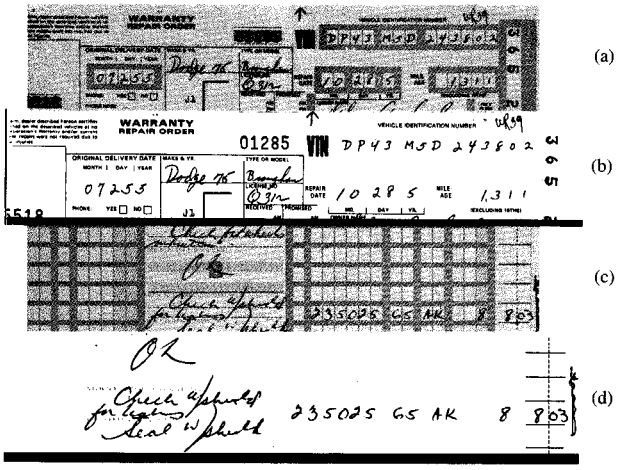

Figure 8 Sections of actual OCR document (a, c) with green "drop-out" boxes surrounding the OCR regions, and (b, d) Dynamic Threshold Algorithm output of corresponding areas.

Figure 9 Result of scanning IEEE Facsimile Test Chart with Dynamic Threshold Algorithm. Reproduction of the large black areas in the face section is accomplished by biasing the lower gray values toward black; otherwise, all largely uniform areas would be white.

tables selected for use with a particular document. As shown in Fig. 2, the bias tables which were used change from bias toward black to bias toward white at 40 percent of full scale.

Integrated function algorithm

• *Functional description*

The second algorithm is intended for extracting characters from complex images in an effective and rapid manner. The main objective is to remove as much of the nonessential background as economically feasible to allow for efficient compression and subsequent handling of the binary image. In some CIE applications this can be difficult to achieve because the contrast ratio of text to the background is extremely small. For bank checks, as an example, a ratio of less than 20% is not uncommon, while ratios within the background alone can exceed 20%. This obviously restricts the use of contrast as the sole measurement in thresholding. Other measurements should be made and appropriately weighted to generate a more idealized output. The Integrated Function Algorithm, therefore, employs several measurements to deal with these complex images. Width of the text, sharpness of the text edges, and constant ratio all contribute to the threshold decision process.

The Integrated Function Algorithm processes images in the following fashion: First, digitized data from the raster scanner are processed by gradient-like operators to identify and label pixels in, or very close to, areas where sharp changes exist in the gray-level image. These regions are typically edges of text (characters) or background areas having high contrast change. This localized edge information is then interrogated to separate the text edges from those

belonging to extraneous background. The separation decision is based on correlating these edges to character stroke widths. Then, internal regions of the character strokes are made black (binary 1) while all other areas are made white (binary 0).

To accurately identify the pixels in the vicinity of an edge, a measurement of the changes in gray-level values is used. The measurement, defined as activity operator $A(i, j)$, is the absolute sum of approximated derivatives for both scan and raster directions taken over a small area. The derivative, dx, as proposed by Sobel [8], in the scan direction for the gray-level u located at pixel i in raster j is

$$dx(i, j) = u(i - 1, j) - u(i + 1, j). \tag{6}$$

Similarly,

$$dy(i, j) = u(i, j - 1) - u(i, j + 1). \tag{7}$$

The change activity for one pixel, $a(i, j)$, is defined in this algorithm as the absolute sum of these Sobel derivatives:

$$a(i, j) = |dx(i, j)| + |dy(i, j)|. \tag{8}$$

To maximize the detectability of edges for most hand- and machine-printed characters, the activity defined in Eq. (8) is summed over nine pixels. The form of the activity operator for one pixel, $A(i, j)$, is then

$$A(i, j) = \sum_{n=-1,0,1} \sum_{m=-1,0,1} a(i + n, j + m). \tag{9}$$

This operator takes on large values in the vicinity of character edges and relatively lower values elsewhere. A simple thresholding technique, therefore, is sufficient to pick only those pixels lying close to character edges. One need not be too concerned about incorrectly identifying some pixels. Their positions are usually uncorrelated (not related to a stroke edge), which results in their being removed or properly labeled in the black/white process. Figure 10 shows the histogram of $A(i, j)$ values collected from a high-speed scanner. The operator is well-behaved, as demonstrated in the distributions for a uniform, black background to the very busy background of a scenic bank check. Few pixels exceed the value of 25 in all cases. $A(i, j)$ behaves much the same in the presence of text. This is shown in Fig. 11. Threshold values between 15 and 24 are sufficient to separate most background pixels from those in the vicinity of character edges.

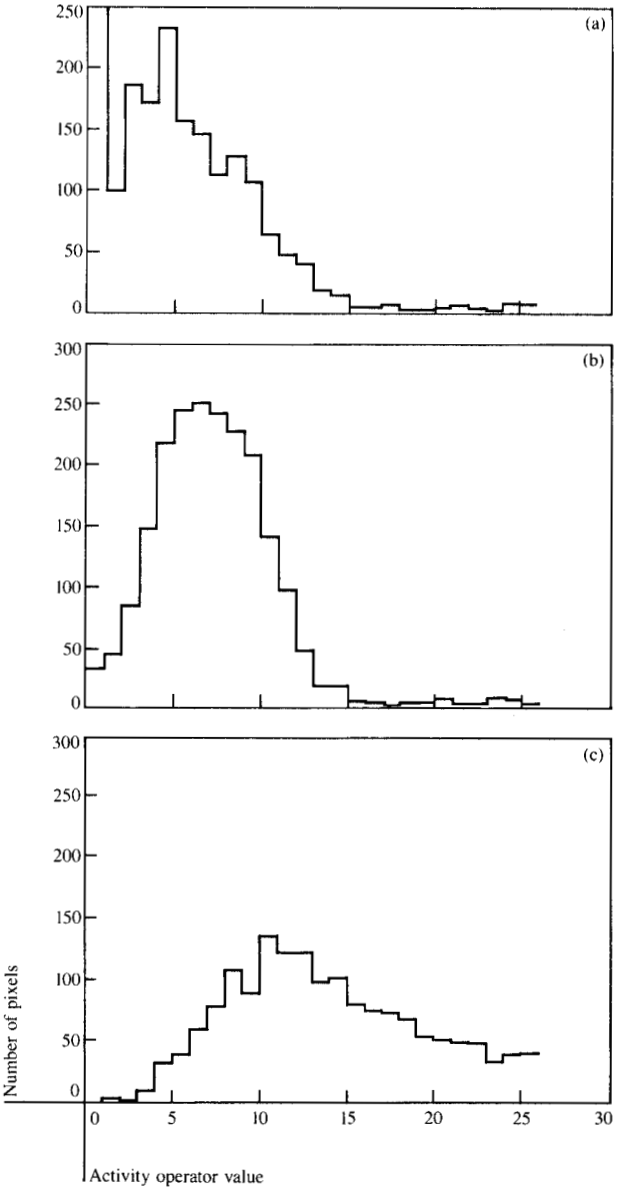

Figure 10 Activity operator histograms. Figures show activity operator, $A(i, j)$, histograms for areas comprised of 2560 pixels, 20 by 128, at 240 pixels per inch. Gray-level data, $u(i, j)$, is 6 bits per pixel, representing values from 0 to 63. (a) Scanner background (noise); (b) plain white paper; and (c) scenic background of a bank check.

The next step is to label all identified edge pixels according to the sign of an approximated Laplacian operator, $ddxy$. Referring to Eqs. (6) and (7), the operator for pixel (i, j) is

$$ddxy(i, j) = dx(i + 1, j) - dx(i - 1, j) + dy(i, j + 1) - dy(i, j - 1). \tag{10}$$

In terms of gray-level values, u, the operator simply is

$$ddxy(i, j) = u(i + 2, j) + u(i - 2, j) + u(i, j + 2) + u(i, j - 2) - 4 \cdot u(i, j). \tag{11}$$

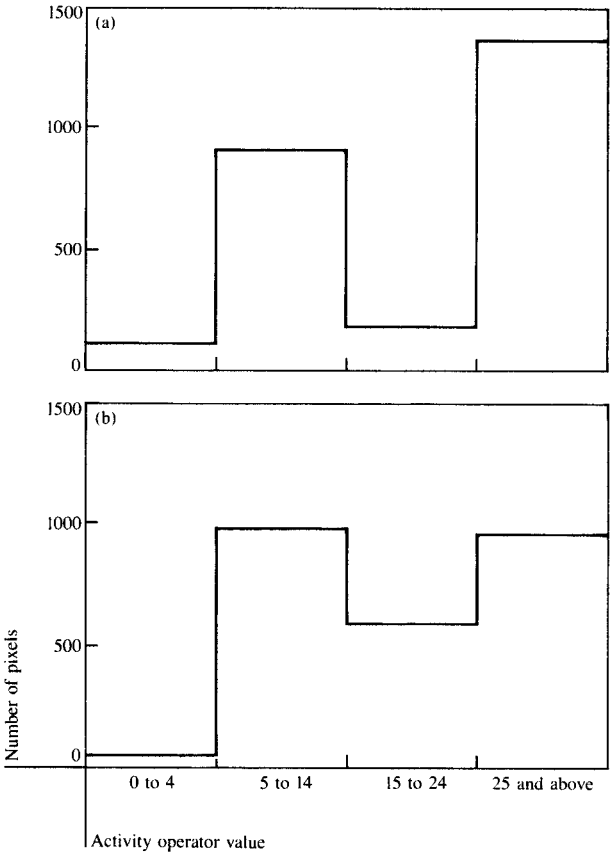

Figure 11 Activity operator influenced by text edges. Scan size and resolution are as described in Fig. 10. (a) Distribution of $A(i, j)$ values for text on a clean background; (b) text on a scenic bank check.

Historically, this operator is not well behaved on typical image data [9]. In this algorithm, however, *ddxy* is only applied for those pixels having an activity operator value, $A(i, j)$, greater than or equal to a threshold, T. This effectively smooths *ddxy* because the Laplacian tends to be stable in these areas.

This combination of $A(i, j)$ and $ddxy(i, j)$ is formed to generate a three-level image in which only the sharpest edges are identified and labeled. A pixel in the new image, $S(i, j)$, is defined as

$$S(i,j) = \begin{cases} 0 \text{ if } A(i,j) < T, \\ - \text{ if } A(i,j) \geq T \text{ and } ddxy(i,j) < 0, \\ + \text{ if } A(i,j) \geq T \text{ and } ddxy(i,j) \geq 0. \end{cases} \quad (12)$$

The inequalities show that the sign of *ddxy* is the label applied to the pixels near an edge. Pixels on the dark side of an edge are labeled +, those on the light side are labeled −, and all others are 0. Figure 12 shows a hand-written stroke image labeled in this manner. The horizontal line underneath

Figure 12 Three-level edge-labeled image. Internal areas of the character and machine-printed strokes are labeled + with the exception of some pixels in the top of the "o." These are still bounded by edges. Noise has caused some pixels to be incorrectly labeled (top of the image). These are uncorrelated and are not bounded by edges.

the stroke is machine-printed. The internal areas of both lines are + because they are darker than the background.

Elementary correlation techniques can now be applied to the S image to create a final, binary black/white image. Any pixel that is internal to a character stroke should be either 0 or + and bounded by ordered sequences as illustrated below:

$$\cdots, -, +, \cdots, [S(i,j) = 0 \text{ or } +], \cdots, +, -, \cdots,$$

where the first and last ellipses indicate any combination of +, −, and 0, and the inner ellipses represent any combination of + and 0. These internal pixels should be black in the output image. Background pixels tend not to be bounded by this ordered sequence and should be made white. For example, they could be within two opposing changes in contrast:

$$\cdots, -, +, \cdots, [S(i,j) = 0 \text{ or } +], \cdots, -, +, \cdots,$$

where each ellipsis is defined, respectively, as in the earlier expression. All pixels with a − label should be made white.

Line width restrictions can be factored into this process to further improve the output image. The bounded sequences should be along some straight line passing through $S(i, j)$. The distance between character edges (− +, + −) along this

line could be calibrated to the line width restrictions, so that wide, dark background areas can now be dropped from the final image.

• *Implementation*

A block diagram showing the major processing functions for this algorithm is presented in Fig. 13. All functions could be realized in one or more microprocessors, depending on processing time requirements. Calculation of the edge-finding and labeling factors should be done in parallel for efficient processing. The edge correlation for generating the final image can be buffered in an economical fashion and performed in another processor operating serially. For high-speed applications parallel processing channels could be used. Each channel has to overlap the adjacent channels by a line width to ensure that no seams are created in the binary image.

The algorithm was implemented in a special-purpose processor. Digitized input from the A/D converter is identical to that described for the Dynamic Threshold Algorithm. The derivatives, dx and dy, for each pixel are calculated with two adders and enough RAM to hold two scan rasters of gray levels. Operators $ddxy$ and A are calculated in a similar arrangement. The threshold, T, is set to be a binary fraction of A values and is bounded by 15 and 25. The threshold is either increased or decreased depending on the number of A values between 15 and 25 in eight rasters.

To limit the amount of hardware and to reduce processing time required to do the edge correlation, the edge search is limited to two directions, x and y. A line or raster store of the S image provides the data necessary to test for the ordered sequences in the x direction. Data for the sequence test in the y direction are accumulated through keeping a history of the most recent edge sequence for each pixel. Only the sequence type (1 for +, − and 0 for −, +) and the number of rasters since it occurred (binary coded run-length) need to be stored. This search matrix is shown on a typical S image in Fig. 14. The basic criterion for making a pixel black requires that the pixel be bounded by edges only in one direction.

More accurate decisions can be made with little added expense by expanding the decision matrix to include two edge tests for each direction (double search decision matrix). Here a four-pixel cluster (2 by 2) is evaluated simultaneously with the edge search in the two rows and columns that contain the cluster. The pixels must be bounded in both rows or both columns to be made black.

The Integrated Function Algorithm was implemented using the double search decision matrix. This further enabled us to remove much of the nonessential background from busy or complex images.

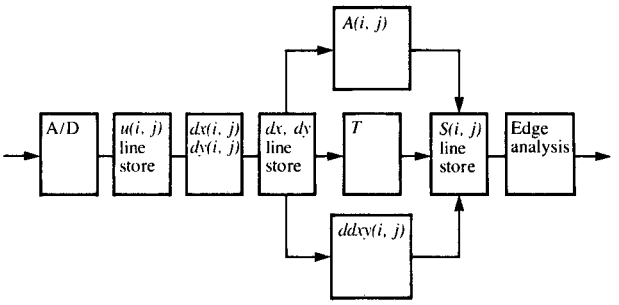

Figure 13 Block diagram of the Integrated Function system.

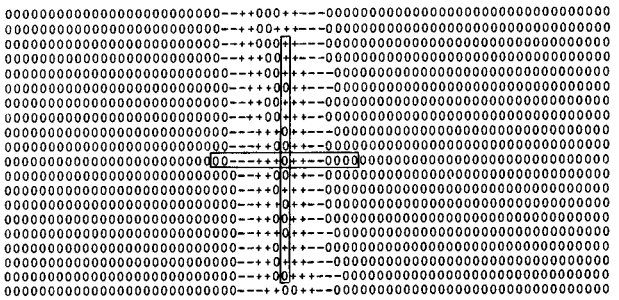

Figure 14 Decision matrix positioned on an internal pixel $S(i, j)$ of a typical stroke image. The edge search is limited to two directions and eight pixels. This is sufficient to define character line widths up to 1 mm for a scanner resolution of 240 pixels per inch.

• *Results*

To evaluate the effectiveness of removing background, we tested the algorithm with a variety of scenic bank checks. The gray-level images of these checks were collected from a high-speed scanner (low signal-to-noise ratio). Thus, in addition to the busy backgrounds, these gray levels contained distortions created by the scanner hardware.

Figure 15 shows small portions of bank check images produced by two versions of the algorithm. The 120/240-pixel images illustrate the benefits of using the double search decision matrix. (The 240-pixel output represents the single row/column search matrix.) Considerably more of the background is dropped from the output in the 120/240-pixel images. The four pixels in the 2 by 2 cluster in the double matrix images are made either all black or all white, leading to a 120-pixel output resolution. This shows that much of the character shapes can be maintained through a reduction in resolution. The double search matrix can also be used to produce binary images at the scan resolution.

Figure 16 shows a scenic bank check and its 120-pixel binary image produced by the Integrated Function system

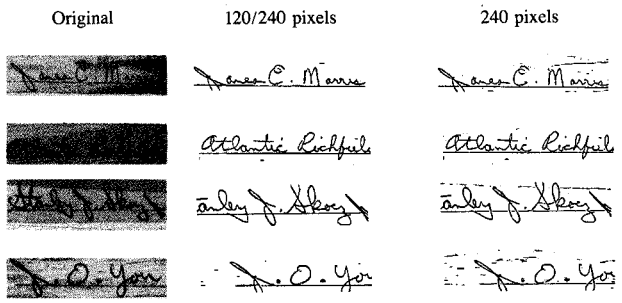

Figure 15 Binary images produced by Integrated Function Algorithm thresholding. Originals (left) were scanned at 240 pixels per inch, creating gray-level images of 128 by 512 pixels. Center images (120/240 pixels are 120-pixel-per-inch binary images produced from 240-pixel-per-inch gray levels. Binary images on right (240 pixels per inch) are produced from the same set of gray levels without the 2:1 reduction feature.

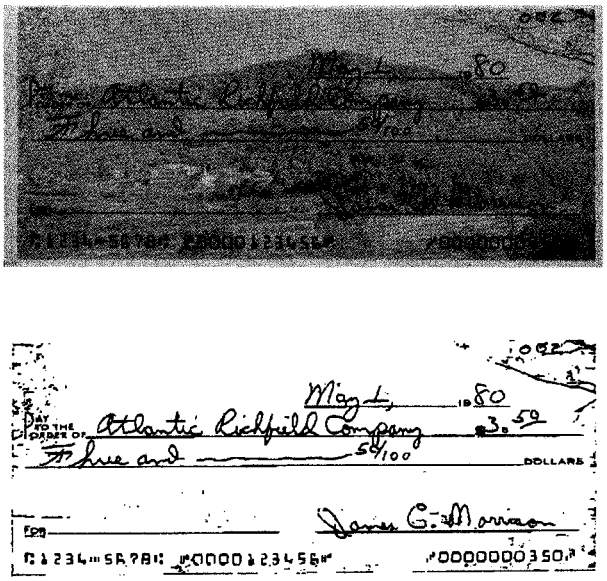

Figure 16 Binary image of a scenic bank check. Document (original at top) was scanned at 240 pixels per inch creating a gray-level image of 660 by 1440 pixels. The binary image (below) produced by the Integrated Function Algorithm is 120 pixels per inch.

with the double search decision matrix. Although the background contains marked changes in contrast, much of it has been removed. Some of the background, however, does correlate with normal stroke widths and is retained. This can be readily seen in the upper right corner of the image.

Summary

The conversion of nonideal analog images into digitized images is a significant problem. Text (characters) must be extracted from unclear backgrounds in a cost-effective manner. This paper has described two solutions to this problem. The Dynamic Threshold Algorithm provides near-optimal performance and can be built with a very small amount of hardware. The test results indicate that this algorithm is appropriate for many CIE applications. By sacrificing some simplicity, the Integrated Function Algorithm is capable of producing more idealized output from very busy or complex documents. Scenic bank-check images have been used to illustrate the background removal capabilities.

Acknowledgments

We remember with appreciation the support of the late Georg Gaebelein in the OCR activity, and we would like to thank co-inventors R. L. Melamud and J. D. Nihart for their contributions to the Dynamic Threshold Algorithm and G. A. Davidson for providing the software to model and test the Integrated Function concept. D. G. Abraham provided assistance in the modeling and testing of the Integrated Function Algorithm, and S. J. Skocz collected the images for Figs. 15 and 16. F. C. Mintzer reproduced Figs. 4, 5, and 9 for us on an electronic photocomposer at the IBM Thomas J. Watson Research Center in Yorktown Heights, NY.

References

1. J. R. Hicks and J. C. Eby, Jr., "Signal Processing Techniques in Commercially Available High-Speed Optical Character Reading Equipment," *J. SPIE* (Society of Photo-Optical Instrumentation Engineers) **180** (Real-Time Signal Processing II), 205–216 (1979).
2. J. M. White, "Recent Advances in Thresholding Techniques for Facsimile," *J. Appl. Photographic Engr.* **6**, 2, 49–57 (April 1980).
3. H. C. Andrews, *Digital Image Processing*, IEEE Catalog No. EHO 133-9, Institute of Electrical and Electronics Engineers, New York, 1978.
4. R. L. Melamud, J. D. Nihart, and J. M. White, "Dynamic Threshold Device," U. S. Patent 4,345,314, August 17, 1982.
5. Y. Yasuda, M. Dubois, and T. S. Huang, "Data Compression for Check Processing Machines," *Proc. IEEE* **68**, 7, 874–885 (July 1980).
6. K. Y. Wong, "Multi-function Auto Thresholding Algorithm," *IBM Tech. Disclosure Bull.* **21**, 7, 3001–3003 (December 1978).
7. R. E. Penny, "Dynamic Threshold Setting Circuit," *IBM Tech. Disclosure Bull.* **18**, 6, 1962–1965 (November 1975).
8. R. O. Duda and P. E. Hart, *Pattern Recognition and Scene Analysis*, John Wiley & Sons, Inc., New York, 1973, p. 271.
9. P. T. Cahill, R. J. R. Knowles, O. Tsen, T. Lowinger, and R. Pouapinya, "Evaluation of Edge Detection Algorithm Applied to Nuclear Medicine Images," *Proceedings of the Fifth International Conference on Pattern Recognition*, December 1980, pp. 1296–1300.

Received November 18, 1982; revised February 18, 1983

Gene D. Rohrer *IBM Information Products Division, 1001 W. T. Harris Boulevard, Charlotte, North Carolina 28257.* Mr. Rohrer is manager of the Advanced Technology Department at the

Charlotte laboratory. The department is currently engaged in projects for the banking industry. His previous experience includes the development of various recognition systems for IBM document processors as well as the advanced development of image systems. He also worked on the design and development of signal processing algorithms and control systems for a large variety of electromechanical devices. Mr. Rohrer received a B.S. in electrical engineering from North Carolina State University in 1967 and an M.S. in electrical engineering from Syracuse University in 1970. He is a member of Eta Kappa Nu, Phi Kappa Phi, and the Institute of Electrical and Electronics Engineers.

James M. White *IBM Information Products Division, 1001 W. T. Harris Boulevard, Charlotte, North Carolina 28257.* Dr. White is a member of the Advanced Technology Department in the Charlotte laboratory, where he is involved in the capture and use of conventional and electronic images. He was graduated with a B.S. in physics from the Georgia Institute of Technology in 1967, obtained an M.S. in applied physics from Stanford University in 1969, spent two years at the U. S. Army electronics laboratory in Fort Monmouth, New Jersey, and then returned to Stanford and completed his Ph.D. thesis in 1973. He started his professional career at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, where he spent two years working on integrated optics, then made major contributions in the design and testing of a CCD/photodiode scanner array, a document capture system, a gray-scale image processing algorithm, and the application of new printing technologies for images. In 1978 Dr. White transferred to the Charlotte location and is currently working on projects related to check reader/sorters. Dr. White is a member of Tau Beta Pi and Sigma Pi Sigma, and a senior member of the Institute of Electrical and Electronics Engineers.