R. G. Casey
C. R. Jih

A Processor-Based OCR System

A low-cost optical character recognition (OCR) system can be realized by means of a document scanner connected to a CPU through an interface. The interface performs elementary image processing functions, such as noise filtering and thresholding of the video image from the scanner. The processor receives a binary image of the document, formats the image into individual character patterns, and classifies the patterns one-by-one. A CPU implementation is highly flexible and avoids much of the development and manufacturing costs for special-purpose, parallel circuitry typically used in commercial OCR. A processor-based recognition system has been investigated for reading documents printed in fixed-pitch conventional type fonts, such as occur in routine office typing. Novel, efficient methods for tracking a print line, resolving it into individual character patterns, detecting underscores, and eliminating noise have been devised. A previously developed classification technique, based on decision trees, has been extended in order to improve reading accuracy in an environment of considerable character variation, including the possibility that documents in the same font style may be produced using quite different print technologies. The system has been tested on typical office documents, and also on artificial stress documents, obtained from a variety of typewriters.

Introduction

Optical character recognition (OCR) can be implemented at low cost by means of a document scanner connected to a microprocessor through an interface. The interface performs elementary image processing functions, such as thresholding of the video signal from the scanner. In addition, it synchronizes operation of the scanner and processor. The processor receives a binary image of the printed information, formats the image into individual character patterns, and classifies the patterns one-by-one.

Such an implementation avoids much of the development and manufacturing costs for special-purpose, parallel circuitry often used in conventional OCR systems [1]. It also permits inclusion of OCR in digital image systems that already have scanners and CPU's at low incremental cost. Certain office copiers and facsimile units fall into this category, for example.

This paper describes a CPU-based system for the reading of documents printed in conventional type fonts with uniform character spacing. Fixed-pitch printing is typical, for example, of office typing.

Recognition of conventional (as opposed to stylized) printing by a processor-based system requires the development of highly efficient segmentation and classification algorithms. The system described here segments a majority of characters from a scanned print line by means of measurements of pitch and baseline parameters. This information is also used as a reference for more complex segmentation methods that are applied to horizontally or vertically touching images. The overall approach is called a two-stage segmentation and registration scheme (TSSRS).

Classification is accomplished by inputting the character patterns to a number of decision trees designed for the particular font being read. Each decision tree is used to examine a subset of the character picture elements and to produce both a classification and a measure of the reliability of that classification. A pattern may either be identified immediately, if the overall confidence level is high, or else submitted to a second stage in which it is shifted locally, reclassified by the decision trees, and identified on the basis of a Bayesian decision algorithm using the various reliability measures furnished by the decision trees.

Recognition experiments were conducted using conventional typewritten documents as well as stress documents constructed to pose difficulties in segmentation.

Extension of the system into more complex environments is also possible.

System description

A typical OCR system carries out four processing steps, i.e., scanning, segmentation, registration, and recognition. The system contains corresponding compartments of hardware, as illustrated in Fig. 1, a block diagram of the entire OCR system. The blocks of this system perform the following functions:

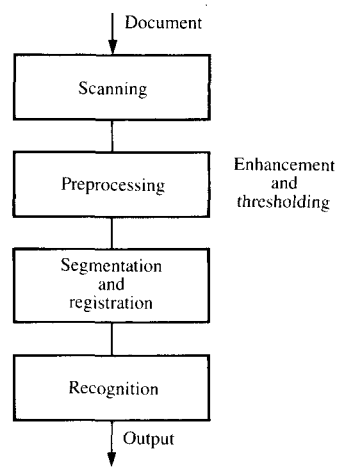

Figure 1 Components of an OCR system.

- *Scanning and preprocessing*

Sensor

An optical scanner is needed as the means for obtaining bit images of each character as input to the OCR system. The actual scanner device could be a self-scanned photodiode array, a "bucket brigade" device, a charge coupled device, or some other type of scanner. Also required are a source of illumination and an optical system to image the characters on the scanner.

Thresholding circuit

The output of the scanner is an analog signal corresponding to the amount of light reflected from each pixel on the source document. A thresholding circuit determines whether a pixel is to be considered black or white. A simple fixed threshold is adequate for high-quality documents on white paper, but a dynamic thresholding scheme is employed in most advanced OCR systems which is capable of generating reliable binary images despite variations in document print quality. Reliable and noise-free binary character images facilitate segmentation, registration, and recognition.

Buffer

Depending on the orientation and size of the scanner, some amount of image buffering is needed to store a number of character images during the segmentation, registration, and recognition steps. The buffer is typically a random access read-write memory.

- *Segmentation and registration logic*

Segmentation and registration are typically performed in combination. The associated logic finds the lines on the page, segments the character images (breaks the scanned image into separate, distinct images of each character), registers the character images (provides proper positional information about each image for use by the recognition logic), and supplies the segmented and registered images to the recognition logic.

- *Recognition logic*

Classification

After the character images have been properly segmented and registered, the recognition logic attempts to classify each image into one of the predefined symbol classes (e.g., an "A," "B," etc.). For high-quality images from machine-printed office documents, a recognition scheme based on a binary decision tree appears to provide acceptable performance. Details of this recognition scheme are described later in the paper. The recognition logic may identify an image as belonging to one of the character classes, or it may reject the image as unrecognizable.

Error recovery

Some means must be provided to handle cases where the recognition logic is unable to make a decision (a "reject error") or where the logic makes the wrong decision (a "substitution error").

Rejected patterns can be dealt with by allowing the operator to enter the correct information or by storing the entire character image with proper registration. Substitution errors can be corrected in a postprocessing step which tests characters in context.

Output to system

Once the characters have been properly classified, the information is ready for use in the system. For example, in a simple revision typing application, the recognized text may be output on a Selectric typewriter. If the document contains character classes other than those available on the particular Selectric type element being used for printing the majority of the classes, the system can stop printing and signal the operator to change type elements and can identify the proper type element number to use.

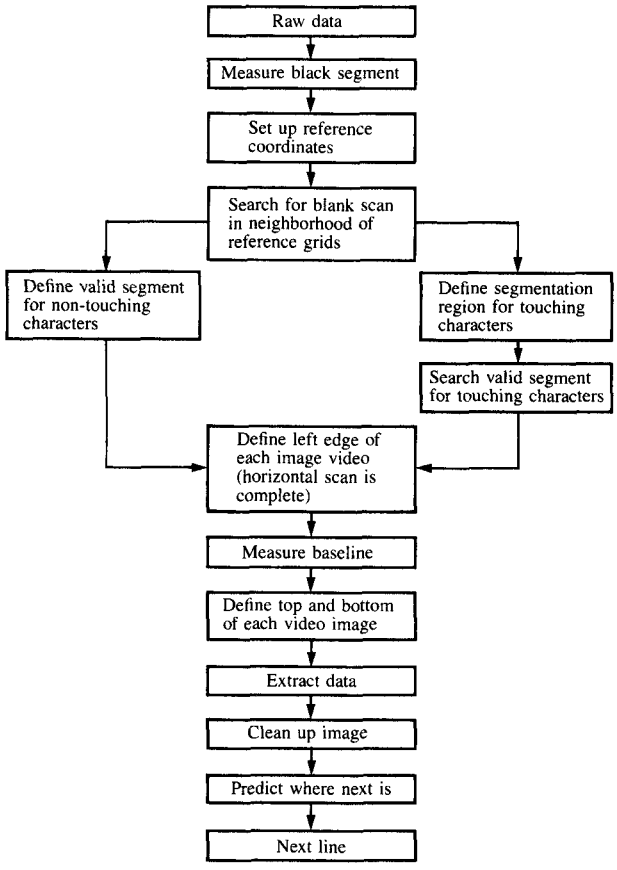

Figure 2 Flowchart of TSSRS algorithm.

Segmentation

Segmentation is a necessary and crucial step for OCR. Any error made in segmentation generally cannot be corrected in the following registration and recognition processes. In other words, segmentation errors directly affect the overall recognition performance.

In the past, some OCR systems have been designed to detect a line of characters by means of a special mark at the beginning of the line. A simple segmentation algorithm was then used with special recognition logic (as in the IBM 1287 optical reader). In some applications, overly wide patterns were simply rejected by the recognition logic so that no segmentation of touching characters was required (as in the IBM 3886 machine). More sophisticated OCR machines have been built to recognize omni-font, mixed-pitch documents with variable line spacing [2]. For this type of machine, a sophisticated line finding method is required before a segmentation algorithm can be applied.

In an office environment, most documents are currently machine-printed (e.g., typewritten) originals or electrographically generated copies containing a single font and a fixed character pitch. The documents usually have relatively good print quality, and the text is arranged within an inherent systematic grid pattern defined by the fixed pitch and line spacing.

Naturally, the systematic grid pattern is an advantageous feature for character segmentation and registration. However, it may be distorted due to misalignment of the printing mechanism or because of nonuniform magnification in a copying process. In addition to the regular segmentation problems of touching, broken, and skewed character images, the presence of underscores and subscripts or superscripts on office documents makes segmentation more difficult. If the recognition logic is sensitive to character registration, the segmentation becomes crucial and must be accurate. Under this circumstance, it is clear that an estimated fixed grid pattern is inadequate for the direct segmentation of a full page of characters; however, it can provide a useful reference. Moreover, for cost reasons, the character segmentation and registration for OCR office applications must be simple enough so that they can be implemented in a microprocessor or hard-wired logic with reasonable speed and accuracy. With these problems and requirements in mind, the study of the segmentation and registration of office documents becomes a challenging problem.

- *TSSRS segmentation method*

A two-stage segmentation and registration scheme (TSSRS) was developed for this application [3, 4]. Basically, the scheme involves use of the inherent document pitch and measured baseline information, defined in the following sections, in a flexible manner to perform the first-stage segmentation. (This takes care of all the nontouching characters.) Second, special segmentation routines are applied to the boundary region of touching images. The segmentation region is determined by the pitch and baseline. Effective and precise character segmentation and registration can be achieved using this two-stage scheme. The flow diagram of the TSSRS algorithm is shown in Fig. 2.

In this paper, horizontal segmentation means separation of the horizontal neighboring characters printed on one line. Similarly, vertical segmentation means the separation of a character from any images associated with the previous or the following print line. A column of pixels in the scanned image of a print line is called a vertical scan (i.e., Z-Z in Fig. 3). After the characters have been segmented and registered, each character image has a standard raster, a two-dimensional dot matrix. The size of the raster is chosen in accordance with the pitch and the scanning resolution. For example, in the experiments reported here, a character array 48 rows by 25 columns was used for 10-pitch documents scanned at 0.1-mm resolution.

Baseline and pitch estimation

In a given font design, each character has a specific vertical position relative to an imaginary horizontal line called the baseline. For example, the baseline of an uppercase X is the bottom horizontal "line" on which the character stands (Fig. 4). However, some characters, such as the lowercase j, have a descender below the baseline.

In standard printing or typing, the baselines of characters on one line are made to be collinear. Ignoring the occasional character with a descender, one can easily visualize a "line" which fits to the bottom of most characters on the same print line. Such a "line" actually is the averaged baseline for the entire sequence of characters.

In order to estimate the baseline, a series of horizontal density histograms representing consecutive scans of the same length across the scanned line of characters is generated. Each frame contains a plot of the number of picture elements received by the scanner as a function of character or line height. The peak in the lower portion of the histogram is used as the predicted location of the baseline. By averaging the histograms for the entire line, a baseline for the complete line can be predicted. The skew is given as the difference between baselines of consecutive frames on the same scan line.

Detailed baseline detection is initiated by means of a "search window" composed of a few picture elements adjacent to the expected location of the baseline. As the scanning proceeds, the position of the lowest black bit in the search window is recorded for each vertical scan. The average of the lowest black bit positions over a number of scans is considered to be the local baseline.

Once the baseline has been calculated, the print line is separated from neighboring lines using the known font dimensions. In addition, underscores can be detected, since their distance below the baseline is determined. A band of sufficient height to contain most of the image data for the line is defined along the baseline. The image data in this band are projected onto the baseline (i.e., the black pixels in each column are counted). The pitch is then estimated, after eliminating atypical segments, by calculating the average center-to-center distance of adjacent segments.

The scanning skew is the inclination of the measured baseline of the actual scanned image from an imaginary horizontal reference defined by the scanning mechanism. For example, in Fig. 3, the inclination of the measured baseline X-X with respect to the reference Y-Y is considered as the scanning skew. The baseline X-X can be approximately detected by the algorithm given above implemented in software. Based on the measurement, any systematic line

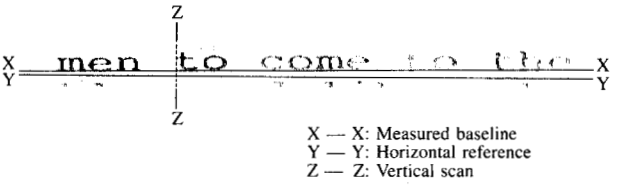

Figure 3 Sample scanned print line.

Figure 4 Character baseline.

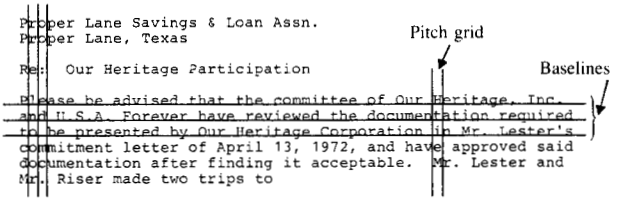

Figure 5 Document grid defined by pitch and baseline.

skew can be detected and then corrected by proper software preprocessing or by manual realignment of the document against some horizontal reference. Normally, such a systematic skew detection procedure may only be applied to the first line of a document. Afterwards, any minor line skew (possibly nonlinear) or vertical offset of a character will be detected and corrected by TSSRS.

Horizontal segmentation

Since there is an inherent grid pattern (Fig. 5) determined by the pitch and the baseline on a single-font, fixed-pitch, machine-printed document, the segmentation may seem to be simply accomplished by following the grid. This approach is called pitch segmentation.

In fact, the rigid grid pattern is not sufficient to provide adequate segmentation. The printing mechanism can easily produce both systematic and local positional errors. Further distortion may be caused by document aging and by copying

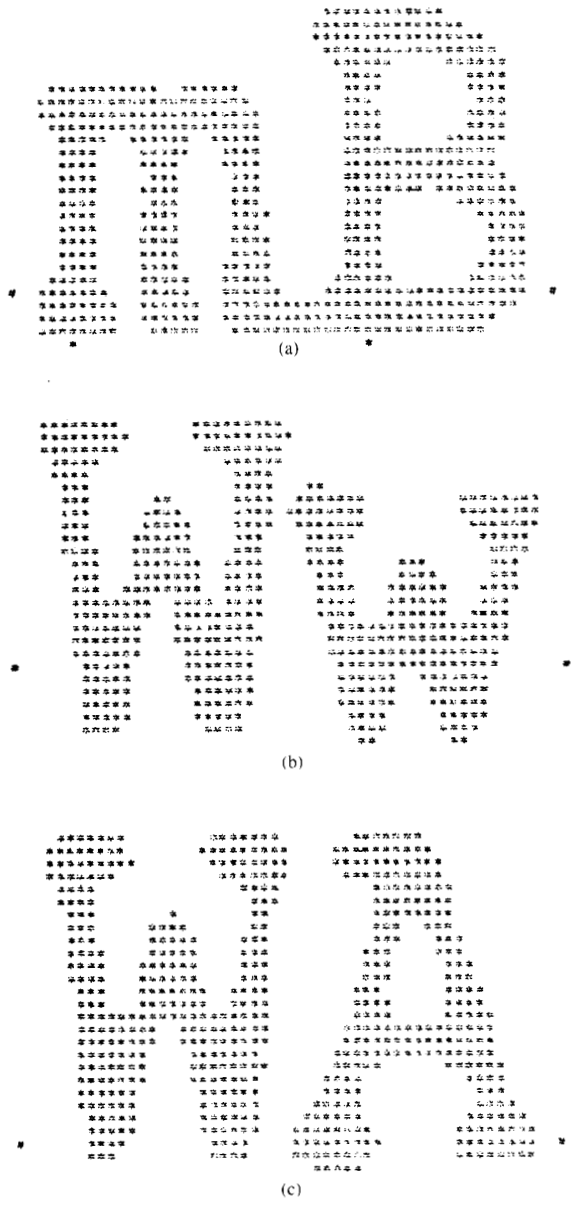

Figure 6 Problems in horizontal segmentation: (a) connected characters; (b) overhanging characters; and (c) zero overlap.

processes. Touching characters on a document may result from poor print quality or from characters which are wider than the standard pitch. At a typical resolution of 0.1 mm, pitch segmentation can result in errors of up to three or four pixels in positioning the character boundary. The classifier is then required to identify partial character patterns or pattern arrays containing portions of several characters. Such patterns have a much higher error rate than well-segmented characters. In the following paragraphs we discuss a method for adjusting the pattern boundaries obtained by grid estimation in order to accommodate local variations in positioning.

Touching character segmentation

A region in the image buffer consisting of three or four vertical scans on each side of a pitch grid line is analyzed by TSSRS in order to determine the best segmentation boundary. In an initial check, any blank scan found in this region determines a valid segmentation point. If no blank scan is found, then the two neighboring characters are considered to be touching. Three types of touching may exist in the segmentation region (see Fig. 6): (a) touching with connecting black strokes; (b) overhanging; and (c) barely touching (overhanging with zero overlap scan).

Type (a) contact is the one most often encountered when two adjacent wide characters touch at the serifs. Proper segmentation can often be carried out simply by finding the position where the pixel density of vertical scans is minimum. This is called the minimum density search method. For relationship (c), an AND of each vertical scan with the one to its left yields an all-zero column where the "touching" occurs. For type (b) (overhanging characters), there are several ways to do the segmentation. The one used by TSSRS is considered to be the simplest and most straightforward. First, the segmentation region is divided into several horizontal zones, each eight pixels in height (see Fig. 7). Within each zone, a set of contiguous vertical blank scans is sought. The leftmost and rightmost of these blank scans are registered as "L" and "R," respectively.

Sometimes only one such scan may be found. In this case, "L" and "R" overlap. If no blank scan exists in the subdivided zone, attempts to segment are abandoned and the pitch grid is used as a default boundary.

After determining all the L's and R's in the segmentation region, the rightmost L and the leftmost R are considered the valid segmentation points for the left and the right characters, respectively. Since the segmentation is made at the edge of a character, there is no registration problem, even if part of the image of an adjacent character lies in the character raster. An overhanging piece of image from a neighboring character can be masked out using boundary information obtained from the zones. If this method fails to find L and R for some zone, the reference grid position is used for segmentation.

The above method is more complicated than type (a) and type (c) segmentations. However, an overhanging case seldom occurs on Selectric-typed 10-pitch Courier 72 original documents. One may expect such touching and overhanging characters to occur more frequently on 12-pitch documents.

Vertical segmentation

Some OCR systems have been restricted to operate on a single print line, on multiple lines with wide line spacing, and

sometimes even with timing marks. In such cases the problem of vertical segmentation can be ignored. However, in office applications, the presence of lowercase characters, symbols, underscores, and subscripts or superscripts on a standard six-line-per-inch printed document creates the possibility that characters may touch the ones above or below, or on a single line lowercase characters with long descenders may touch underscores (Fig. 8). Vertical segmentation is required in order to keep a character image free of pattern components from adjacent lines or from its own underscore.

Baseline information is needed in order to assist in vertical segmentation and to define the proper image buffer for horizontal segmentation. Underscores can be detected and masked out from the segmented character image if the baseline position is known.

Besides providing line skew information, the current baseline location, together with knowledge of the line spacing, can be used to predict the location of the next line. The predicted baseline location is accurately updated by TSSRS based on the actual measurement. Through this adaptive procedure, any cumulative error in line spacing can be compensated. This is a very important feature for full page segmentation.

Broken characters or characters with legitimate separated components, like i, j, ;, etc., usually present some problems in segmentation. For omni-font application, in an uncontrolled environment, one has to detect the separate pieces (cells) and assure that they are noise-free before combining them into a character image. Fortunately, for single-font, fixed-pitch, machine-printed documents, the baseline and the reference grid coordinates, as used in TSSRS, form a net that more or less frames the character, so that no further recombination is required. Thus, TSSRS handles broken characters with little extra effort. This ability is even more important when one is dealing with electrophotographic copies of the original document, on which more broken characters emerge.

In practice, TSSRS segments at least 90% of the scanned character images directly based on the reference coordinates defined by the pitch and the baseline information. The remaining 10% are mostly touching characters. The reference coordinates are used to define the confined segmentation region so that the method described above can be applied locally.

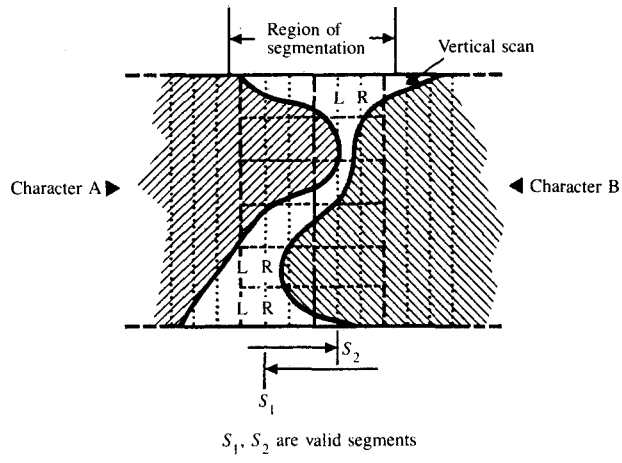

Figure 7 Segmentation of overhanging characters.

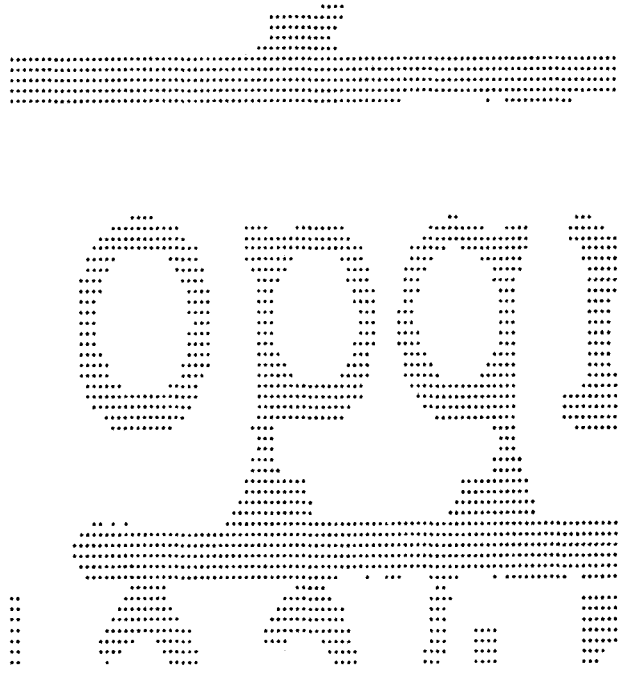

Figure 8 Vertically touching character patterns.

Decision tree classification

In a decision process a number of actions are performed in order to gain information needed to choose among alternatives. If the process is *sequential*, then only one action is performed at a time, and this action is selected on the basis of the results of previous actions. Classification of pattern shapes into a set of class identifiers, as done in OCR, is an example of a decision process. The typical commercial OCR machine is nonsequential, however, employing a number of subunits operating in parallel to extract information from each input pattern. The same set of actions is performed on every pattern, regardless of its shape. Some of the subunits may at times supply useless or redundant information, but this has no effect on performance as long as the ensemble provides sufficiently reliable classification.

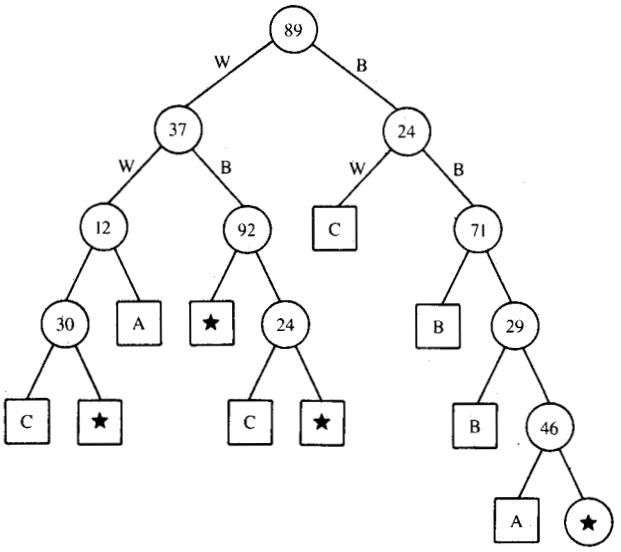

Figure 9 Decision tree. Interior nodes, shown as circles, indicate which pixel is being tested. Leaves are shown as squares, with the star indicating no decision (reject).

On the other hand, if classification is to be implemented by a processor, then operations that require time, but fail to assist in identifying an input, reduce throughput. A sequential mode, in which each action is selected to contribute maximal information to the overall process, becomes much more attractive.

It is such a sequential procedure that has been developed for office OCR. The elementary actions, or "tests," of which the process is composed consist of the examination of the colors of individual pixels. The procedure envisioned, therefore, is one in which a sequence of pixels is examined, with the location of each succeeding pixel determined on the basis of the colors observed in the pixels previously examined. The process terminates when enough pixels have been examined to permit reliable identification of the input pattern.

The procedure just described can be represented graphically by a binary decision tree (Fig. 9). It contains a unique starting node, the root, which has no incoming branches, and a set of terminal nodes, which have no outgoing branches. The remaining interior nodes each have a single incoming branch and two outgoing branches.

Each terminal node contains a class identifier, or ID, while each nonterminal node in the tree represents a pixel location. An outgoing branch is labeled with a white or black color value. The pixel specified by the root node is examined first, and the branch corresponding to its observed color is followed to arrive at a successor node, which specifies the next pixel to be examined, and so on. The path thus traced through the tree depends on the makeup of the input pattern. The path-following process ends when a terminal node is reached, whereupon the input is assigned the ID found at that node.

The tree is structured and the node parameters assigned so that only pixels useful for identifying the pattern in question are evaluated as a path is traced from the root node to a terminal node. Although the total tree specifies many pixels, and is therefore capable of recognizing a variety of inputs, only a small subset of these pixels need be examined in order to identify a particular pattern.

In principle, a tree that sequentially examines the elements of the character array can be made to recognize with accuracy as great as that achievable by any classifier. However, a truly optimal decision tree may require an immense amount of storage. The techniques developed in this investigation are directed toward obtaining high performance with only moderate storage requirements, as discussed in the next section.

A previous paper described a procedure for the design of a decision tree on the character pixels using a probabilistic model of the pattern variations within each character class [5]. The method seeks to maximize the information gained by a tree of specified size. By means of this technique the designer can produce a classifier to occupy a prespecified amount of storage. The basic tree design capability is taken as a starting point in the following discussion of classification using a number of decision trees.

Multiple decision trees

Although in principle a decision tree can be constructed to realize any desired classification logic, in practice the implementation of a sufficiently accurate tree may call for an excessive amount of storage space. Figure 10 shows curves of estimated and actual error rate versus tree size in an experiment where pixel statistics were obtained from scanned typewritten characters. Note that the curves level off after a few hundred nodes. For example, increasing the tree size from 850 to 1800 nodes did not decrease the experimental error rate at all. The reason for this behavior is the exponential growth property of such trees. The exact rate of increase in path length (i.e., the number of pixels examined before a decision is made) as the tree grows larger depends on the degree of imbalance of the tree. However, in experiments, doubling tree size resulted in an increase in average path length in the tree of slightly more than a single pixel. Improving a large tree by appending pixels to each path is therefore very costly.

An alternative approach to error rate reduction is to design more than one tree and to combine decisions obtained from

all of them. Empirically it was found that three 1000-node trees yield an average total path length of 30–40 pixels, but occupy no more storage than a single tree averaging 12–15 pixels per path. Since its classification is based on an examination of more pixels, a multiple-tree scheme offers hope for improving the trade-off of accuracy versus storage, at the expense of additional time spent examining pixels.

Two problems must be dealt with in order to classify using multiple trees:

1. The tree design procedure must be modified to assure that each tree yields independent information about the identity of the input pattern.
2. A rule must be devised to specify classifier response when more than one decision result is available.

A simple extension has been made in order to achieve the objective of tree independence. The tree design program is adjusted to select a different root node pixel for each tree produced. The probability distribution of the character classes for the two successor nodes to the root varies with the choice of the root pixel. Since the information value of the pixels considered for assignment to the successor nodes is a function of this probability distribution, the new root node produces a different choice for the successor pixels. This effect propagates to lower levels of the tree as well. Although the method does not guarantee that each tree will examine a completely different set of pixels in response to a given input pattern, in tests the frequency of repeated pixels has been less than 25%. Figure 11 shows the pixels examined for the character 4 in sample trees designed by the above method. In general, we observe that not only are different pixels being tested, but also different areas of the pattern. The union of the pixels examined constitutes a broad sampling of the pattern area.

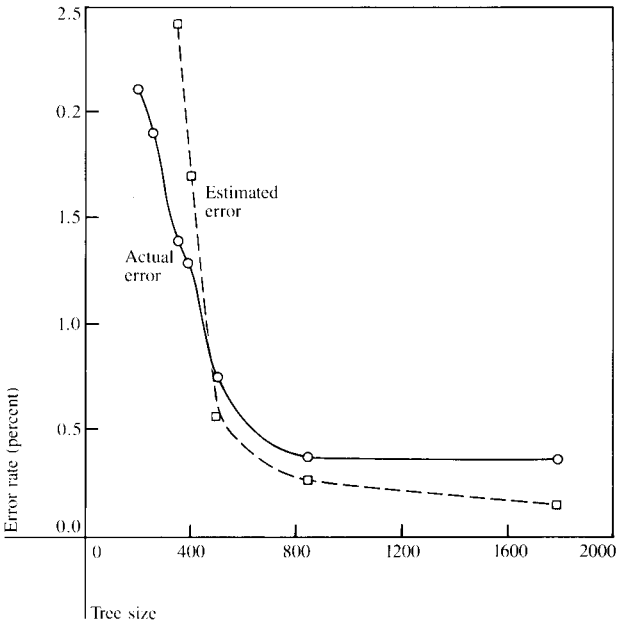

Figure 10 Actual and estimated error rate as a function of the number of nodes in the tree. The estimated error for each of seven trees is computed as part of the design process on the basis of the pixel probabilities in the design sample. New samples were used to determine the actual classification error for each tree.

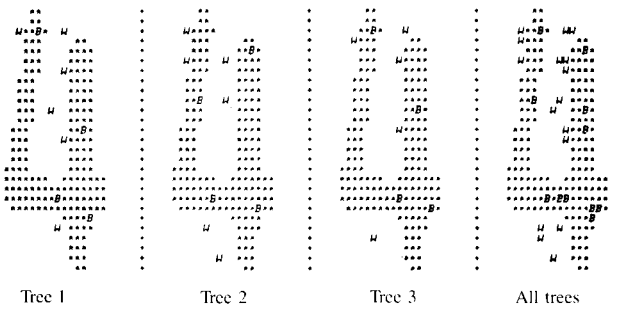

Figure 11 Pixels examined in recognizing a sample character, where B indicates that a black pixel was observed and W a white one.

• *First-stage classification*

When several trees are used for recognition of an input pattern, each tree provides both a decision as to the identity of the input and an estimate of the probability of error for that decision. This information is used in an initial attempt to classify the pattern. If each decision tree yields the same ID, and if each estimated error rate is below a prespecified threshold t, then the input character is labeled with the proposed ID. Otherwise, no decision is made at this stage, and identification of the input character is deferred to another stage of classification, as described below. Thus, any pattern that gives rise to conflicting ID codes, or that yields too high an error estimate, is deferred.

Parameter t, the threshold on estimated error probability, can be used to control the rate of acceptances versus deferrals. Using typewritten characters as input to three decision trees and a 0.5% threshold value, the rate of deferrals has ranged from 2% to 10%. This procedure yielded an extremely low error rate in the more than 90% of the input that is identified at this stage. Indeed, in a recognition test using over 200 000 inputs (to be described later), no substitution errors were made at this stage. The unanimous-vote, low-error-estimate test is a screening device that reliably classifies easily identified patterns, while passing difficult samples along to a more complex classification procedure.

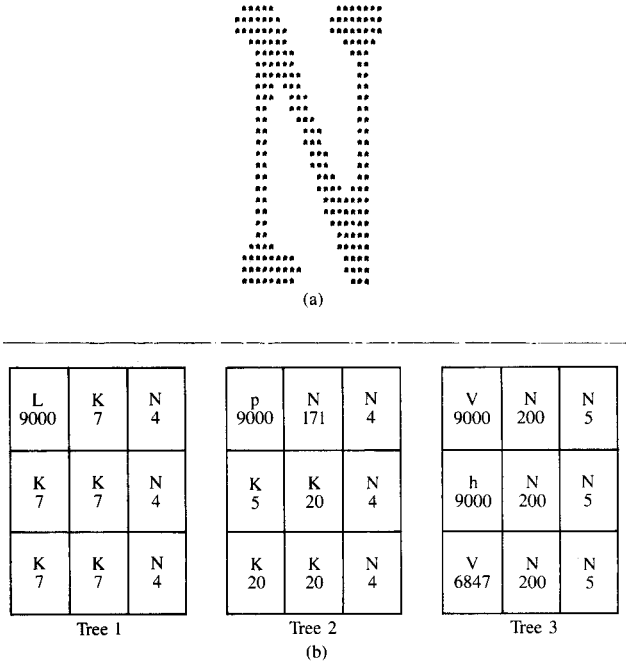

Figure 12 Multiple-tree decisions. (a) Input pattern. This letter is rather thinner than characters used for design, which results not only in a different pixel configuration, but also in a slight misregistration. (b) Decisions over shifts of the input. The center element in each matrix contains the decision for the unshifted input. Other decisions are obtained by shifting the input one pixel in the direction indicated by location from the center. The number beneath each decision is the estimated probability of error for the respective decision, multiplied by 10 000.

In experiments with multiple classifiers we have consistently used three decision trees. It is left to future investigations to establish the trade-offs among the number of trees, the speed of operation, and the recognition accuracy. If more decision trees were used, then the unanimous vote rule would need to be modified, perhaps along the lines of the Bayesian procedure to be described later.

• *Second-stage shift-and-retry*

Figure 12(a) shows a pattern that failed the unanimous vote requirement in a test using three decision trees. If the pattern is translated by a small amount and reclassified by the same trees, the resulting decisions convey additional information about the identity of the input character. Figure 12(b) shows the results when this shift-and-retry is done over the eight possible one-pixel translations of the pattern from its initial box-registered position. With the inclusion of the classifier outputs for the unshifted pattern, there are 27 different decision results, which are arranged as a collection of 3 × 3 matrices in Fig. 12(b). The estimated error probability, indicating the "confidence" associated with a decision, is also exhibited.

The figure shows that when the pattern is shifted by small amounts and reclassified, a number of correct decisions is produced by each tree. This happens because inputs differ slightly from design samples in registration position or in local pixel configuration. A decision rule can be implemented to assess the various outcomes and to identify the input by means of a weighted voting scheme.

A statistical decision rule using multiple classifications is derived in the Appendix. It is assumed that the collection of elementary tree decisions is mutually independent and that the estimated error probabilities are accurate representations of the true error probabilities. A further simplification in the rule is obtained by assuming that when an incorrect ID is produced by a tree, it is equally likely to belong to any of the symbol classes. Without this last assumption it would be necessary to store a large matrix of confusion probabilities for each tree. We have experimented with intermediate schemes, such as storing the most likely confusion ID along with the first choice ID at each leaf node of a decision tree; however, the resulting performance improvement does not seem to justify the extra cost in storage and CPU time.

The Bayesian decision rule is one that minimizes the probability of classification error under the assumptions made. It is implemented by computing a score for each ID that occurs in the list of tree decisions. The score for the Nth ID is

$$S_N = \sum_{j \in I_1(N)} \log [1 - P_e(j)] + \sum_{j \in I_2(N)} [\log P_e(j) + K],$$

where

$I_1(N)$ is the index set for decisions favoring the Nth ID,
$I_2(N)$ is the index set for decisions favoring a different ID,
$P_e(j)$ is the estimated error probability for the jth decision, and
K is the log of the probability that the jth decision, if incorrect, actually was in response to a pattern belonging to the Nth class.

The quantity K, according to our assumptions, should have the value $[1/N_c - 1]$, where N_c is the total number of symbols in the alphabet to be recognized. In practice, however, K is treated as a parameter. Its effect is to assign the log of a fixed portion of the jth error probability to the score for class N if the jth decision is unequal to N. For example, if $K = 0$, then the entire error probability for the jth decision is assigned to the score for class N. In experiments, the probability values were normalized to one million, and the logarithms were computed to the base two, so that each term in the score calculation is a number less than 20. Further-

more, the values of log P_e and log $(1 - P_e)$ were rounded to the nearest integer and stored as a table. Then it is necessary merely to place in each leaf of the decision tree an index to the appropriate row of the table.

The scores S_n are used either to identify the input or to reject it. The input is rejected if the two largest scores differ by an amount less than a prespecified reject tolerance, D. Otherwise the ID having the largest score is chosen. The reject tolerance provides the system with a degree of control over the substitution rate. Increasing D, i.e., raising the separation required between the highest score and the nearest contending score, increases confidence in the decisions that are actually made, at the cost of rejecting additional inputs.

The decision rule described above has been effective on good-quality inputs. However, where some of the patterns input to the classifier have been mutilated by the segmenter, by poor printing, or by other causes, it can err by assigning IDs to such patterns instead of rejecting them, as would be desirable. The errors are due in part to the assumption of independence among the decisions made by a given tree as the input pattern is shifted. An erroneous decision will frequently be repeated, and so a mutilated pattern can give rise to a set of scores that clearly favor one ID.

In order to help reject mutilated characters without affecting the recognition of undeformed characters, a single condition has been added to the scoring rule just described. An ID that would be output under the preceding rule is rejected by the classifier if one or more of the decision trees fails to list the ID among its outcomes.

The error rate improvement offered by the multiple-tree, two-stage scheme is illustrated by an experiment (see Fig. 13). Using a well-segmented character set, the recognition accuracy of a single decision tree of 3000 nodes was compared with that of three trees, each containing 1000 nodes. The reject criterion for the single tree is a threshold test on the estimated error probability. For any given rejection rate, the two-stage procedure using multiple trees yielded error rates lower by a factor of 5 or more, although both classifiers require the same total storage (3000 nodes each). About 3% of the inputs were deferred by the multiple-tree method, and since each deferred input is subjected to 24 additional tree decisions, the time required for the multiple-tree test is slightly more than three times the time required for classification by the single large tree.

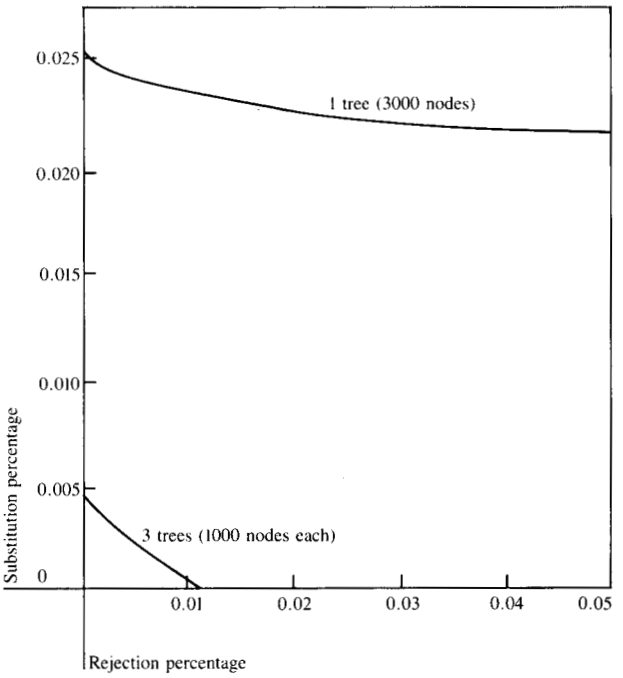

Figure 13 Multiple-tree vs single-tree recognition. In the recognition experiment, 55 000 typewritten samples from five typewriters were tested. Characters were well spaced to eliminate segmentation errors. Each classifier was allotted the same amount of storage. (Storage is proportional to the number of nodes in the tree.)

Experiments

• *Input documents*

The techniques described for segmenting and classifying printed characters have been tested on documents comprising more than 250 000 typewritten characters in Courier 10-pitch font. The documents (Fig. 14) were printed by ten different IBM Selectric typewriters from master magnetic cards, in order to avoid typographical errors. In some cases new print elements were provided, but where a type ball having the required symbol set was already in operation, it was used for the experiment. New carbon film ribbons were provided, although in one case a different ribbon was inadvertently used, yielding patterns of somewhat inferior quality in comparison with the other typewriters.

The documents may be categorized into three groups. One set contained repetitive sequences of characters separated by spaces, assuring ideal segmentation after scanning. These "alphabets," as printed by five of the typewriters (randomly chosen from among the ten), provided approximately 500 samples of each symbol for use in designing the decision tree classifier. Since "period" and "comma" are repeated on the 88-symbol Courier element, and since "underscore" is recognized during the segmentation process, the total symbol set consisted of 85 character types.

The remaining two document groups were used for test purposes. One group represents typical office correspondence and memoranda, whereas the other consists of character

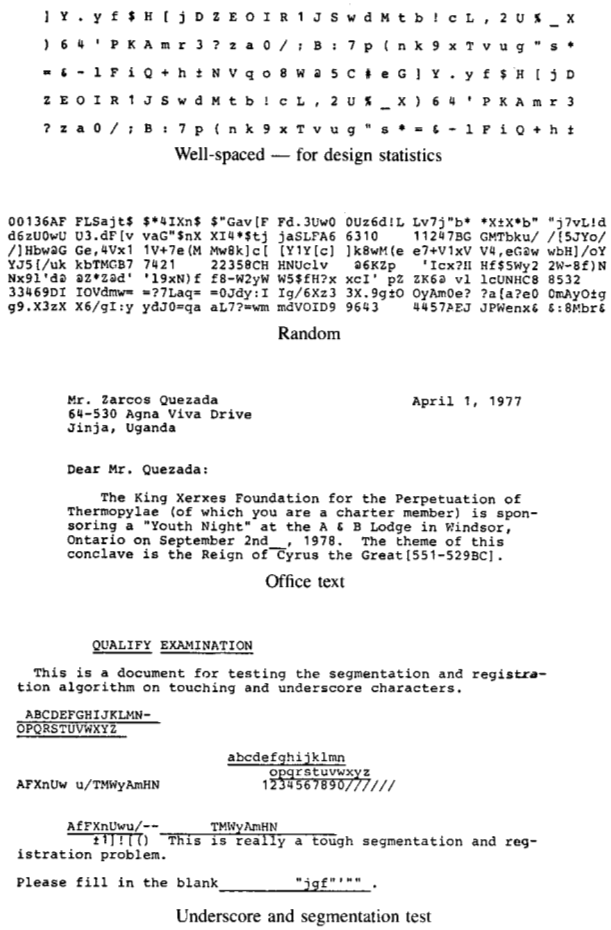

Figure 14 Sample documents for recognition tests.

sequences containing all possible adjacent character pairs, plus special stress documents containing heavy underscoring and difficult segmentation combinations. Thus, the recognition results may be used to estimate performance on typical office documents, as well as to detect special problem cases for the recognition system.

- *Scanning of the documents*

All documents were scanned at 0.1-mm resolution (250 pixels per inch) and stored on magnetic tape after thresholding. The scanner was a 64-pixel column of silicon diode sensor elements, swept horizontally across the document. It was programmed to scan one print line, then step down one-sixth of an inch (the standard line spacing at 10-pitch) to scan the next line. This open-ended operation caused the print lines to creep up in the field of view as the document was traversed. In practice, a feedback scanning mode would be used to avoid this. The baseline position estimated during the process of segmenting a print line would be employed in order to set the scanner parameters to scan the following line. The lack of such correction was not a serious problem in this experiment, however. Its effects were minimized by careful manual alignment of the first print line on each document scanned. Nevertheless, baseline skews of up to 20 pixels from start to end of a print line were observed.

As scanned onto magnetic tape, each print line occupied a record of approximately 10 000 bytes. These records, read off the tape one at a time, constituted the input to the segmentation and classification routines.

- *Segmentation and registration*

Individual characters were segmented by the TSSRS method. The 64-pixel-high window containing a text line also frequently held portions of the lines above and below. In a number of cases there was touching contact between two lines, and there was frequent overlap between adjacent characters within a line. Nevertheless, segmentation was generally successful and no patterns were rejected as unsegmentable, although several of the eventual recognition errors were attributable to missegmentation.

Each character pattern was registered so that its bounding rectangle was centered in the standard 48 × 25 array presented to the classifier.

- *Decision tree design*

Figure 15 illustrates the pixel statistics gathered from 500 samples of each character. As is typical of printing in a single font, many pixels are either always black or always white for the entire sample of a given symbol. However, these pixels are considered during design as "highly reliable" rather than "perfectly reliable," in order to assure that a number of redundant pixels are examined before the pattern is identified [5]. This is achieved by estimating the probability that a given pixel will be black at a future occurrence of the symbol as $(B + 1)/(N + 2)$, where N is the number of samples (500 in this case) and B is the number of samples for which the pixel was black.

Three decision trees were designed for the Courier font, each containing 2000 nodes. The choice of a tree size of 2000 nodes was made as a trade-off among implementational and performance considerations. A tree node requires four bytes of data in a typical processor memory. The first two bytes contain the coordinates of the pixel to be examined. The second byte pair holds the address of the next node to be tested. This address plus either two or four bytes is entered if the pixel is black. A flag bit in the second byte pair determines whether the skip is two or four bytes. A decision node is indicated by a flag bit in the first two bytes, and requires only two bytes. The first byte contains the flag plus a 7-bit entry into an error probability table. The second byte

contains an 8-bit ID. Thus, a tree of N test nodes is stored in $6N + 2$ bytes. The three 2000-node trees designed for the present experiment require 36 kilobytes of storage with this implementation.

Classification

Segmented patterns were input in ordinary reading sequence to the three-tree classifier. The reject parameters, t and D, for this experiment were ones which had given good results in past work with typewritten inputs. The unanimous vote requirement at the first stage resulted in the identification of about 98% of the characters. No substitution errors were incurred in this portion of the classifier. The remaining 2% of the segmented patterns were classified in the second stage (local-shift-and-retry), where a character was rejected if its Bayes score was an insufficient amount higher than that of the next best candidate ID. All reject and substitution errors occurred in this stage.

The results are tabulated in Tables 1 and 2. Table 1 shows a breakdown by typewriter. It is here that the effect of a nonfilm ribbon appears. The corresponding typewriter contributes almost one-half of the total errors. However, the overall rate of one reject in 21 000 characters and one substitution per 42 000 characters is still in the acceptable range for such a system. The breakdown of the performance data by document group [Table 2] shows even better recognition accuracy for those documents representative of correspondence typewriting. The error rates on this part of the data, if sustained in practice, indicate that on average 25 documents containing 2000 characters apiece could be read without error before a reject had to be identified manually. Fifty such documents would be read before a substitution error occurred.

The nonfilm ribbon does not appear to have caused difficulty on the typical text class of data. Most errors occurred among the pseudorandom character strings. Underscores were not a problem in detection nor in their effects upon neighboring characters. Sample error patterns are shown in Fig. 16.

Conclusion

The automatic recognition of typewritten text using decision tree classifiers has been described. The text is assumed to be printed at a fixed pitch; however, it may be in a conventional font style and may include underscoring.

A segmentation method that locally adjusts segmentation points predicted from measurement of character pitch has been shown to work well for typewritten inputs. The method copes with both touching lines and touching characters, and it recognizes underscores during operation.

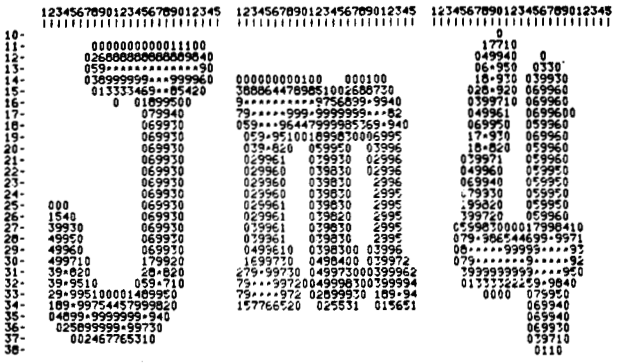

Figure 15 Pixel statistics. Each digit represents the relative frequency of black values for the corresponding pixel. Actual percentages obtained from 500 samples of each character were divided by 10, then rounded down. A blank denotes that no black was observed, an asterisk, no white.

Table 1 Recognition errors listed by typewriter.

Typewriter	*No. of documents*	*No. of characters*	*Errors*	
			Rejections	*Substitutions*
1	13	20,845	2	—
2	"	"	—	—
3	"	"	—	—
4	"	"	—	1
5	"	"	—	—
6	"	"	—	—
7	"	"	3	4
8	"	"	2	—
9	"	"	1	—
10	"	"	2	—
Totals	130	208,450	10	5

Table 2 Recognition errors listed by document type.

Document type	*No. of documents*	*No. of characters*	*Errors*	
			Rejections	*Substitutions*
Random character	30	88,770	7	3
Office text	80	113,460	2	1
Underscore	20	6,220	1	1

The decision trees are applied in a two-stage scheme that permits easily identified patterns to be quickly classified on the basis of a single pass through each tree. Patterns not

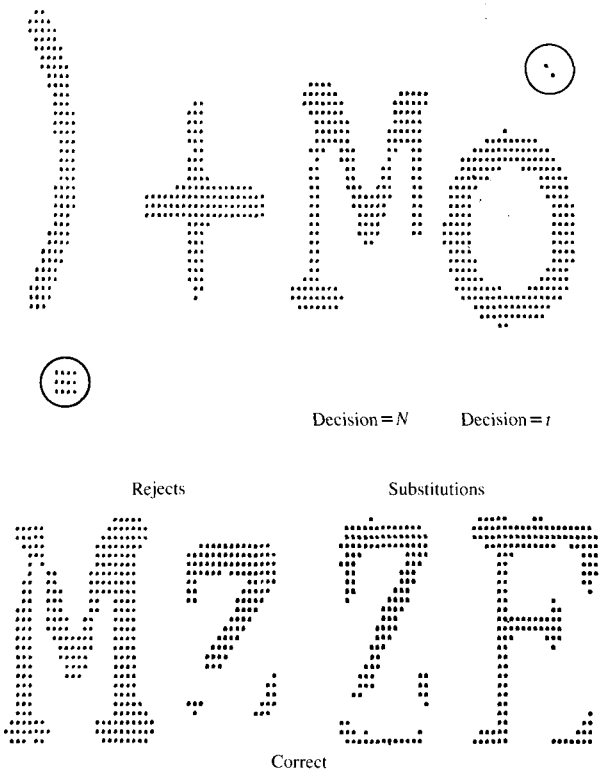

Figure 16 Sample recognized characters. The first choice decision for the two rejected characters shown was actually correct, but the patterns were rejected on the basis of low difference between scores for first and second choices. Noise pixel groupings that caused incorrect registration are circled.

classified with assurance by the initial stage are shifted to a number of neighboring positions and reclassified by the decision trees. A weighted voting scheme makes the final identification.

The decision trees are designed automatically using statistics gathered from identified sample character patterns. The design method permits specification of the number of decision trees to be used and the size of the trees. Typically, a few hundred examples of each character are scanned for tree design. However, as few samples as one per character have been used with surprisingly good results.

Reading accuracy of the segmentation and classification methods in combination has been tested using conventional office typing as well as special stress documents constructed to make recognition more difficult. Error rates on the conventional text were two rejections and one substitution per 50 000 inputs, in an experiment involving ten typewriters.

The OCR system has been developed with the objective of implementing it by a processor (for example, a microprocessor). Estimates of the reading rates obtainable with commercial processors range up to several hundred characters per second. Since the system may be implemented in any computer having an attached scanner, and since its design requires only a moderate amount of data collection and is largely automatic, the method appears highly suited either to personalized applications or to low-cost recognition of fixed-pitch fonts.

Acknowledgment

The authors owe a debt of gratitude to G. Farmer of IBM, who initiated the project and gave it fervent support. Kwan Wong of IBM's Research Division managed the project in its later stages and is providing a testing ground for it in his Document Analysis System, currently under development. Technical assistance was provided at various stages by R. N. Ascher, R. Findlay, and J. Kellner. T. Cassada of IBM Lexington, Kentucky, tested some of the methods reported here in his own work and supplied useful feedback and ideas. Finally, G. Nagy, of the University of Nebraska, helped to formulate the original decision tree approach to classification.

Appendix: Bayesian weighting of multiple tree decisions

Let $X = \{X_1, X_2, \cdots, X_m\}$ be m tree decision outcomes, where $X_j = (c_j, e_j)$, c_j is a class label, and e_j is the probability that this decision is incorrect. Let $C_1, C_2, \cdots, C_n$ be the set of possible class labels. We assume that the classes are equiprobable *a priori* and that the individual tree outcomes are statistically independent.

By Bayes' Rule,

$$Pr(C_N|X) = \frac{\prod_j^m Pr(X_j|C_N)}{\sum_t^n \prod_j^m Pr(X_j|C_t)}. \quad (1)$$

But

$$Pr(X_j|C_t) = \begin{cases} 1 - e_j & \text{if } C_t = c_j, \\ e_j Pr(c_j|C_t) & \text{if } C_t \neq c_j. \end{cases}$$

Letting S_N be the logarithm of this expression, we obtain

$$S_N = \sum_{J_1(N)} \log (1 - e_j) + \sum_{J_2(N)} [\log e_j + \log Pr(c_j|C_N)] + \text{constant}, \quad (2)$$

where

$J_1(N) =$ indexes j such that $c_j = C_N$,

$J_2(N) =$ indexes j such that $c_j \neq C_N$,

and the constant indicated represents the logarithm of the denominator of (1), which does not depend on N.

S_N represents the likelihood that a sample belonging to the Nth class was input. In practice, only the value of N that maximizes this quantity is sought; i.e., only relative values of S_N are of interest, and so the constant may be dropped from (2).

The probabilities e_j are estimated during design of the decision tree, and so are immediately available. The quantity $Pr(c_j | C_N)$ represents the probability that the tree chooses class j when the input actually belongs to class N. This quantity may be estimated either from the known distribution of class probabilities at the terminal nodes or else by testing the tree on identified samples. A third alternative is simply to divide the error probability among the contending classes, which corresponds to an assumption of both equal *a priori* probabilities and a uniform distribution of classes at the terminal node when an error occurs. In this case $Pr(c_j | C_N) = 1/(N - 1)$. As an expedient, log $Pr(c_j | C_N)$ may be replaced by an arbitrary constant K, which is then chosen experimentally to maximize the recognition rate. This replacement yields the formula given in the text.

References

1. G. Balm, "An Introduction to Optical Character Reader Considerations," *Pattern Recognition* **2,** 151 (1970).
2. R. B. Hennis et al., "The IBM 1975 Optical Page Reader," (three papers) *IBM J. Res. Develop.* **12,** 346 (1968).
3. C. R. Jih, "Segmentation Method for Fixed Pitch, Machine Printed Documents," *IBM Tech. Disclosure Bull.* **23,** 1194 (August 1980).
4. C. R. Jih, "Optical Character Recognition Using Baseline Information," U. S. Patent No. 4,251,799, February 17, 1981.
5. R. G. Casey and G. Nagy, "Decision Tree Design Using a Probabilistic Model," *IEEE Trans. Info. Theory* **IT-29,** (1983, in press).

Received December 15, 1982; revised February 24, 1983

Richard G. Casey *IBM Research Division, 5600 Cottle Road, San Jose, California 95193.* Dr. Casey has worked primarily in pattern recognition, particularly optical character recognition, since joining IBM in 1963. Until 1970, he carried out projects in this area at the Thomas J. Watson Research Center in Yorktown Heights, New York. After transferring to San Jose, he worked initially on the analysis of data bases, but returned to the character recognition area in 1975, soon after completing a one-year assignment for IBM in the United Kingdom. Dr. Casey received a B.E.E. from Manhattan College in 1954 and an M.S. and Eng.Sc.D. from Columbia University, New York, in 1958 and 1965. From 1957 to 1958, he was a teaching assistant at Columbia University, and from 1958 to 1963 he investigated radar data processing techniques at the Columbia University Electronics Research Laboratories. While on leave of absence from IBM in 1969, he taught at the University of Florida.

Chentung Robert Jih *IBM General Products Division, 5600 Cottle Road, San Jose, California 95193.* Dr. Jih is a development engineer, responsible for developing advanced suspension technologies for future DASD files. In 1974, he joined IBM in the Advanced Technology Department of the Office Products Division in San Jose as a staff engineer, developing acoustic scanners. From 1976 to 1979, he was in charge of OCR technology development for office applications, working jointly with the IBM Research Division, San Jose. In 1980 and 1981, Dr. Jih was engaged in evaluating and developing speech recognition technologies for office system applications with the IBM Thomas J. Watson Research Center in Yorktown Heights, New York. In 1981, he transferred to his present position in the General Products Division, San Jose. Dr. Jih received a B.S. in mechanical engineering from the National Taiwan University in 1968 and an M.S. and a Ph.D. in mechanical engineering from the University of California at Berkeley in 1971 and 1974, respectively. He is a member of Sigma Xi.