OPTICAL CHARACTER RECOGNITION

Character recognition is the process by which bitmaps within desired fields (regions) of a document are converted into char-

acter codes (e.g., ASCII, SGML, or Unicode). In other words, it is the process of mapping a visual image of iconic representation of text into a symbolic representation. Character recognition based on scanned images is also known as *optical character recognition* (OCR), where the word "optical" arose historically from the fact that the earliest approaches used a set of optical masks for recognition (1) as opposed to magnetic-based approaches. In modern approaches to character recognition, optical processing is no longer used except for scanning documents, and recognition is typically implemented by software running on digital computers or specialized chips. Because of the increased sophistication of today's recognition algorithms and the range of input that they can handle, such as handwritten data and omnifont machine print, the industry trend is to use the term *intelligent character recognition* (ICR).

Based on the means which produce the document, OCR problems are often grouped into the following types: (1) machine print; (2) dot-matrix print; (3) constrained or discrete hand print, where characters are printed either in boxes provided or with separation by sufficiently wide gaps; (4) unconstrained hand print, where neighboring characters may touch each other; (5) cursive script, where characters are produced continuously, without taking the pen off the paper, by means of ligatures which connect adjacent characters; and (6) mixed hand print and cursive script. The level of difficulty in recognizing these types generally increases in that order, although it can be further complicated by the quality of scanned images. Examples of these types of OCR problems are shown in Fig. 1.

Considerable variability of machine-print fonts and handwritten styles make OCR a very challenging problem, which has attracted more than 45 years of research (1,2). The major sources of variability include:

1. There are about 100 commonly used fonts for machine-print text. A letter in a particular font can have a very similar shape to a different letter in another font. Moreover, serifs often join adjacent characters, which causes segmentation difficulties. Horizontal and vertical spacings can be irregular due to subscripts, superscripts, proportional spacing, and kerning.
2. The size of characters varies a lot for both machine-print and handwritten characters.
3. Text to be recognized maybe accompanied by underlines, highlights, annotations, and graphics.

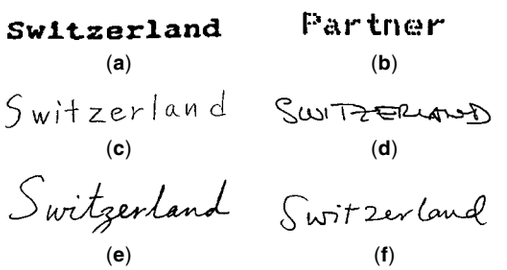

Figure 1. Types of OCR problems: (a) machine print, (b) dot-matrix print, (c) constrained hand print, (d) unconstrained hand print, (e) cursive script, (f) mixed hand print and cursive script.

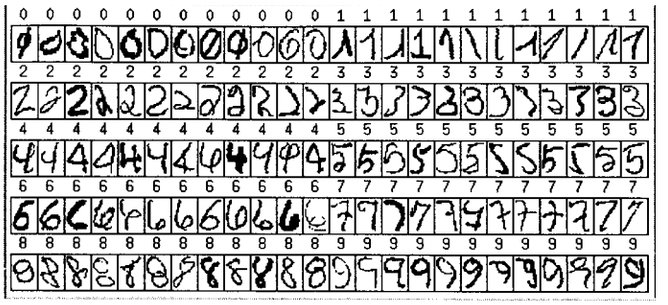

Figure 2. A sample set of handwritten digits. The size of each digit is normalized to a fixed size of 24 (height) by 16 (width) pixels. Each bitmap is labeled by its truth label above the bitmap.

4. Document generation and scanning either add noise or remove valid pixels, which may result in noisy, smudged, or broken strokes. They may also introduce skew and nonlinear distortion into character bitmaps.
5. A large variety of writing instruments are in use, which produce character strokes with different thickness, darkness, and uniformity.
6. Handwritten text is often slanted in various degrees.
7. There are significant variations in writing styles among people of different nationalities, regions of origin, ages, levels of education, professions, and their physiological and emotional states. Even for the same writer, the style changes with the writing instrument, paper (especially forms of different types and sizes), and emotional state.

Figure 2 shows a sample set of size-normalized digit bitmaps, where significant shape variations can be observed.

With the prevalence of fast computers, computer networks, and large and inexpensive memory and storage, there is an increasing trend towards online access and processing of a large variety of information that is traditionally available on paper media, such as books, journals, and business forms. Since manual key entry of information on these documents is labor-intensive and expensive because of the huge volumes, the need for automatic data capture and processing from paper documents has fostered increasing interest in document image analysis, whose objective is to recognize the text and graphics in the scanned-in document images and then convert them into computer-understandable form (e.g., ASCII, SGML, or Unicode).

OCR technology has been widely used to solve a large variety of real-world problems:

1. *Forms Readers for Tax Forms, Census Forms, and Various Application Forms (3).* The task of a forms reader is to automatically extract and recognize user-filled-in data from paper forms. The whole process involves scanning the forms, recognizing form types, and compressing them for subsequent archiving and OCR. An automatic tax forms reader can read about 75% of the forms (3). In such a case, information on the remaining ones will have to be entered with the assistance of an image of the document displayed on the screen and appropriate databases or dictionaries. IBM, Lockheed-

Martin, and Caere are three of the leading providers of forms-processing solutions.

2. *Postal Mail Sorting (4,5).* Reading postal addresses, sorting mail, and delivering mail are labor-intensive tasks for postal services throughout the world because of the huge mail volumes. Automation is increasingly employed in this task by scanning the envelopes or parcels, locating the destination-address block, reading the address using OCR, interpreting the address, and imprinting a bar code for the delivery point. The mail pieces are automatically sorted using the delivery-point code in the walk sequence of the postal carrier. This solution eliminates (or reduces) the need for manual operation until the point of delivery. For US addresses, the five-digit zip code can be obtained from the zip code field in the address, which can be verified by city–state name recognition. If the additional four or six digits are not present, they have to be inferred from the street address. Leading providers of postal mail sorting solutions are IBM, Siemens, Lockheed-Martin, Alcatel, and Elsag.
3. *Bank Check Reading (6,7).* Checks are one of the most popular instruments of financial transactions. The annual check volumes that US banks alone process exceed 55 billion per year. Such high volumes of documents are processed using high-speed reader/sorters that make use of magnetic ink character recognition (MICR). Preprinted codelines on checks carry the bank name, account, and check sequence numbers in E13B magnetic ink. Typically, the bank of first deposit completes the codeline by determining the payment amount and inscribing this datum on the check using E13B characters. Traditionally, this has been done by manually inspecting the amount written in numbers, called the convenience or courtesy amount (CA), and key-entering the data. The amount written in words, called the legal amount (LA), is inspected only when the courtesy amount is not legible. Because of the large volumes, this operation is labor-intensive and expensive. Check imaging systems are increasingly used by banks to minimize the labor costs by implementing automated data entry with the help of OCR. Good CA recognition is critical for this application. It requires about 50% read rate with less than 0.5% error rate to justify the cost of the system. While this is easy to achieve on commercial checks, which have machine-printed CAs, it is a harder target for personal checks, which usually have handprinted CAs. One effective way of reducing the error rate is to use an alternate source for the data. Deposit slips and adding machine tapes that accompany the checks provide a second source for CA recognition. Yet another source of redundant information is the LA. By comparing the results of the LA and CA recognition, the error rate can be minimized. In fact, LA recognition is an ideal application for cursive OCR as the number of possible words in the lexicon is very small, of the order of 30 entries ("one," "two," . . . "ten," "eleven," . . . "twenty," "thirty," . . . "hundred," "thousand," "million," etc.). Products offering CA and LA recognition are available from companies such as a2i, and Parascript. IBM, Unisys, and NCR are the leading providers of commercial check-processing systems.
4. *Desktop OCR (8–11).* Desktop OCR products are used by end users for converting various documents of daily use into coded form. For example, users may want to scan magazine articles, recipes, letters, etc., and import them into a word processor for later search, retrieval, and update. Since these are low-volume applications, the expectations of accuracy (read and error rates) are high. On clean machine-printed documents, common desktop OCR products can achieve better than 99% accuracy on a per character basis. The accuracy degrades to the range of 94% to 98% on noisier documents. Spelling checkers to quickly detect errors and correct them are often an integral part of desktop OCR. The leading desktop OCR products are Caere OmniPage, Xerox Text Bridge, Expervision TypeReader, and Ocron. They all support some amount of automatic segmentation of text areas from drawings, sketches, images, and the like in the document. They also support retention of multiple columns of text and the ability to import the OCR results into word-processing software. Typically, they cost less than $100.

OCR SYSTEMS

Figure 3 shows the basic architecture of a typical OCR system. The input and output to each module in the system are also indicated. A paper document is first scanned by an optical scanner or a camera to obtain a digital (gray-scale or binary) image of the document. Scanning resolution of 80 pixels/cm (200 dots/in.) is adequate for recognition purposes. Desktop scanners tend to use a resolution of 120 pixels/cm. This scanned image undergoes a set of preprocessing operations such as noise filtering, binarization, skew correction, layout analysis, and field extraction (extracting the fields of interest). Different sets of preprocessing operators may be required for different applications. Then, for each field of interest, the field bitmap is processed by the segmentation module to produce a list of character bitmaps (see the following section). Next, the recognition module generates a list of hypotheses of character identities for each character bitmap (see the section "Recognition of Segmented Characters"). The charac-

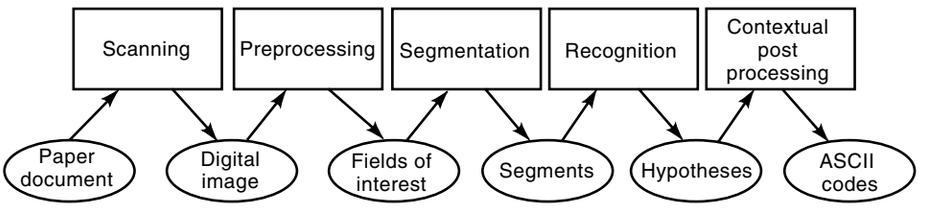

Figure 3. The basic architecture of a typical OCR system.

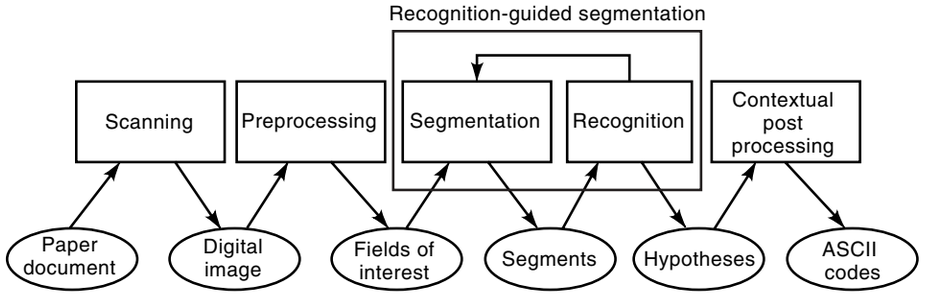

Figure 4. The architecture of an OCR system that employs a recognition-guided segmentation scheme.

ter hypotheses obtained from this step are subject to errors due to wrong segmentation, noise, and ambiguous writing styles, as well as character classification errors. The contextual postprocessing module is designed to handle these OCR errors (see the section "Contextual Postprocessing"). It recovers some of these OCR errors by exploiting certain syntactic and semantic constraints as well as redundant information existing in many OCR applications such as form processing, check reading, and postal address recognition.

Note that there is no interaction between the segmentation module and recognition module in this basic architecture. This design may work reasonably well on good-quality machine-print, dot-matrix-print, and constrained hand-print text where most characters are separated from their neighbors. If a document contains many touching characters, this architecture will make lots of segmentation errors due to a large degree of inherent ambiguity in identifying character boundaries.

It is well known that human readers rely heavily on recognition in identifying character boundaries. This suggests a recognition-guided segmentation scheme (see the section "Recognition-Guided Segmentation"). In this scheme, the final segmentation points are determined by the quality of recognition results on the individual segments. The architecture of an OCR system that employs such a scheme is depicted in Fig. 4.

In both the architectures in Figs. 3 and 4, the contextual processing module is a postprocessor that does not have any interaction with the segmentation and recognition modules. This limits its ability to resolve many segmentation and recognition errors. For difficult OCR problems such as recognizing cursive script, the multiple hypotheses returned by the segmentation and recognition modules often do not contain the correct answer. Quite often, these hypotheses make no sense at all. A more sophisticated design attempts to bring contextual information into the early stage of segmentation and recognition, rather than to use it for postprocessing. Figure 5 shows such an architecture where contextual information is integrated into the segmentation and recognition modules.

The recognition-guided segmentation scheme and the more advanced segmentation–recognition scheme are normally more computationally intensive than simple segmentation-followed-by-recognition scheme. Therefore, a particular OCR system may employ all three architectures and adaptively determine which one to use, based on the type and quality of the input image.

SEGMENTATION–DISSECTION TECHNIQUES

Segmentation refers to the process of determining character boundaries for a given image of a line of text or a word. A number of approaches have been proposed in the literature. An excellent survey of strategies in character segmentation is provided in Ref. 12, where various segmentation schemes are grouped into three categories: (1) dissection techniques, (2) recognition-based segmentation, and (3) oversegmentation-based techniques. (The "holistic" category in Ref. 12 does not deal with segmentation.) Techniques in the second and third

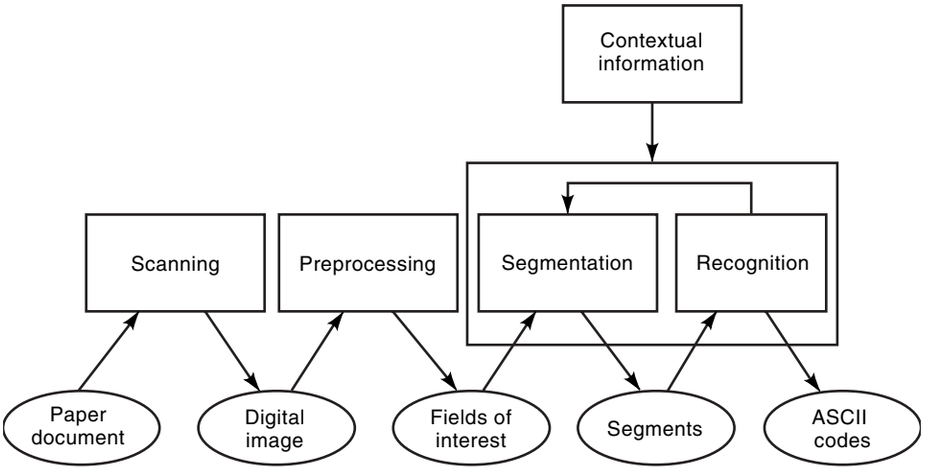

Figure 5. The context-guided architecture where contextual information is integrated into the early stages of segmentation and recognition.

categories belong to the recognition-guided segmentation scheme and will be discussed in the section on that subject.

Dissection techniques attempt to decompose the image into classifiable units on the basis of general image features. These techniques include white-space and pitch analysis, vertical projection analysis, connected-component analysis, and landmark analysis.

The *vertical projection* of a line of text consists of a count of black pixels in each column. By analyzing the vertical-projection profile, one can identify character boundaries. Consecutive zero counts in the profile indicate a vertical gap between two adjacent characters. A local nonzero valley may imply a possible slightly touching or overlap of two adjacent characters. However, simple projection may give a false impression of the continuity of the image. For example, if one pattern terminates in column c while its neighbor starts in column $c + 1$, the projection is the same irrespective of the relative vertical positions. Such cases are quite common in machine printing. The end of one character may intrude upon the space of its neighbor with only light contact or none at all. *Kerning* of adjacent characters (e.g., "Ta") and italic text give the same effect. Such "shadow" effects can be reduced by an overhang-reduction process before applying a projection rule. This process creates a new image in which each column is the logical AND of three neighboring columns in the original image. This process can produce local valleys at the shadowed positions in the vertical projection. It can also make the existing valleys deeper and thus easier to detect.

Connected-component analysis is also a popular method for segmentation. A *connected component* is a region of black pixels that are connected together, that is, one can walk, pixel by pixel in eight or four directions, from any black pixel to any other black pixel in the connected component without going through a white pixel. Connected components in an image can be efficiently computed using a standard two-pass algorithm. Further processing may be necessary to combine or split these connected components into character images. For noncursive characters, the distribution of bounding boxes of connected components and their positions provides a fairly reliable clue to decide what connected components to merge or split.

When a connected component is elected to be split, a clue is typically available to detect the point of contact of the two characters. The clue may be described as a concavity located at the contact point and existing on at least one side of the penetrating stroke. This suggests that when a pattern of merged characters is suspected, the segmenter should seek such a concavity in the most likely part of the image, viz., the middle section of the pattern [see Fig. 10(a, b)]. Contour following is commonly used in such a search (13,4). A set of potential split points are located in portions of the outer contour that are locally concave and have nonvertical direction. The points are searched in order of an estimate of local curvature, and a split point (or cut) is identified if it is within a specified distance of a point on the opposite side of the same contour. This guarantees that a break of the pattern between these two points will divide it into two subpatterns (single contact point).

Dissection techniques are suitable for segmenting good-quality machine-print and constrained hand-print text where characters touch only occasionally and lightly. But they often fail on unconstrained hand-print and cursive script.

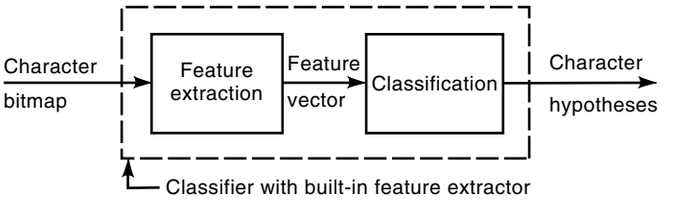

Figure 6. Basic modules for recognizing segmented characters. Most OCR systems have two separate modules for feature extraction and classification, while others employ classifiers with built-in feature extraction capability.

RECOGNITION OF SEGMENTED CHARACTERS

Recognition is the process of assigning a symbolic label to the input character bitmap. Figure 6 shows a diagram of a typical recognizer for isolated characters. It consists of a feature extractor and a classifier. The input to the recognizer is a character bitmap, and the recognizer returns a list of hypotheses on the character identity. Most classifiers also associate a confidence value (or probability) to each hypothesis. These hypotheses can later be used to guide segmentation of touching characters or to resolve OCR errors by contextual postprocessing.

Note that it is not always necessary to have a separate feature extraction module in an OCR system. Some classifiers, such as feedforward neural networks, have built-in feature extraction capabilities. Figure 7 illustrates these two dif-

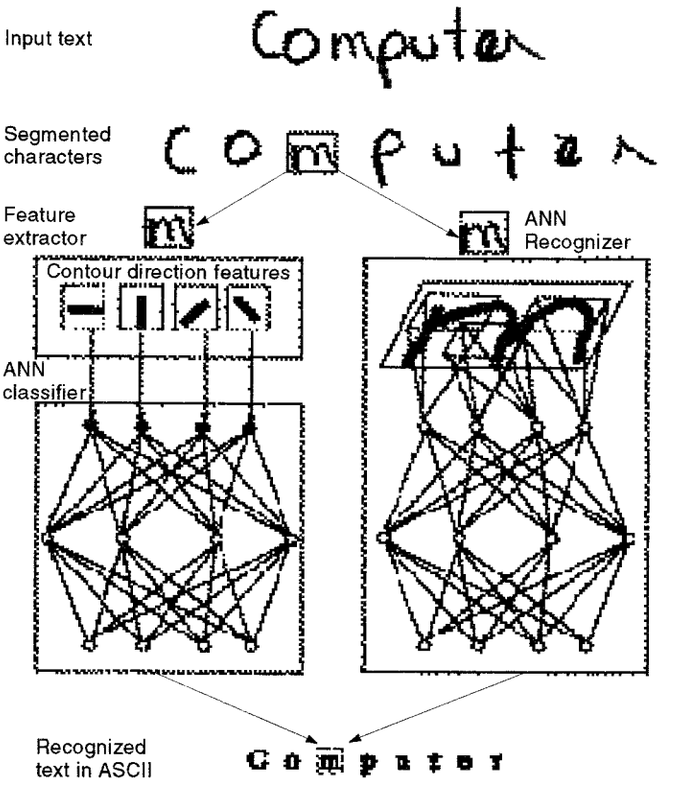

Figure 7. Two schemes for recognizing isolated (segmented) characters: A recognizer with an explicit feature extractor (left) and a recognizer with a built-in feature extractor (right).

ferent schemes. An example of such systems is LeCun's network (14) for digit recognition. A 16 pixel by 16 pixel normalized gray-level image is presented to the feedforward network with three hidden layers (not identical to the network shown in Fig. 7). The units in the first hidden layer are locally connected to the units in the input layer, forming a set of local feature maps. The second hidden layer is constructed in a similar way to the first hidden layer. Each unit in the second hidden layer also combines local information coming from feature maps in the first hidden layer.

The goal of feature extraction is to extract the most relevant (to classification) measurements (features) from the character bitmaps, so as to minimize the within-category variability while increasing the between-category variability. Feature extraction often leads to dimensionality reduction, which helps to avoid the well-known "curse of dimensionality" in pattern recognition. But unlike data compression, good features need not be those with small reconstruction error, as long as they can discriminate between classes.

The feature extractor also specifies the complexity of the classifier following it. If features are good in the sense that in the feature space, patterns from different classes are well separated while patterns from the same class are highly concentrated, a simple classifier such as a linear classifier or the nearest-mean classifier would be sufficient. Unfortunately, such good features are often very difficult to obtain. On the other hand, if features are at a very low level, then a classifier capable of forming complex decision boundaries must be used. A tradeoff between the complexity of the feature extractor and classifier must be considered in designing an OCR system.

As we can see from Fig. 2, there are considerable variations in character shapes for every single category. A recognizer that is able to map all these variations of a single category into a single label must achieve a high degree of invariance with respect to these variations, including scaling, skew (or rotation), slant, stroke thickness, nonlinear deformation, noise, and shape variations due to writing styles. The invariance property can normally be achieved through the following four different ways:

1. *Preprocessing.* Scaling invariance is fairly easy to achieve by size normalization, which scales the original bitmap up or down to a fixed size, say 24 pixels in height and 16 pixels in width. However, care must be taken to decide when to preserve the character aspect ratio of vertically or horizontally elongated characters (e.g., "I," "l,", "-") and relative position and size information of punctuation. Many OCR systems employ a skew and slant detection and correction algorithm. However, features that are invariant to small-angle skew and slant are still desirable, because no skew correction and slant correction algorithm can guarantee perfect deskewed and deslanted character bitmaps. Invariance to noise is usually achieved by noise filtering and contour smoothing. Invariance to nonlinear distortion and different writing styles is very difficult to achieve by any preprocessing methods.
2. *Using Invariant Features.* Features invariant to translation, scaling, and rotation can be derived from geometric moments, Zernike moments, and Fourier coefficients (15). Some other features, such as lakes and bays, are invariant to scaling, skew, and slight local deformations, but are very sensitive to writing styles and noise. It is very difficult to obtain features that are truly invariant to all the variations.
3. *Training the Classifier with a Large Data Set That Covers as Much Variability as Possible.* Since class boundaries in the feature space can be very complex due to the high variability, a classifier that is able to form complex decision boundaries should be used. A large feedforward network and the k-nearest-neighbor classifier (see the subsection "Character Classification") are the two popular choices. In order to avoid overfitting to noise, mislabelings, and variations in the training data, a large training data set must be collected. The data collection and manual truthing of collected data can be very labor-intensive. Several character degradation models (16) can be used to artificially generate more training data from existing training data by adding noise to the stroke contours, nonlinearly deforming the bitmaps, and randomly slanting and skewing the bitmaps according to certain distributions.
4. *Using Invariant Recognition Methods.* Examples are nearest-neighbor classifiers using tangent distance (17) and deformable template matching (18). These approaches only achieve invariance to small linear transformations and nonlinear deformations. Besides, they are computationally very intensive.

No single approach is able to achieve perfect invariance. A combination of these four approaches in various degrees is often used in designing OCR systems.

Feature Extraction

A large number of feature extraction schemes have been proposed in the literature (15). Commonly used character features can be grouped into the following categories: (1) topological features such as loops, junctions, stroke end points, bending (high-curvature) points, lakes, bays, and Euler number; (2) moment features, such as geometric moments and Zernike moments; (3) transform-based features such as the Fourier transform and the Karhunen–Loeve transform (or principal-component analysis); (4) projection-based features; and (5) local or zoned contour direction features and multiresolution gradient features.

As an example, we now briefly describe a scheme for encoding the bending-point features. Bending-point features represent some topological characteristics of a character, such as high-curvature points, terminal points, and fork points. Strong curvatures, labeled as bending points, are detected in the character image by tracing the contour of strokes. A special geometrical mapping from bending points and their attributes (e.g., acuteness, position, orientation, and convexity or concavity) to a fixed-length (96) feature vector has been designed. The size-normalized image is evenly divided into 12 (4×3) regions. The bending points in the normalized image are coded by their positions specified by corresponding region indices and by their curvature orientations, which are quantized to eight cases (four orientations, each of which is either convex or concave). The acuteness of a bending point is used as the magnitude for the corresponding component in the fea-

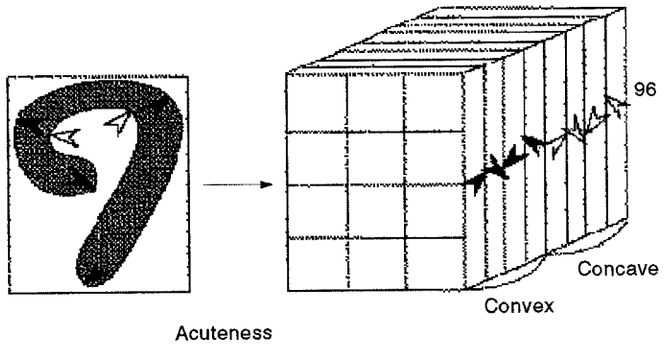

Figure 8. Bending-point features.

ture vector. An example of bending points and this mapping scheme is shown in Fig. 8.

Character Classification

The task of pattern classification is to assign symbolic names (or labels) to observations (feature vectors) of patterns. In the context of OCR, character classification assigns an ASCII code (in general, a symbolic name) to the feature vector extracted from the raw character bitmap.

A large number of classifiers have been used for character classification. These classifiers were developed in different communities such as statistical pattern recognition, structural pattern recognition, neural networks, machine learning, and fuzzy set theory. Although most of these classifiers are applicable to character classification, two predominant approaches are feedforward networks (function approximation in general) and k-nearest-neighbor classifiers. This can be seen from the test results at the First Census Optical Character Recognition System Conference held in 1992 (19), where more than 40 different handwritten-character recognition systems were evaluated based on their performance on a common database. The top ten performers among them used either some type of multilayer feedforward network or a nearest-neighbor-based classifier.

In the next two subsubsections, we briefly introduce these two types of classifiers.

Feedforward Neural Networks. Figure 9 shows a typical three-layer perceptron. In general, a standard L-layer feedforward network (we adopt the convention that the input nodes are not counted as a layer) consists of one input stage, $L - 1$ hidden layers, and one output layer of units, which are successively connected (fully or locally) in a feedforward fashion with no connections between units in the same layer and no feedback connections between layers. For the character classification, each input unit receives a character feature, and each output unit represents a character category. Given an input feature vector, the network is first evaluated. Then the class corresponding to the largest output is selected as the character hypothesis. Multiple hypotheses can also be formed by selecting the largest few outputs.

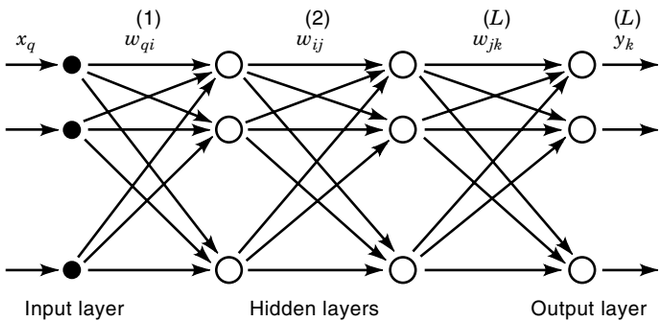

Figure 9. A typical three-layer feedforward network architecture.

The most popular class of multilayer feedforward networks is *multilayer perceptrons,* in which each computational unit employs either the thresholding function or the sigmoid function. Multilayer perceptrons are capable of forming arbitrarily complex decision boundaries and can represent any Boolean function (20). The development of the *backpropagation* learning algorithm for determining weights in a multilayer perceptron has made these networks the most popular among researchers as well as users of neural networks.

We denote by $w_{ij}^{(l)}$ the weight on the connection between the ith unit in layer $l - 1$ and the jth unit in layer l.

Let $\{(\boldsymbol{x}^{(1)}, \boldsymbol{d}^{(1)}), (\boldsymbol{x}^{(2)}, \boldsymbol{d}^{(2)}), \ldots, (\boldsymbol{x}^{(p)}, \boldsymbol{d}^{(p)})\}$ be a set of p training patterns (input–output pairs), where $\boldsymbol{x}^{(i)} \in R^n$ is the input vector in the n-dimensional pattern space, and $\boldsymbol{d}^{(i)} \in [0, 1]^m$, an m-dimensional hypercube. For classification purposes, m is the number of classes. The squared-error cost function, which is most frequently used in the neural network literature, is defined as

$$E = \frac{1}{2} \sum_{i=1}^{p} \|\boldsymbol{y}^{(i)} - \boldsymbol{d}^{(i)}\|^2 \tag{1}$$

where $\boldsymbol{y}^{(i)}$ is the output vector of the network given the input pattern $\boldsymbol{x}^{(i)}$.

The backpropagation algorithm (21) is a gradient-descent method to minimize the squared-error cost function in Eq. (1).

A special class of multilayer feedforward networks is the *radial basis function* (RBF) network (22), a two-layer network. Each unit in the hidden layer employs a radial basis function, such as a Gaussian kernel, as the activation function. The radial basis function (or kernel function) is centered at the point specified by the weight vector associated with the unit. Both the positions and the widths of these kernels must be learned from the training patterns. The number of kernels in the RBF network is usually much less than the number of training patterns. Each output unit implements a linear combination of these radial basis functions. From the point of view of function approximation, the hidden units provide a set of functions that constitute an arbitrary *basis* for representing input patterns in the space spanned by the hidden units.

There are a variety of learning algorithms for the RBF network (22). The basic algorithm employs a two-step learning strategy (hybrid learning): estimation of kernel positions and kernel widths using some unsupervised clustering algorithm, followed by a supervised least-mean-square algorithm to determine the connection weights to the output layer. Since the output units are linear, a noniterative (closed-form solution) algorithm can be used. After this initial solution is obtained, a supervised gradient-based algorithm can be used to refine the network parameters.

This hybrid learning algorithm for training the RBF network converges much faster than the backpropagation algorithm for training multilayer perceptrons. However, for many problems, the RBF network often needs a larger number of hidden units than a multilayer perceptron. This implies that the run-time (after training) speed of the RBF network is often slower than the run-time speed of a multilayer perceptron. The efficiencies (error versus network size) of the RBF network and the multilayer perceptron are, however, problem-dependent. It has been shown that the RBF network has the same asymptotic approximation power as a multilayer perceptron.

Another special case of feedforward networks is the polynomial classifier (23), which consists of two layers. The hidden layer computes the polynomial combinations of input features, and the output layer computes a linear combination of these polynomials. As a result, each output unit implements a polynomial function of input features. As in the RBF networks, the weights associated with the output layer can be determined by matrix inversion. The polynomial classifier has been successfully used in character classification (23).

***k*-Nearest-Neighbor Classifiers.** The k-nearest-neighbor (k-NN) classifier (24,25) is a well-known and commonly used classifier in statistical pattern recognition. It is a nonparametric approach, which makes no assumption about the underlying pattern distributions.

Let n training pattern vectors be denoted as $\boldsymbol{x}^{(i)}(l)$, $i = 1, 2, \ldots, n_l$, $l = 1, 2, \ldots, C$, where n_l is the number of training patterns from class ω_l, $\sum_{l=1}^{C} n_l = n$, and C is the total number of categories. The k-NN classifier examines the k nearest neighbors of a test pattern $\boldsymbol{x}$ and classifies it to the pattern class most heavily represented among the k neighbors. In mathematical terms, let $K_l(k, n)$ be the number of patterns from class ω_l among the k nearest neighbors of pattern $\boldsymbol{x}$. The nearest neighbors are computed from the n training patterns. The k-NN decision rule $\delta(\boldsymbol{x})$ is defined as

$$\delta(\boldsymbol{x}) = \omega_j \qquad \text{if} \quad K_j(k, n) \geq K_i(k, n) \quad \text{for all } i \neq j$$

The Euclidean distance metric is commonly used to calculate the k nearest neighbors. Other metrics (26), such as the optimal global nearest-neighbor metric, can also be used.

A severe drawback of the k-NN classifier is that it requires a large amount of computation time and memory. The standard k-NN classifier stores all the training patterns. In order to find the k nearest neighbors of an input pattern, we have to compute its distances to all the stored training patterns. Because of this computational burden, the k-NN classifier is not very popular where real-time requirements have to be met. Many modifications to the k-NN classifier have been proposed in the literature. For example, one can eliminate a large number of "redundant" training patterns using the condensed nearest-neighbor (CNN) rule, the reduced nearest-neighbor (RNN) rule, or the edited nearest-neighbor (ENN) rule (26). Alternatively, one can use efficient algorithms for computing the nearest neighbors, such as the branch-and-bound algorithm (25).

Feedforward networks tend to be superior in speed and memory requirements to nearest-neighbor methods. Unlike the nearest-neighbor methods, the classification speed using feedforward networks is independent of the size of the training set.

RECOGNITION-GUIDED SEGMENTATION

When a document contains many touching characters, the dissection techniques discussed in the section "Segmentation–Dissection Techniques" often make a number of segmentation errors due to a large degree of inherent ambiguity in identifying character boundaries. Recognition-guided segmentation attempts to use character recognition to improve the reliability of segmentation. It includes the recognition-based segmentation strategy (12) and the oversegmentation-based strategy (12).

Recognition-based segmentation avoids dissection and segments the image either explicitly, by classification of specified windows, or implicitly, by classification of subsets of spatial features collected from the image as a whole. Examples in this category include the hidden Markov model (HMM) with a sliding window (27). The oversegmentation-based scheme is a combination of the dissection and recognition-based segmentation strategies, employing dissection to segment an image into so-called graphemes (primitives) and then employing dynamic programming to recombine graphemes into admissible segments based on recognition results. Oversegmentation-based HMMs can also be built (28). In general, oversegmentation followed by dynamic programming does not require probabilistic modeling. Heuristics can be easily incorporated into the recognition. On the other hand, word-level optimization is difficult to perform without any probabilistic modeling.

To illustrate how segmentation works in detail, we now focus on the oversegmentation-based scheme. This scheme consists of the following phases: (1) identifying potential split points, (2) constructing a graph, and (3) finding the shortest path in the graph from the leftmost node to the rightmost node.

Identifying Potential Split Points

This phase attempts to identify a set of split points that is a superset of the ideal split points. The idea of oversegmentation is not to miss any character boundaries, even at the cost of including false alarms.

The dissection techniques discussed in the section "Segmentation–Dissection Techniques" can be used to oversegment machine-print and hand-print text. However, for cursive script a different scheme must be employed. This is because in cursive script, adjacent characters are joined by a ligature, which is normally very smooth [i.e., no point with a high curvature value can be found; see Fig. 10(c, d)]. Most of pure cursive script can be cut into graphemes at the valley points

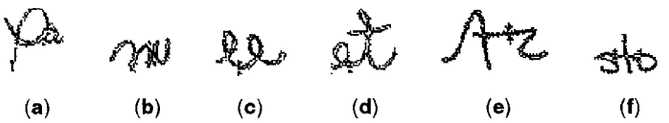

Figure 10. The three main features on the upper contour used to detect cuts. The relevant cuts are denoted by an arrow in the upper contour and a corresponding arrow in the lower contour: (a, b) high-curvature points; (c, d) smooth valleys; (e, f) horizontal stretches.

of the ligatures. These points, together with the high-curvature points and the middle points of horizontal stretches [see Fig. 10(e, f)], can cover approximately 98% of true split points in mixed hand-print and cursive script.

Finally, some cuts are dropped, using various criteria: (1) If two cuts are close by, then the longer cut is dropped. (2) If the cuts cross each other, then also the longer cut is dropped. (3) The longer cut of two adjacent cuts that produce a very small segment is also removed. (4) A cut is dropped if the resulting two segments have too small vertical overlap but too large horizontal overlap.

Constructing a Segmentation Graph

The potential split points are ordered according to their horizontal locations. Each neighboring pair of split points defines a grapheme of the image. Neighboring triples define larger segments, and so on. A segmentation graph is a directed graph in which each interior node represents a split point, and the first and the last nodes represent the left and right sides of the image, respectively. All the edges in the graph are directed from left to right. Each edge represents the subimage (segment) defined by the two split points that the edge connects. If the oversegmenter rarely generates more than three graphemes for a single character, any edge that covers more than three graphemes can be removed. A segmentation of the image can be described by a path from the leftmost node to the rightmost node, with the nodes on the path being the final split points. The number of characters is one plus the number of interior nodes on the path. Figure 11(b) shows the segmentation graph for the word images shown in Fig. 11(a).

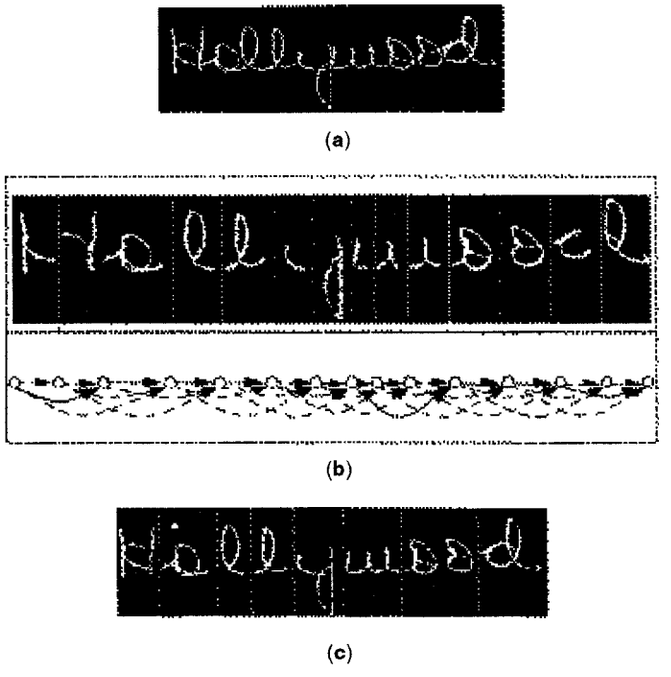

Figure 11. Oversegmentation followed by dynamic programming: (a) original word image, (b) graphemes and segmentation graph, (c) segments on the correct path (following the solid edges) in the segmentation graph.

Finding the Shortest Path

If we assign a length to each edge in the graph, the length of a path is taken to be the sum of the lengths of the edges along the path. The optimal path can then be defined as the shortest path. Now the problem is how to assign these length values so that the shortest path is likely to be the desired path containing the correct segmentation points.

To assign a length value to an edge, a common practice is to use a transformed confidence or probability (e.g., negative confidence or negative log probability) associated with the recognition of the corresponding segment. Various other information can also be integrated to improve the reliability of such an assignment, using neural and fuzzy approaches (29).

A dynamic programming technique is used for finding the shortest path from the leftmost to the rightmost node. The algorithm is modified to generate the shortest N paths (N is typically 3), so that a contextual postprocessor can take the advantage of these alternative paths.

Note that this sequence of characters may not form a valid word (or string) in a dictionary. Many paths in the segmentation graph can contain segments that appear to be valid characters. Very often, the desired path may not appear in the top N paths selected by the dynamic programming algorithm. Therefore, in situations where a lexicon of limited size can be derived (e.g., in postal address recognition, a list of city–state names can be retrieved from a database once we know the zip candidates), a lexicon-driven matching is more desirable (13,29,4). For each entry in the lexicon, the dynamic programming technique is used to choose which path in the segmentation graph best matches with the entry, and a matching score is then assigned to the entry. The entry with the highest matching score is chosen as the recognition hypothesis.

Given a sequence of M graphemes and a string (lexicon entry) of length W, the dynamic programming technique can be used for obtaining the best grouping of the M graphemes into W segments. A dynamic table of size $M \times W$ must be constructed to obtain the best path. The dynamic table size can be reduced to $(M - W + 1) \times W$ by taking advantage of the oversegmentation assumption (i.e., no undersegmentation) (4). The reduction is significant when W is large (e.g., city–state–zip matching). Given a lexicon of L entries, the complexity of the lexicon-driven matching is $O(L \times (M - W + 1) \times W)$. As we can see from the above analysis, the speed of the lexicon-driven system decreases inversely as the lexicon size. Recognition accuracy also decreases when the lexicon size becomes larger. Therefore, it is very important to perform lexicon reduction in a lexicon-driven recognition system.

The lexicon-driven recognition scheme belongs to the context-guided architecture. It brings contextual information into recognition to avoid producing invalid hypotheses.

CONTEXTUAL POSTPROCESSING

Over the past four decades, significant improvement has been made in the recognition accuracy of basic OCR systems. The recognition accuracy of a state-of-the-art OCR system on isolated characters is comparable to human performance (30). Even with this level of performance on isolated characters, the word-level recognition may be still unacceptable. Consider the following scenarios: an OCR with a 98% recognition accu-

racy on isolated digits would achieve only 83.4% accuracy on 9-digit social security numbers under the independence assumption on errors. If the digits are touching, the recognition accuracy could be even lower. A high accuracy recognition of handwritten (especially cursive) words or strings remains an open problem.

Due to the ambiguity in character shapes and in segmenting touching characters, most of the OCR errors cannot be resolved at the character level of recognition—for example, there is an inherent ambiguity in distinguishing between letters "O," "l," "S" and digits "0," "1," "5," respectively. A touching pair of letters "I" and "V" can form a valid letter "N" and the letter "W" can be split into letter pair "IN" or "VV." These OCR errors have to be resolved using contextual information.

In many applications of OCR systems, such as forms processing, postal address recognition, and bank check reading, there always exists a certain amount of contextual information, most of which is introduced by design to increase the reliability of the extracted information. Common sources of contextual information include:

1. *Syntax Constraints.* There are limits on the length of a word or string, and there is an allowable character set for each position in the string. Examples are social security numbers, dates, account numbers, and telephone numbers.
2. *Semantic Constraints.* There is a lexicon of valid words (e.g., city names, state names, person's names), a checksum, etc.
3. *Redundancy in Documents.* To increase the reliability of information, certain entries are duplicated or can be derived from others by certain rules. For example, on a bank check, the LA field and CA field indicate the same amount; on forms (e.g., tax forms), some fields can often be derived from other fields; in postal addresses, the city–state address and the ZIP code are redundant. This redundant information can be used for cross-validation.
4. *Statistical Information.* For recognizing free text, the above constraints are often not applicable. However, for a given language, the co-occurrence of neighboring letters often exhibits a heavily skewed distribution. For example, only 70% of all letter pairs (digrams) occur in English text, and their frequencies are very different. The percentage of valid letter triples (trigrams) is much smaller. The digram and trigram statistics can be derived from a dictionary.

This contextual information can be utilized in many different ways to correct raw OCR errors (31,32). For example, many OCR systems return multiple hypotheses for segmentation and recognition. In this case, a contextual postprocessor may pick out the one that satisfies the given constraints. If none of the hypotheses satisfies the constraints, a new hypothesis is formed by editing the original hypothesis that requires the minimum number of editing operations in order to satisfy the constraints.

THE STATE OF THE ART IN OCR

The performance of an OCR system varies with the quality of the image data and writing style. According to the Fourth Annual Test of OCR Accuracy in 1995 (33), where eight commercial OCR systems were tested on machine-print, multifont English documents, the recognition rates of the best commercial OCR system ranged from 94.26% to 99.03%, depending on the quality of the document images. In May 1992, NIST (National Institute of Standards and Technology) held a conference in which 44 different hand-print OCR systems were considered. This conference tested the ability of these systems to recognize presegmented hand-printed characters. For constrained hand-printed characters, the best OCR system could achieve a recognition accuracy of 98.60% for numerics, 96.36% for uppercase, and 91.00% for lowercase (30). The significantly lower accuracy on lowercase was due to the fact that the test data set had a significantly different distribution from the training data set. Also note that in this test, numeric, uppercase, and lowercase were tested separately with known types. Recognition accuracy on mixed data would be significantly lower. The overall conclusion from the NIST study was that the state-of-the-art OCR systems are as good as human readers on discrete hand-printed characters. However, for nonsegmented hand-print fields (with some cursive fields), the recognition accuracy of the best OCR system was significantly lower than human performance, according to the second NIST conference in 1994 (30). The best OCR system achieved only 60.3% accuracy at field level, while human operators were able to achieve 91.5%. In order to achieve the same accuracy, the OCR system had to reject 50% of the fields.

We should point out that it is not necessary for OCR systems to read hand-print or cursive words and phrases as well as human readers before they can be economically viable for use in various applications (30). This has been evident from many successful real-world applications such as forms processing, postal mail sorting, bank check reading, and desktop OCR.

Combining multiple OCRs has received a considerable interest in recent years. A large number of experimental studies have shown that it is an effective way to design OCR systems with high recognition accuracy, reliability, and robustness (34,35).

It is well known that human readers rely heavily on context, knowledge, and experience. The effectiveness of using contextual information in resolving ambiguity and difficult cases is the major differentiator between human reading ability and machine reading ability. How to effectively use contextual information in segmentation and recognition in OCR systems is still an open problem.

BIBLIOGRAPHY

1. S. Mori, C. Y. Suen, and K. Yamamoto, Historical review of OCR research and development, *Proc. IEEE,* **80**: 1029–1058, 1992.
2. G. Nagy, At the frontiers of OCR, *Proc. IEEE,* **80**: 1093–1100, 1992.
3. S. Gopesitty et al., Automated forms-processing software and services, *IBM J. Res. Dev.,* **40** (2): 211–230, 1996.
4. J. Mao, P. Sinha, and K. Mohiuddin, A system for cursive handwritten address recognition, *Intl. Conf. on Pattern Recognition,* Brisbane, Australia, 1998, pp. 1285–1287.
5. S. N. Srihari, Recognition of handwritten and machine-printed text for postal address interpretation, *Pattern Recognition Lett.,* **14**: 291–302, 1993.

6. J. C. Simon, O. Baret, and N. Gorski, A system for the recognition of handwritten literal amounts of checks, *Document Anal. Syst.,* 1994, pp. 135–156.
7. T. Paquet and Y. Lecourtier, Automatic reading of the literal amount of bank checks, *Mach. Vision Appl.,* **6** (2–3): 151–162, 1993.
8. H. S. Baird, Anatomy of a versatile page reader, *Proc. IEEE,* **80**: 1059–1065, 1992.
9. M. Bokser, Omnidocument technologies, *Proc. IEEE,* **80**: 1066–1078, 1992.
10. S. V. Rice, F. R. Jenkins, and T. A. Nartker, The fourth annual test of OCR accuracy, in annu. rep., UNLV Inf. Sci. Res. Inst., 1995, pp. 11–49.
11. S. Tsujimoto and H. Asada, Major components of a complex text reading system, *Proc. IEEE,* **80**: 1133–1149, 1992.
12. R. G. Casey and E. Lecolinet, Strategies in character segmentation: A survey, *Proc. Int. Conf. Document Analysis Recognition,* Montreal, Canada, 1995, pp. 1028–1033.
13. G. Kim and V. Govindaraju, A lexicon driven approach to handwritten word recognition for real-time applications, *IEEE Trans. Pattern Anal. Mach. Intell.,* **19**: 366–379, 1997.
14. Y. Le Cun et al., Back-propagation applied to handwritten zipcode recognition, *Neural Comput.,* **1**: 541–551, 1989.
15. O. Trier, A. K. Jain, and T. Taxt, Feature extraction methods for character recognition—a survey, *Pattern Recognition,* **29** (4): 641–662, 1996.
16. J. Mao and K. Mohiuddin, Improving OCR performance using character degradation models and boosting algorithm, *Pattern Recognition Lett.,* **18**: 1415–1419, 1997.
17. P. Simard, Y. LeCun, and J. Denker, Efficient pattern recognition using a new transformation distance, in S. J. Hanson et al. (eds.), *Advances in Neural Information Processing Systems 5,* San Mateo, CA: Morgan Kaufmann, 1993.
18. A. Jain, Y. Zhong, and S. Lakshmanan, Object matching using deformable templates, *IEEE Trans. Pattern Anal. Mach. Intell.,* **18**: 267–278, 1996.
19. R. A. Wilkinson et al. (eds.), The First Census Optical Character Recognition System Conference, Tech. Rep., NISTIR 4912, US Dept. Commerce, NIST, Gaithersburg, MD, 1992.
20. M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry,* Cambridge, MA: MIT Press, 1969.
21. D E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Exploration in the Microstructure of Cognition,* Cambridge, MA: MIT Press, 1986.
22. S. Haykin, *Neural Networks: A Comprehensive Foundation,* New York: MacMillan, 1994.
23. U. Kressel and J. Schurmann, Pattern classification techniques based on function application, in H. Bunke and P. S. P. Wang (eds.), *Handbook of Character Recognition and Document Image Analysis,* Singapore: World Scientific, 1997, pp. 49–78.
24. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis,* New York: Wiley, 1973.
25. K. Fukunaga, *Introduction to Statistical Pattern Recognition,* 2nd ed., New York: Academic Press, 1990.
26. B. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques,* Los Alamitos, CA: IEEE Computer Society Press, 1991.
27. M. Mohamed and P. Gader, Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques, *IEEE Trans. Pattern Anal. Mach. Intell.,* **18**: 548–554, 1996.
28. M. Y. Chen, A. Kundu, and S. N. Srihari, Variable duration hidden Markov model and morphological segmentation for handwritten word recognition, *IEEE Trans. Image Process.,* **4**: 1675–1688, 1995.
29. P. D. Gader et al., Neural and fuzzy methods in handwriting recognition, *Computer,* **30** (2): 79–86, 1997.
30. J. Geist (eds.), The second census optical character recognition system conference, Tech. Rep. NISTIR 5452, US Dept. Commerce, NIST, Gaithersburg, MD, 1994.
31. A. Dengel et al., Techniques for improving OCR results, in H. Bunke and P. S. P. Wang (eds.), *Handbook of Character Recognition and Document Image Analysis,* Singapore: World Scientific, 1997, pp. 227–254.
32. R. Lorie, V P. Riyaz, and T. K. Truong, A system for automated data entry from forms, *13th Int. Conf. Pattern Recognition,* vol. III, Vienna, Austria, 1996, pp. 686–690.
33. T. A. Nartker and S. V. Rice, OCR accuracy: UNLV's third annual test, *INFORM,* **8** (8): 30–36, 1994.
34. T. K. Ho, J. J. Hull, and S. N. Srihari, Decision combination in multiple classifier systems, *IEEE Trans. Pattern Anal. Mach. Intell.,* **16**: 66–75, 1994.
35. R. Bradford and T. Nartker, Error correlation in contemporary OCR systems, *Proc. First Int. Conf. Document Analysis Recognition,* St. Malo, France, 1991, pp. 516–523.

K. M. MOHIUDDIN
JIANCHANG MAO
IBM Research Division