Prototype Extraction and Adaptive OCR

Yihong Xu, *Member*, *IEEE*, and George Nagy, *Senior Member*, *IEEE*

Abstract—To maintain OCR accuracy with decreasing quality of page image composition, production, and digitization, it is essential to tune the system to each document. We propose a prototype extraction method for document-specific OCR systems. The method automatically generates training samples from unsegmented text images and the corresponding transcripts. It is tolerant of transcription errors, so a transcript produced automatically by an imperfect omnifont OCR system can be used. The method is based on new algorithms for estimating character widths, character locations in a word, and match/nonmatch probabilities from unsegmented text. An experimental word recognition system is designed and developed to combine prototype extraction algorithms and segmentation-free word recognition. The system can adapt itself to different page images and achieve high recognition accuracy on heavily degraded print.

Index Terms—Optical character recognition, adaptive classification, template matching, segmentation, document image analysis, text reader.

1 Introduction

Poor page production and digitization procedures can result in document image degradation severe enough to cause current omnifont OCR systems to make an unacceptable number of errors.

Research has revealed that *document-specific* OCR systems, which are single-font OCR systems designed for a particular typeface, are far more accurate than omni-font systems [14], [2]. For document-specific OCR systems, the training set should contain the representive character bitmaps from the page image we want to recognize (we call these character bitmaps *prototypes*). However, manually extracting prototypes from page images is very expensive. The lack of automatic prototype extraction methods has hindered the development of document-specific OCR systems.

We propose an automatic prototype extraction method which is based on comparing the bitmaps of pairs of words that contain the same character. Then, prototypes are used to train a document-specific OCR system to perform word recognition on page images of the same or similar quality and typesetting. Some preliminary results of this research have already been published [15], [16], [17]. The contribution presented here is a sound Bayesian method which makes use of as much information as possible from the word bitmaps and labels to estimate character widths, character locations, and match/nonmatch probabilities. We designed a document-specific word recognition system and integrated it with a commercial OCR development package to carry out bootstrap recognition. Our experimental results indicate that bootstrap recognition increases the recognition accuracy. The method advocated here can be combined with morphological and lexical correction. But, because we wanted to focus on shape adaptation, we forbore from exploiting linguistic context. Our method helps most just where context is weak, such as unpredictable digit fields, proper names, and abbreviations.

Extending the recognition process from characters to words and using document-specific methods has long been advocated by Hong and Hull [7] and by Spitz [20]. Ingold used character identities derived from the text under consideration [8]. Among the most exciting current methods for document-specific OCR are the sophisticated dynamic programming approaches developed by Kopec et al. [9], [11].

2 Document-Specific OCR System and Prototype Extraction

In a conventional omnifont OCR system, the recognition engine is trained with a large in-house training set of multifont character bitmaps. Document-specific OCR is motivated by the observation that the error rate of a font-specific recognition system is significantly lower than that of an omnifont system. Because general typesetting rules require uniform typeface appearance to maintain document readability, most pages are set in a single typeface. The occasional italic and boldface can be considered as part of an extended typeface. We expect that even the image deformations due to printing, copying, and scanning tend to remain the same within one page image or a batch of page images. Interfont confusions can also be avoided by restricting the classifier to the current font.

For a given document image, the most representative training samples will be the character bitmaps drawn from the same image. We cannot expect an OCR system to perform well on heavily degraded document images if it has never "learned" what a poor quality character image looks like (Fig. 1).

How can an OCR system obtain a representative training sample for each document? Manually segmenting the page image into character bitmaps is expensive and impractical.

- *Y. Xu is with Hewlett-Packard Laboratories, 1501 Page Mill Rd., MS 3U-3, Palo Alto, CA 94304. E-mail: xuy@hpl.hp.com.*
- *G. Nagy is with the Electrical, Computer, and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY 12180.*

Manuscript received 20 Jan. 1999; revised 19 Oct. 1999.
Recommended for acceptance by T.K. Ho.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 109016.

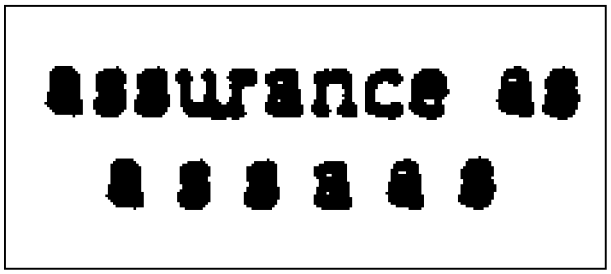


Fig. 1. a and s from degraded images.

Character segmentation methods based on general features, such as vertical white spaces between characters, or character pitch information, tend to have high error rates on both very dark or very light page images.

We are therefore motivated to devise a practical and accurate method of extracting training samples for trainable, document-specific OCR systems. We propose a Bayesian *word-shift* algorithm for automatically extracting prototypes from unsegmented text images using the known transcript of the image. We use an imperfect transcript generated by omnifont OCR, but we must still align each symbol of the transcript with its character bitmap in the page image. The appropriate alignment information is estimated from the transcript and the word bitmaps.

For each character class, multiple prototypes are extracted. A *prototype selection* routine is used to evaluate the prototype quality by shape. Low quality prototypes due to noise or imperfect extraction are eliminated from the prototype set. Templates are built by averaging the selected prototypes for each class.

2.1 Terminology

Before we introduce the proposed algorithm, we would like to emphasize and clarify some definitions.

Page image: The bitmap which is digitized from a document page on paper is called a page image. A page

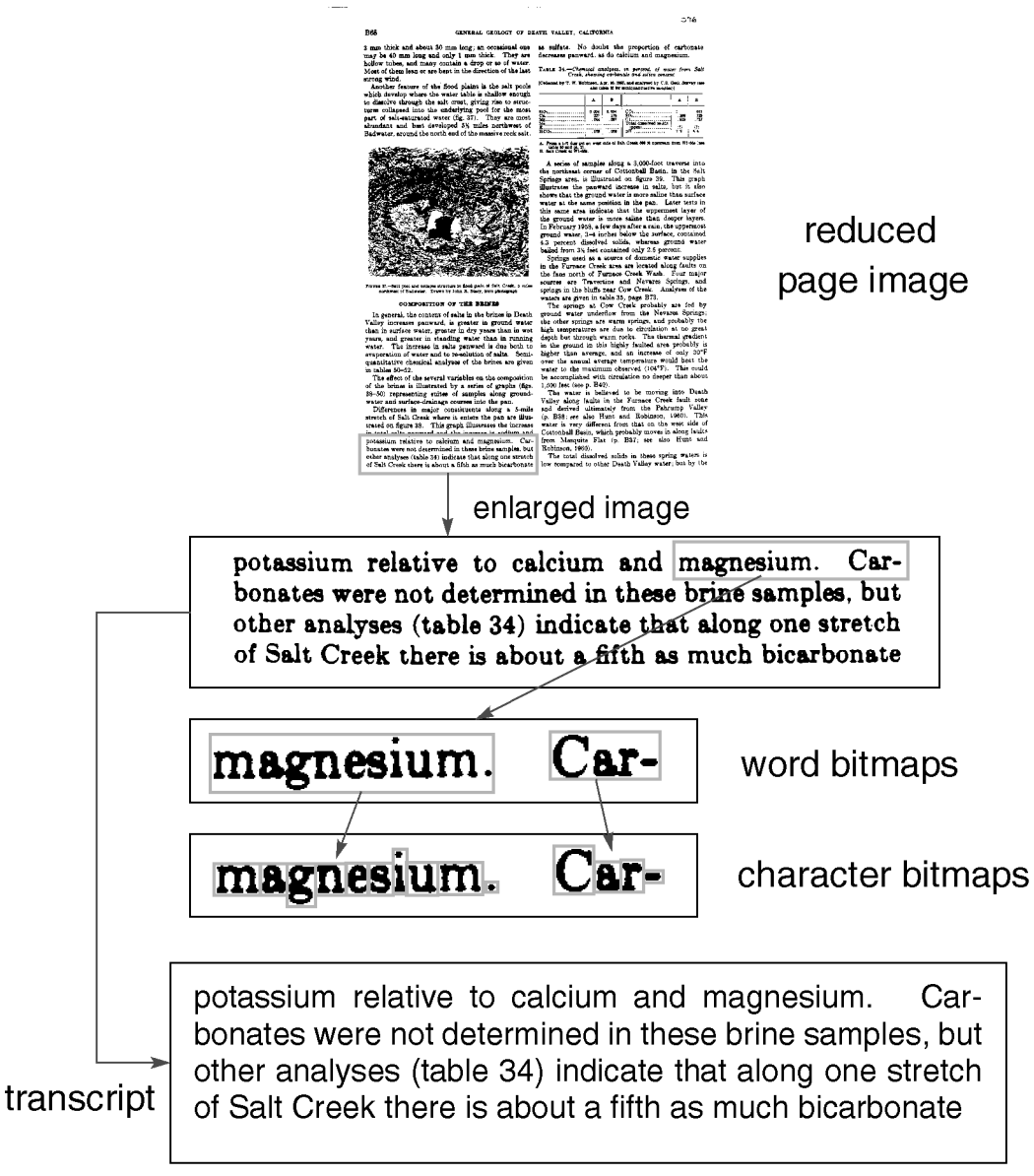


Fig. 2. Top: a sample page image and examples of word bitmaps and character bitmaps. Bottom: transcript of the page image.

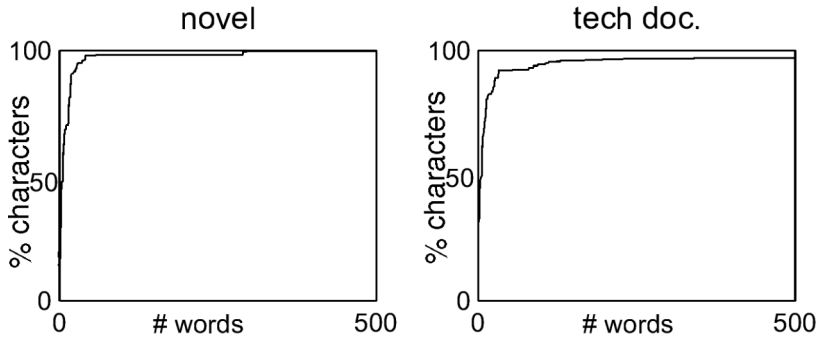


Fig. 3. The statistics of number of characters covered by prototypes vs. number of words from the first 500 words of a novel and a technical document.

image can be divided into smaller units, such as *word bitmaps* and *character bitmaps* (Fig. 2). A word bitmap does not necessarily correspond to a lexical word. It could correspond to a partial lexical word, such as a stub due to hyphenation, to a word followed by a punctuation symbol, or to a sequence of tightly spaced lexical words.

Transcript: For a given page image, a sequence of symbols associated with the character bitmaps is called a transcript. *Ground truth* is a special transcript of high accuracy. We want to differentiate the transcript from the ground truth because an imperfect transcript can be generated by an OCR device without human interaction.

Prototype: The individual labeled character bitmaps are called prototypes.

Template: A set of prototypes from the same class can be aligned and superimposed to construct a template. A template is a 2D array where each element is the sum of the corresponding black pixels from all the constituent prototypes. Each sum is then converted to an estimate of the probability that the corresponding pixel is black.

2.2 Language Model—Letter Frequency

A document-specific OCR system is trained with character bitmaps extracted from a portion of the given document image. Thus, the first issue is how much text is required to provide an adequate sample. Different languages have their own alphabet of symbols and morphology. In English, the most frequent 21 lower-case characters account for about 96 percent of all text, while k, x, j, q, and z, all the upper-case letters, and punctuation, comprise less than 4 percent of the text [10], [6]. We can rely on the statistics of the English language to extract at least four prototypes for each of the alphabetic classes that represent 96 percent of all text from a five-line text passage of about one hundred words (500 letters).

Fig. 3 shows the fraction of the text for which prototypes can be generated from a passage of given length. In the first text sample, more than 98 percent of characters belong to the classes which occur in the first 100 words. Similarly, more than 94 percent of characters in the second text sample are covered, even though we have captured less than two-thirds of the classes. For a full-length printed page of 500 words, more than 99.8 percent and 97.1 percent characters are covered in the two text samples, respectively. Therefore, even in a document with a great variety of symbols, we can expect the first 100 words to provide enough prototypes to cover almost all of the text.

When more and more words are used to obtain prototypes, the effect of increasing the number of words is twofold. A class which occurs for the first time can be added to the prototype set. More words will also provide better estimates for the existing classes and therefore improve the prototype quality. We expect that the system can modify itself for both cases, which we call *adaptation*.

3 Word-Shift: A Bayesian Prototype Extraction Algorithm

If two words contain characters from the same class according to the transcript, then we can extract the common character bitmaps by examining the similarities between these two word bitmaps. In Fig. 4, the words linear and oscillations have a common character a. We shift these two words against each other to calculate the similarity between each pair of column bitmaps. If only the columns within the two as are similar between the two words, then we can extract these columns from the two words as prototypes for class a.

But, for most of the scanned word bitmaps, the similarity between columns is far more complicated than in the ideal case. As shown in Fig. 5a, not every column of a has high similarity score, even in two words which are perfectly aligned on a. Furthermore, there are many false high scores elsewhere, such as r matching with partial n in Fig. 5b. To find the true match, we have to estimate the most likely location of the prototype in a word bitmap, estimate the

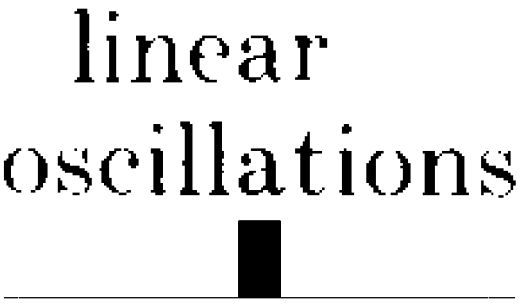


Fig. 4. Two words contain same character a; bottom is the ideal matching score when the two as are aligned.

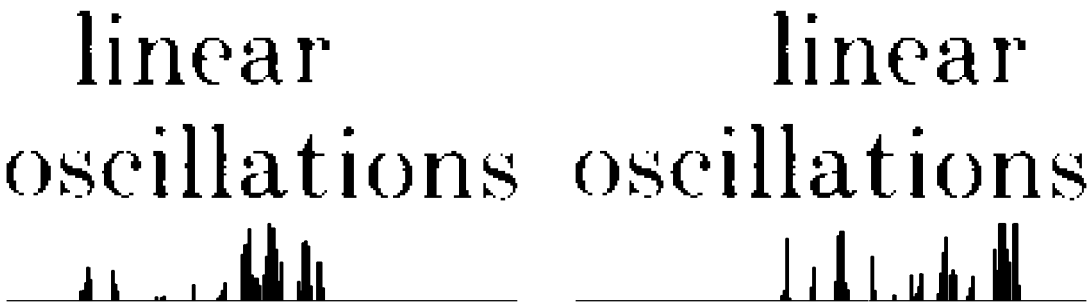


Fig. 5. Real matching scores: (a) Two words are aligned at a, (b) in a different shift position, the r from the first word linear is well-matched with part of the n in the second word.

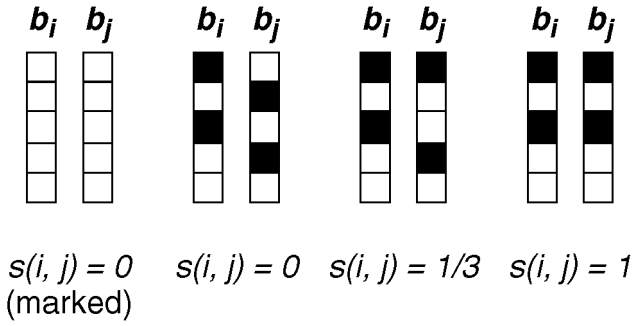


Fig. 6. Similarity scores for an all-white column pair, a nonmatching column pair, a partially matching column pair, and a perfectly matching column pair.

width of the prototype, and find the columns which have the highest overall match probability.

3.1 The Word-Shift Problem

We approach the word-shift problem probabilistically. Given two word bitmaps B_1 and B_2 which contain l_1 and l_2 columns, respectively, the similarity score between column i from B_1 and column j from B_2 is $s(i,j)$, where $0 \le i \le l_1$ and $0 \le j \le l_2$. S is the set of similarity scores between all possible overlapping column pairs when we shift two word bitmaps against each other.

To calculate the column-wise similarity scores $s(i,j)$, we align the two word bitmaps at their baselines and normalize them to the same height by patching 0s (white pixels) to the top or bottom of the word with smaller height. We consider each column as a vector $\mathbf{b}$ with 0 (white) or 1 (black) elements. For the ith column from the first word and the jth column from the second,

$$s(i,j) = \frac{\mathbf{b}_i \wedge \mathbf{b}_j}{\mathbf{b}_i \vee \mathbf{b}_j}.$$

We want to extract columns which contain some black pixels and belong to a character bitmap. We mark the scores of all-white column pairs to distinguish them from $s(i,j) = 0$, but $(\mathbf{b}_i \vee \mathbf{b}_j) \neq 0$. We will exclude these marked-up all-white column pairs from the match/nonmatch probability estimation that we will discuss next. Fig. 6 shows similarity scores for different column pairs.

We shall maximize the conditional probability $P(x,y,w|S)$ that w columns from B_1 starting at x, and w columns from B_2 starting at y, originate from similar characters (Fig. 7). This a posteriori probability can be expressed, using Bayes' rule, as

$$P(x,y,w|S) = \frac{P(S|x,y,w)P(x)P(y)P(w)}{\sum_{all\ x,y,w} P(S|x,y,w)P(x)P(y)P(w)}. \quad (1)$$

Here, $P(x)$, $P(y)$ are the prior probabilities of the location of the expected prototypes in each word bitmap. $P(w)$ is the a priori probability distribution of the prototype width. $P(x)$, $P(y)$, and $P(w)$ can be reasonably assumed to be statistically independent. $P(S|x,y,w)$ is the most complicated term in the above Bayesian formula because it must take into account that some pairs of columns come from similar bitmaps and other pairs do not.

3.2 Probability Distribution of Prototype Width

The width estimation is based on the sidebearing model that describes the placement of each character with respect to adjacent characters [12]. The model specifies the character width w, left sidebearing l, right sidebearing r, and character pair spacing adjustments k_{12} (*kerning*). The intercharacter space, $d = r_1 + l_2 + k_{12}$, depends on each character pair.

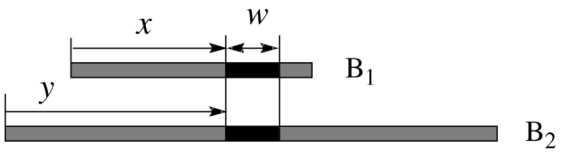


Fig. 7. The Word-Shift problem.

Fig. 8. The intercharacter space distribution with zero mean.

We know only the width of each word bitmap and the label sequence corresponding to the word bitmap. To extract the necessary metrics, we make two simplifying assumptions:

1. The width w_i of each character class i has a uniform distribution. The mean of the width distribution is the average number of columns in a bitmap of this class and the variance is due to noise and sampling errors.
2. The intercharacter space d also has a uniform distribution. The mean of the space distribution is the average intercharacter space of the given font (i.e., the sum of the average left sidebearing and right sidebearing) and the variance reflects the pair spacing adjustment for different character pairs.

We treat the intercharacter space as a special character class. Instead of estimating sidebearings for different characters and estimating pairwise character spacing adjustment for every different pair, we estimate the average width of the inter-character space and its variance together with the widths of the other character classes. Fig. 8 shows two examples of intercharacter space. The variance will take care of the kerning adjustment.

The sum of the character widths (including spaces) according to the word label must be equal to the known bitmap width for each word. So, we can restate the problem as that of solving multivariable regression equations. Each word bitmap gives rise to a single equation. With N_c classes and N_w words, we model the problem as a linear equation system:

$$\mathbf{Aw} = \mathbf{b},$$

where $\mathbf{A}$ is an $N_w \times N_c$ matrix, $\mathbf{w}$ is an $N_c \times 1$ vector, and $\mathbf{b}$ is an $N_w \times 1$ word length vector. We usually have more word bitmaps than character classes, i.e., $N_w > N_c$, so the system is overdetermined. The least-square solution is

$$\mathbf{w} = (\mathbf{A^T A})^{-1}\mathbf{A^T b}.$$

To guarantee positive mean widths, we minimize $|\mathbf{Aw} - \mathbf{b}|^2$ subject to $\mathbf{w} \geq \mathbf{0}$. An algorithm for solving the nonnegative least-square problem, and the finite convergence of the algorithm, is stated in [13]. MATLAB includes this algorithm as a standard function.

3.3 Probability Distribution of Character Location

The location of each character bitmap with respect to the beginning of its word bitmap is the sum of the widths of the preceding character bitmaps (including intercharacter spaces). Therefore, the sum of the widths of a string of character bitmaps can be obtained by convolving the individual width distributions. When the character is far away from the beginning of the word, the cumulative errors will corrupt the location estimation. It is better to estimate the location from both ends of a word (Fig. 9).

3.4 Probability Distribution of Match/Nonmatch Similarities

We have derived estimates of the prior probabilities $P(x)$, $P(y)$, and $P(w)$ in (1). Now, we focus on the term $P(S|x, y, w)$. If we assume that all column pairs are independent, then:

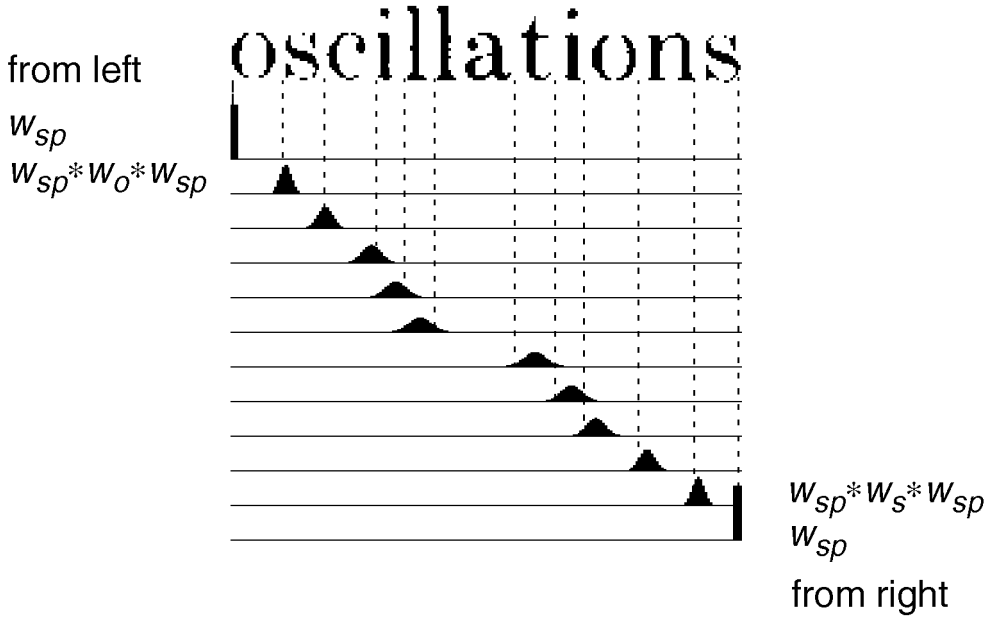


Fig. 9. Probability distribution of character bitmap locations in a word bitmap, estimated from both ends.

$$P(S|x,y,w) = \prod_{0\le i\le l_1, 0\le j\le l_2} P(s(i,j)|x,y,w)$$
$$= \prod_{x\le i\le x+w, y\le j\le y+w} P(s(i,j)|x,y,w) \prod_{other(i,j)} P(s(i,j)|x,y,w)$$
$$\stackrel{\text{def}}{=} \prod P_m(s(i,j)) \prod P_n(s(i,j)).$$

This is the probability that the w-column bitmap portions, $x \le i \le x+w$ from the first word and $y \le j \le y+w$ from the second word, belong to characters of the same class. As mentioned before, not all column pairs from the same-class characters have high similarity scores. On the other hand, column pairs from different character bitmaps could also achieve high similarity scores. The two terms we defined above, $P_m(s(i,j))$ and $P_n(s(i,j))$, are probabilities of column similarity scores from character bitmaps of the same class and from character bitmaps of different classes, respectively. We call these two terms *match probability* and *nonmatch probability*.

To estimate the match and nonmatch probability distributions, we extract some prototypes from a scanned page image. For any two prototypes from the same class, we align them at the best matching position and calculate their column-wise similarity scores. When two prototypes belong to different classes, we shift them against each other and calculate the column similarity scores at all possible shift positions. We exclude the all-white column pairs from the estimation. Fig. 10 shows cumulative probability distributions of column similarity scores.

We approximate these two cumulative probability distributions with the cumulative probability distributions shown in Fig. 11. Because these distributions do not vary much from page to page, they are permanently set in our algorithm to convert similarity scores to probabilities.

3.5 Algorithm

The Bayesian word-shift prototype extraction method finds the positions of a pair of character bitmaps in two word bitmaps with the largest a posteriori probability in the Bayesian formula (1), i.e.,

$$(x^*, y^*, w^*) = argmax[P(x,y,w|S)].$$

The input of the algorithm is a set of scores S that reflects the similarity between every pair of columns in two words that contain at least one instance of the same letter. Character width estimation and character location estimation were discussed above. The similarity scores are converted to probabilities according to the match/nonmatch probability distributions. Then the a posteriori probabilities are computed for every probable location and width of the target character. The pair of character bitmaps with the largest a posteriori probability is saved.

The pseudocode of the word shift algorithm is shown in Fig. 12. The match probability conditioned on similarity scores is $P(x,y,w|S)$. The range $W^+_{x,y}$ of column scores consists of every pair of columns within the window w located at x in word 1 and y in word 2. $W^-_{x,y}$ consists of every other column pair in the two words, at every (x,y). $P(x), P(y), P(w)$ are the previously estimated a priori probabilities for location and width (x, y, and w are assumed to be independent of one another).

4 WRAP: Word Recognition with Adaptive Prototype

We now focus on how to use the prototypes extracted from document images to increase the recognition accuracy. We develop an experimental *Word Recognition with Adaptive Prototypes (WRAP)* system. This system is designed for document-specific OCR. The prototype extraction algorithm and a template-matching word-recognition algorithm are integrated to perform the OCR task.

4.1 System Overview

In the last section, we discussed the algorithm for extracting prototypes from a page image with a known transcript. After we obtain prototypes for the specific page, we can reduce the recognition task to a single-font recognition problem.

The overall system structure is illustrated in Fig. 13. When a document page image is presented to WRAP, a *page image preprocessing* routine is applied to estimate the page skew angle, analyze the page layout, segment the page image into word bitmaps, and estimate the word baselines.

Transcript generation is accomplished manually or automatically. Multiple prototypes are extracted with the Bayesian algorithm for every character class which appears in the text according to the transcript. The *prototype selection* routine is used to reject anomalous prototypes. Templates are built by averaging the selected prototypes for each class using the *template construction* routine.

For recognition, we use a simple template matching routine. We recognize each word without character-level segmentation. *Level building*, a dynamic programming algorithm, is used to perform the word recognition by finding a sequence of the best matching templates for each word bitmap.

4.2 Page Image Preprocessing

The objective of the page image preprocessing is to select text regions, perform skew correction, segment the word bitmaps from the input page image, and estimate the baseline for each word bitmap. Word bitmaps with baseline information are the basic units for both system training and word recognition in the WRAP system. Since WRAP is intended primarily as a postprocessor for existing OCR systems, it has only simple preprocessing algorithms based on published methods [1], [4]. In this article, we show how to make use of the preprocessing routines of the omnifont OCR that produces the transcript.

We avoid rotating the page image to preserve the character bitmap shape. We assume that the baseline within each word bitmap is still horizontal, but it is shifted along the real baseline of the text line. Fig. 14 shows the baseline of each word bitmap in a text line with skew.

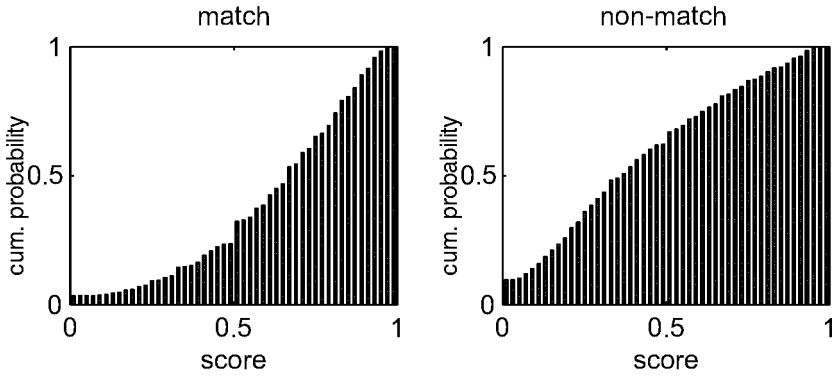


Fig. 10. Empirical cumulative column probability distributions of match and nonmatch similarity scores.

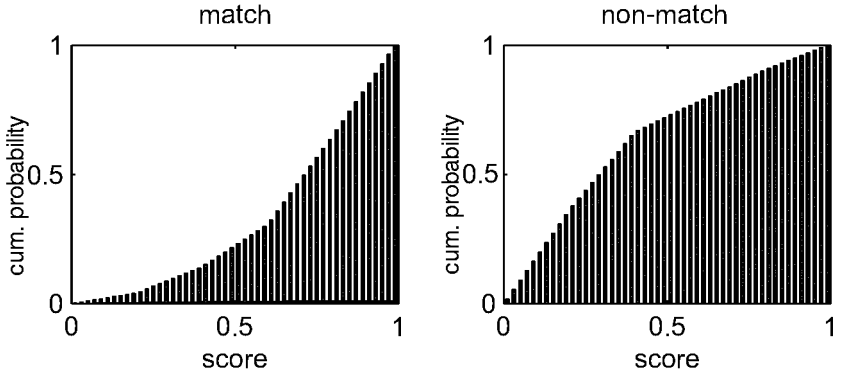
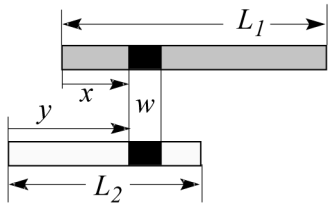


Fig. 11. Approximated match and nonmatch cumulative probability distributions of column similarities.



// *x, y: location of window in each word*
// $P_+(\cdot)$: *match probability*
// $P_-(\cdot)$: *no-match probability*
// *P(x), P(y): location probability*
// *P(w): width probability*
// L_1: *width of word 1 ;* L_2 *: width of word 2*

PROCEDURE *word-shift (word1, word2)*
FOR $\forall x, \forall y, \forall w$ **DO**

$$Q(x, y, w) = P(x)\,P(y)\,P(w) \prod_{i \in W^+_{x,y}} P_+(S_i) \prod_{i \in W^-_{x,y}} P_-(S_i) \propto P(x, y, w \mid S);$$

END

$$(x^*, y^*, w^*) = \textit{argmax}\, Q(x, y, w);$$

END

Output:
Extract w^* columns from each word at shift position x^* and y^* respectively.

Fig. 12. Pseudocode for word-shift and character bitmap extraction.

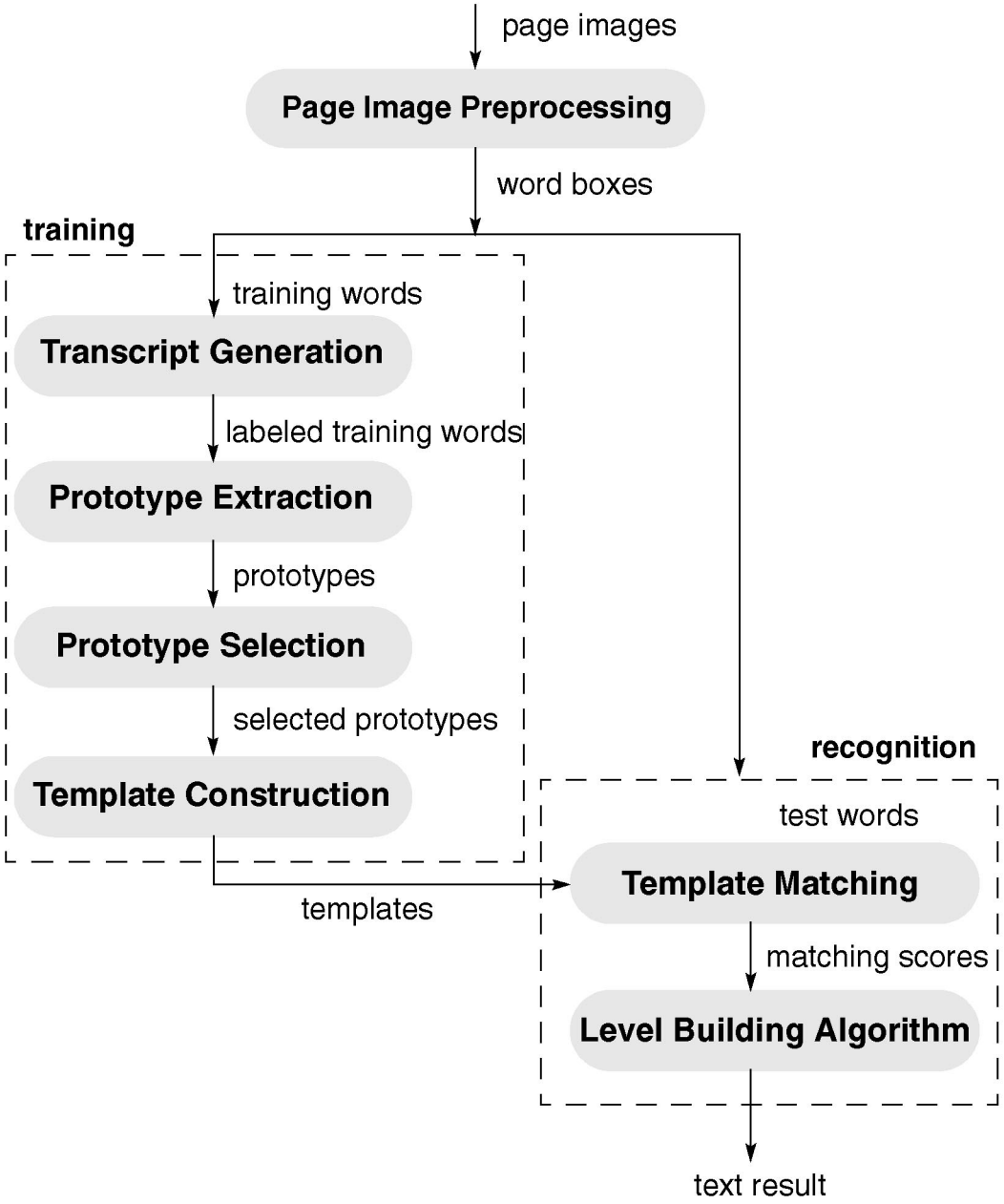


Fig. 13. WRAP: overall system structure.

4.3 System Training

We will train a classifier with the prototypes extracted from the specific document page image. We have already discussed the core prototype extraction algorithm in the last section. Here, we will focus on some implementation issues.

4.3.1 Transcript Generation

We have already documented prototype extraction by using a transcript which is manually generated from a short paragraph [15]. Here, we demonstrate use of a commercial OCR device to generate the transcript. In order to align the commercial OCR results with our word segmentation results (two systems may break words in different places), we need a package which can not only recognize characters, but also provide the coordinates of each word bitmap. In

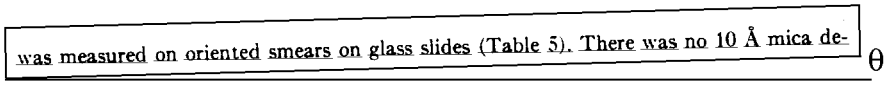


Fig. 14. Text line with the shifted word baselines.

```
2. Any extraneous matter included in any statement by a Member,
either under the 1-minute rule or permission granted to extend at
this point, will be printed in the "Extensions of Remarks" section,
and that such material will be duly noted in the Member's statement
as appearing therein. One-minute speeches delivered during the
morning business of Congress shall not exceed 300 words. Statements
exceeding this will be printed following the business of the day.
```

```
a:19 -- 3(5/10), 4(2/6), 7(1/3), 8(3/9), 10(1/1),
        19(3/7), 22(1/2), 32(4/8), 34(1/3), 35(3/4),
        37(2/8), 37(7/8), 45(3/9), 46(1/2), 47(1/9),
        47(5/9), 58(3/5), 63(3/10), 74(2/4)
b: 8 -- 9(1/2), 11(4/7), 26(1/2), 39(1/2), 44(4/8),
        55(1/8), 67(1/2), 71(1/8)
c: 6 -- 5(3/8), 33(3/8), 36(3/4), 50(5/8), 60(3/6),
        64(3/9)
d:17 -- 5(6/8), 5(8/8), 13(3/5), 19(7/7), 21(6/6),
        27(7/7), 34(3/3), 40(1/4), 41(5/5), 51(1/9),
        51(9/9), 52(1/6), 60(6/6), 62(4/6), 64(6/9),
        68(7/7), 74(1/4)
```

Fig. 15. Top: the sample transcript. Bottom: the original lexical list.

this case, we turn off the word segmentation in WRAP and use the word coordinates provided by the commercial OCR package instead.

4.3.2 Transcript Analysis

The objective of transcript analysis is to keep track of the words which contain the same character. A lexical list is generated from the transcript to record 1) how many characters of each class occur in the page image; 2) in which words a specific character occurs; 3) the position of a character in a word. Fig. 15 shows a transcript and the partial lexical list obtained initially from this transcript. For example, `a:19` means that there are 19 as in this paragraph. The list `3(5/10)` shows that the first a is from the third word, which contains 10 characters, and a is the fifth character.

The lexical list can also be sorted in different orders to provide more reliable prototype extraction. As discussed in the last section, location estimation is more accurate for the characters near the ends of the words (left end or right end). So, we can sort the list for each class by the location of a character in a word.

In the word-shift prototype extraction algorithm, we always have to shift two words against each other and calculate all possible column matching scores. The shorter the words, i.e., fewer scores need to be calculated, the faster the whole procedure. So, another way to sort the lexical list is by word length.

The transcripts generated by our commercial package also include a confidence number with each word label. We can sort the lexical list by these confidence numbers and extract prototypes from words with high-confidence labels.

4.3.3 Prototype Selection

For each character class, we extract multiple prototypes. Redundancy is a key point here to improve the robustness of the algorithm. Fig. 16 shows 15 prototype i's from a sample page.

Our objective is to eliminate outliers. We expect that the majority of prototypes from the same class are correct and similar to each other because of the consistency of the character appearance in the same page image. We apply the well-known *Agglomerative Hierarchical Clustering* method to cluster the different prototypes of a class by shape [3]. Fig. 17 shows the clustering result for the prototypes in Fig. 16. We keep the largest cluster as good prototypes to be used for template construction.

4.4 Template Construction

In WRAP, the core classifier is a single-font template matching classifier trained on document-specific prototypes. Two interlaced steps are required to construct a template: *prototype alignment* and *prototype summation*.

All prototypes extracted by the word-shift algorithm retain the baseline of their parent word. To compensate for skew, we allow a one or two pixel vertical adjustment. Then, we shift the first two prototypes horizontally and vertically to find the best alignment. The smallest Hamming distance indicates the best matching position.

An initial template is obtained by prototype summation, i.e., by adding up the overlapping pixels of the two aligned prototypes. Then, a new prototype from the same class is aligned and added to this initial template. A complete template is eventually constructed by superimposing every selected prototype, one by one, from the same class. Fig. 18 shows a set of prototype es and the corresponding template.

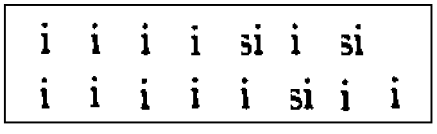


Fig. 16. Prototype `i` from a sample text page.

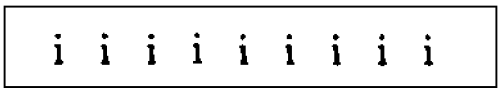


Fig. 17. All selected prototypes `i`.

4.5 Segmentation-Free Word Recognition

In WRAP, the word recognition problem can be stated as finding the optimal sequence of templates which achieves the best match for the test word bitmap.

4.5.1 Template Matching

A template is a *2D* array which is constructed from a set of prototypes from the same class. If we use p prototypes to construct a template, we can normalize each element of the template as $\frac{b+1}{p+2}$, where b is the number of black pixels. Then, each element in the normalized template is between $(0, 1)$ and its value indicates the probability that the corresponding pixel in the character bitmap is black, given a uniform a priori probability of "blackness."

We interpret the template matching algorithm in the framework of statistical decision theory. Each template is a *2D* array with n_c columns and n_r rows, where each element $p(i, j)$ is the probability of the corresponding pixel being black.

$$T = \{p(i,j) : p(i,j) \in [0,1], 1 \le i \le n_c, 1 \le j \le n_r\}.$$

Similarly, an $n_c \times n_r$ binary bitmap, which we try to match with the template, can be written as

$$B = \{b(i,j) : b(i,j) \in \{0,1\}, 1 \le i \le n_c, 1 \le j \le n_r\},$$

where $b(i, j) = 0$ indicates a white pixel and $b(i, j) = 1$ indicates a black pixel.

We assume that pixels in the character bitmap are independent (a very questionable, but expedient assumption). Then, the logarithm of the probability of observing character bitmap B for the given template T is:

$$\log P(B|T) = \sum_{i,j} [b(i,j) \log p(i,j) + (1 - b(i,j)) \log (1 - p(i,j))].$$

We have a set of templates $\mathbf{T} = \{T_k : 1 \le k \le N\}$, where N is the number of templates. The test bitmap B will be classified to class k^*, where

$$k^* = \arg\max_k \log P(B|T_k).$$

More elaborate template matching metrics are discussed in [22].

4.5.2 Level-Building Word Recognition Algorithm

For a set of N templates and L characters in the word, there are N^L possible template sequences. Since we do not know L, we have to try every possible value of L. An exhaustive approach would have to concatenate every possible combination of templates and perform a global template matching. *Level-building* is a dynamic programming algorithm which has been successfully used in connected speech recognition. We modify the level-building algorithm to provide a nonexhaustive solution to the word recognition problem.

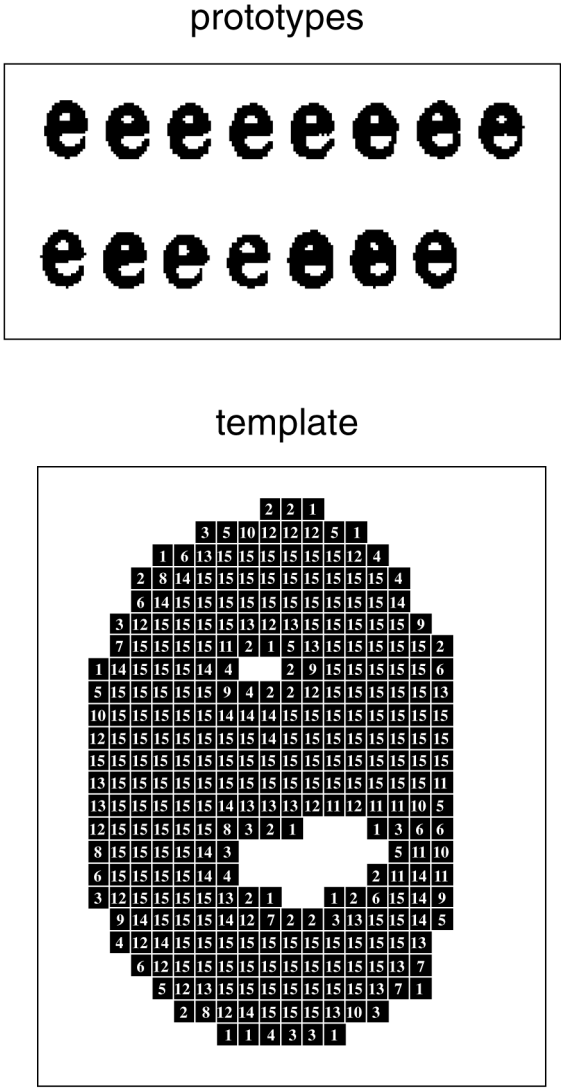


Fig. 18. Top: prototype examples from the sample page image. Bottom: the resulting template.

Since we know the baseline of every template, and of each test word bitmap as well, we can always align the template and word bitmap at the baseline in the vertical direction. To simplify the notation, we redefine each template as a *1D* column array

$$T_k = \{t_k(1), t_k(2), ..., t_k(c_k)\},$$

where c_k is the number of columns in template T_k and $t_k(i)$ represents the ith column of template T_k. The test word bitmap can also be denoted as a column array

$$B = \{b(1), b(2), ..., b(N_c)\},$$

where N_c is the number of columns in the test word bitmap and $b(i)$ represents the ith column of the bitmap B.

We can build an N_c-column template array by concatenating a sequence of L templates, i.e.,

$$\mathcal{T} = \{T_{s_1} \oplus T_{s_2} \oplus ... \oplus T_{s_L}\} = \{t(i) : 1 \le i \le N_c\},$$

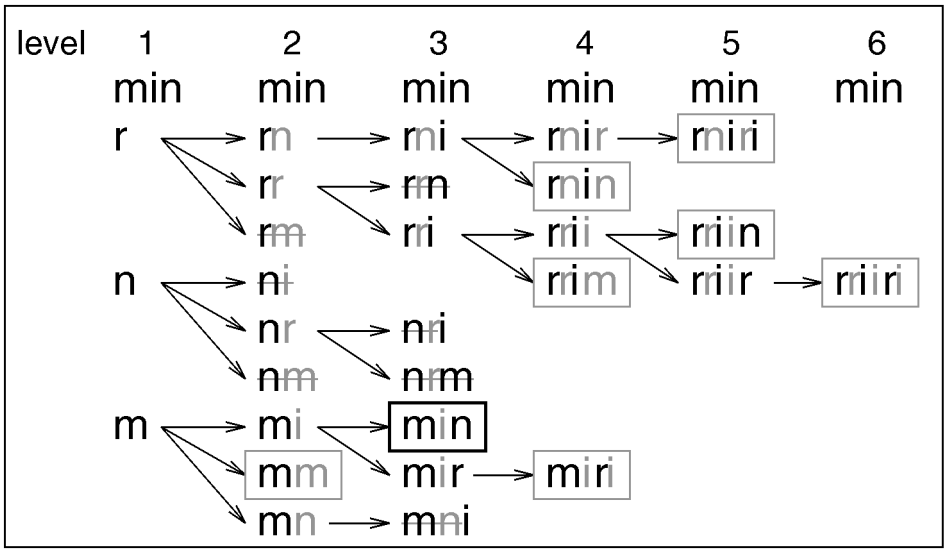


Fig. 19. Illustration of levels (left to right) of the level-building algorithm.

where each s_l ($1 \leq s_l \leq N$) is the index of some template. Hence, the word recognition problem is to find the optimal sequence of templates, $\mathcal{T}^*$, which best matches word bitmap B.

Using the template matching method we discussed above, the matching probability between the template sequence $\mathcal{T}$ and the given bitmap B can be written as $P(\mathcal{T}|B)$. To determine a global best match $\mathcal{T}^*$, we optimize $P(\mathcal{T}|B)$ over every possible sequence of templates, $\{T_{s_1}, T_{s_2}, \ldots, T_{s_L}\}$, and over every possible value of L, i.e.,

$$\mathcal{T}^* = \arg\max P(\mathcal{T}|B).$$

We denote level l as the hypothetical number of characters in the test word bitmap B, i.e., we assume at level 1 that there is only one character, at level 2 that there are two characters, and so on. At each level, we determine the best-fitting template array with l concatenated templates.

We define $P_l^k(i)$ as the matching probability, at level l, using template T_k, and ending at column i of the test word bitmap. Clearly, $P_l^k(i)$ is defined for $1 \leq l \leq L$, $1 \leq k \leq N$, and $1 \leq i \leq N_c$.

Fig. 19 illustrates the level-building algorithm. From left to right, each column corresponds to a different level. At the first level, every template is matched against word bitmap B starting at column 1. The end column depends on the number of columns n_k in each template k. The output of level 1 is the array of accumulated probabilities over the end range $m_{\min}(1) \leq i \leq m_{\max}(1)$. For each column in the ending range, we need to store:

- the maximum probability at level l to column i:

$$P_l{}^*(i) = \max_{1 \leq k \leq N} [P_l^k(i)],$$

- the template index which achieves the above probability:

$$s_l{}^*(i) = \arg\max_{1 \leq k \leq N} [P_l^k(i)],$$

- a backpointer to the best column ending at the previous level:

$$F_l{}^*(i) = F_l^{s_l{}^*(i)}(i).$$

Here, we define $F_1{}^*(i) = 0$ for every column i since every template starts at column 1 at the first level. By storing only $P_l{}^*(i)$, $s_l{}^*(i)$, and $F_l{}^*(i)$, we significantly reduce the storage at each level and still retain all the information required to select the optimal path through the entire trellis.

The second level computation starts after the first level computation is finished. For each template T_k, the range of starting columns is $m_{\min}(1) + 1 \leq i \leq m_{\max}(1) + 1$ since the template can be placed right after every ending column of the first level. In Fig. 19, multiple rows show different template combinations. For the same end column, we keep only the best match. The rest of combinations are marked by crossbars and discarded.

The final solution can be selected from levels which contain template sequences ending at the last column (N_c) of the word bitmap (all boxes shown in Fig. 19). The best match is obtained as:

$$P^* = \max_{1 \leq l \leq l_{\max}} [P_l{}^*(N_c)].$$

By tracing back the best path ending at $i = N_c$, we will recover the sequence of template indices.

4.5.3 Modification for Kerning

So far we have assumed that templates are placed right after one another to build a template sequence. For the printed text, two characters could be spaced apart by a few white columns (*intercharacter spacing*) or overlap by a few columns (*kerning*). When we concatenate two templates, we have to take these facts into consideration.

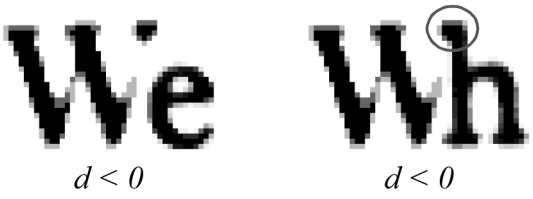


Fig. 20. (a) Ideal kerning case. (b) Overlapping black pixels.

Accordingly, we modified the level-building algorithm as follows: At each level, when we match a template to the word bitmap, we shift the template around the starting column within a small range. The range depends on the sizes of the two templates and on the average intercharacter space estimated in the prototype extraction procedure.

When two templates have overlapping black pixels, a *union template* is formed by taking the darker element value from the two templates. Fig. 20 shows both kerning and overlap. By allowing overlapping columns between two templates, WRAP can successfully handle not only kerning, but also heavily touching characters.

4.6 System Adaptation

Compared to conventional OCR systems, the main advantage of document-specific OCR is that it can be adapted to page quality and achieve higher accuracy. So, the system should have the ability to learn from the page presented to it. Its accuracy should increase on longer documents.

Usually, WRAP is trained with a portion of the document page to be recognized. If some rare characters do not appear in the training set, then we cannot obtain templates for their classes. The system could either reject the whole word (by evaluating the template matching score) and send it to an omnifont commercial OCR device or mark the doubtful words for manual correction. The words labeled by commercial OCR or by users can be used as new training data. New templates will be generated with the same method as the original templates and added to the template set.

Due to errors in the transcript, templates obtained from the original training samples may contain some wrong prototypes, even after the prototype selection. After recognition of some new text, the characters with high template matching probabilities can be reused to extract better prototypes. These additional prototypes are used to improve the templates.

5 Bootstrap Recognition Experiments and Results

Manual preparation of the transcript is practical only for a small data set. An alternative way to generate the transcript is to use a commercial OCR package. Here, we explore the idea of bootstrap recognition, i.e., using the imperfect commercial OCR output as the input to WRAP and adapting WRAP to achieve better recognition accuracy.

5.1 A Commercial OCR Development Package

We selected a commercial OCR development package which provides many sophisticated routines for users to build their own OCR system. These routines cover almost all aspects of an OCR task, from page image analysis to language analysis. The development package also provides a very good OCR engine for recognition.

We built an OCR system on top of this commercial development package. To distinguish the development system from WRAP, we call it *OCR-1*. OCR-1 is customized to provide information required by WRAP. The page segmentation routine in OCR-1 automatically detects the text regions in an input page image. Alternatively, users can define the text regions by drawing convex polygons on the displayed page image. OCR-1 then recognizes all text regions on the input page image. The output contains the following information.

Page Image Information: Page image size (height and width) and skew angle are detected by the system. It also provides an index of the text regions with the top-left-bottom-right coordinates of each region.

Font Information: The system detects and estimates font characteristics for every font it encounters. The font information contains: font name, style, average character width, pitch, average x-height, average height of uppercase characters, average height of lowercase characters with descender, and scale. If some information cannot be determined, then the system outputs a 0 instead.

Word Bounding Box: The left, top, right, and bottom coordinates of each word bitmap are available from the system, as well as the baseline location in each word. Individual character locations are not provided.

Word Label: Each word bounding box is labeled by the recognition result.

Character Confidence: The system assigns an integer $(0-999)$ to each character in the word label. The bigger the number, the more accurate the character label.

Word Confidence: The lowest character-confidence number within a word is assigned as the word confidence. Word confidence is also in (0-999).

5.2 Bootstrap Recognition Scheme

We applied the customized OCR-1 system described above to several page images. By examining the output, we observed that the word bounding-box information, including baseline location, is fairly accurate. Word-confidence numbers do reflect the recognition accuracy. But, the font information is neither accurate nor complete. For example, OCR-1 often mistakes proportional-pitch font for fixed-pitch font.

We designed the bootstrap recognition experiment as follows: A test page image is sent to OCR-1 for initial recognition. The results from OCR-1, including word bounding boxes, word labels, and word confidence numbers, are used by WRAP as inputs. Since WRAP does not use any lexical information, for a fair comparison we would like to turn off the lexicon of OCR-1 as well. But, the lexicon is built into the recognition engine of this development package and cannot be turned off.

The word labels from OCR-1, which include recognition errors, are used by WRAP as a transcript. Since the font information provided by OCR-1 is not very accurate, WRAP estimates character size with its own least-squares estimation routine.

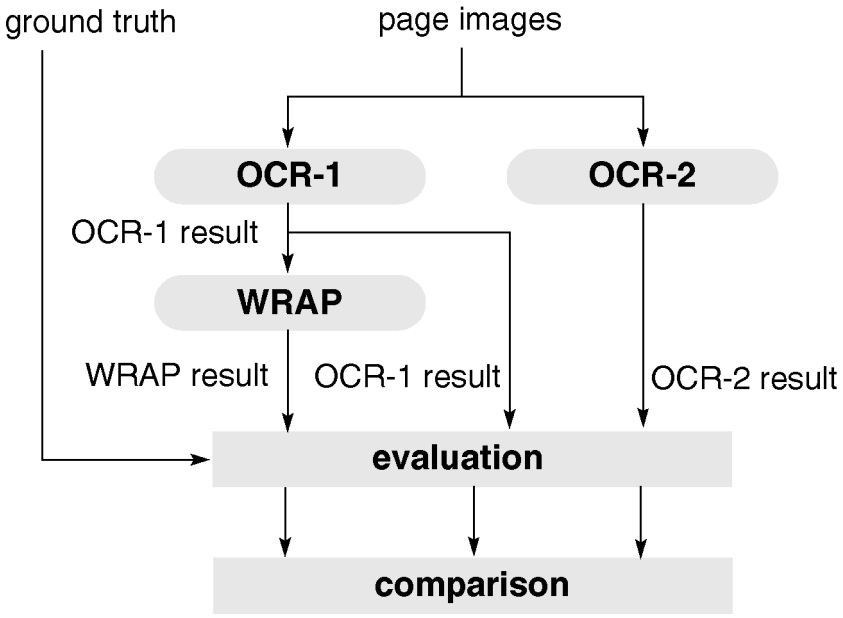


Fig. 21. The data flow of the bootstrap recognition experiment.

Taking the initial outputs from OCR-1, WRAP extracts prototypes from the same page image and adapts its template matching recognition engine. Then, WRAP is used to recognize the same page image on the expectation that the results will be better than those of OCR-1. We also tested another commercial OCR system on all page images for comparison. We call this second commercial OCR package *OCR-2*. The language analysis (dictionary) *can* be turned off in OCR-2. So, we can use OCR-2 to estimate the effects of language analysis on OCR.

We use the ground truth provided with the test pages to evaluate the recognition results from OCR-1, OCR-2, and WRAP. The evaluation routine was developed in DocLab, RPI [5], based on a string matching algorithm that calculates the edit distance between two character strings [21]. Fig. 21 shows the data flow in the bootstrap recognition experiment.

5.3 Data Preparation

We selected a dozen page images from the *DOE sample* of the ISRI database [18]. The *DOE sample* contains 785 pages from a large group of scientific and technical documents collected by the U.S. Department of Energy. When ISRI conducted the annual tests in 1992-1996, the median character accuracy achieved by a set of OCR packages was used as a "page quality" measurement. All test page images were divided into five approximately equal size groups by using this quality measure. Group 1 contains the pages with the highest median accuracy (best page quality), while Group 5 contains the pages with the lowest median accuracy (worst page quality). We selected the 12 test pages from Group 5.

Each of the 12 test page images contains a dominant font, i.e., only a few words, such as subtitles or italicized phrases, belong to different fonts or sizes. We manually block out halftone images and line drawing regions to send only text regions to the OCR systems for recognition. In each text region, words with different fonts or sizes are also blocked out.

Fig. 22 shows a sample test page image (file name: 5649-076). The lower part is printed at the original size, and shows the word bounding boxes and word confidences which are outputs from OCR-1. The word confidence numbers are divided into three levels: high (999-600), medium (600-300), and low (300-0). The position of a horizontal line within each bounding box shows the level. For example, the first word, 3, is recognized with medium accuracy, while the second, mm, with low accuracy.

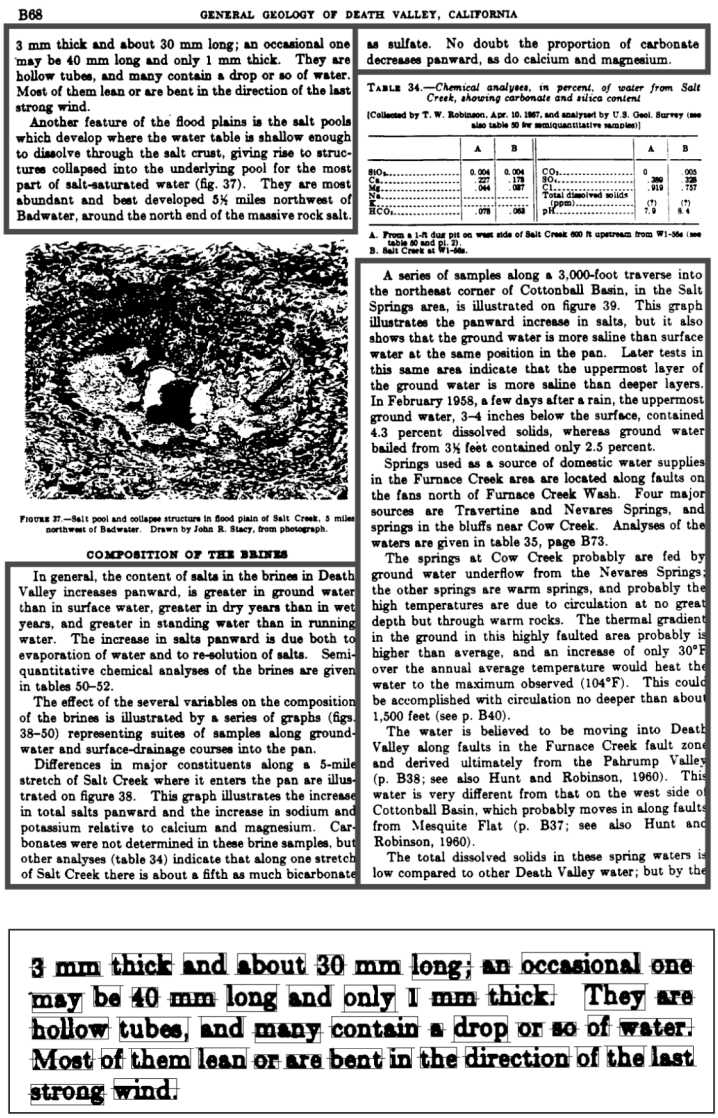
Among the 12 test page images, Set-A contains six pages from six different documents. Set-B includes six pages belonging to two documents, three pages from each. Multiple page images from the same document (Set-B) will be used to test how the training sample size affects the performance of an adaptive OCR system.

When WRAP extracts prototypes by using the transcript from OCR-1, the linked list of lexical analysis is sorted by word confidence number in order to extract prototypes from the most reliable initial word labels. In WRAP, only character widths are estimated from the training pages, no other parameters are adjusted prior to the recognition test.

5.4 Test Results and Comparison with Commercial OCR

To compare fairly the recognition results from different OCR packages, we should test all packages under the same test settings. But, the n-gram and language analysis cannot be turned off in OCR-1, OCR-2 can only turn off the language analysis, but still uses n-grams, and WRAP has no n-gram or language analysis at all.

To estimate how much an OCR package benefits from the language analysis, we tested several pages on OCR-2 with language analysis both on and off. The recognition error rate was reduced 4-6 times with the language analysis on.



B68 GENERAL GEOLOGY OF DEATH VALLEY, CALIFORNIA

3 mm thick and about 30 mm long; an occasional one may be 40 mm long and only 1 mm thick. They are hollow tubes, and many contain a drop or so of water. Most of them lean or are bent in the direction of the last strong wind.

Another feature of the flood plains is the salt pools which develop where the water table is shallow enough to dissolve through the salt crust, giving rise to structures collapsed into the underlying pool for the most part of salt-saturated water (fig. 37). They are most abundant and best developed 5½ miles northwest of Badwater, around the north end of the massive rock salt.

FIGURE 37.—Salt pool and collapse structure in flood plain of Salt Creek, 5 miles northwest of Badwater. Drawn by John R. Stacy, from photograph.

COMPOSITION OF THE BRINES

In general, the content of salts in the brines in Death Valley increases panward, is greater in ground water than in surface water, greater in dry years than in wet years, and greater in standing water than in running water. The increase in salts panward is due both to evaporation of water and to re-solution of salts. Semiquantitative chemical analyses of the brines are given in tables 50–52.

The effect of the several variables on the composition of the brines is illustrated by a series of graphs (figs. 38–50) representing suites of samples along groundwater and surface-drainage courses into the pan.

Differences in major constituents along a 5-mile stretch of Salt Creek where it enters the pan are illustrated on figure 38. This graph illustrates the increase in total salts panward and the increase in sodium and potassium relative to calcium and magnesium. Carbonates were not determined in these brine samples, but other analyses (table 34) indicate that along one stretch of Salt Creek there is about a fifth as much bicarbonate as sulfate. No doubt the proportion of carbonate decreases panward, as do calcium and magnesium.

TABLE 34.—*Chemical analyses, in percent, of water from Salt Creek, showing carbonate and silica content*

[Collected by T. W. Robinson, Apr. 10, 1957, and analyzed by U.S. Geol. Survey (see also table 50 for semiquantitative samples)]

	A	B		A	B
SiO_2	0.004	0.004	CO_3	0	.005
Ca	.227	.178	SO_4	.389	.228
Mg	.044	.087	Cl	.919	.737
Na			Total dissolved solids (ppm)	(?)	(?)
K			pH	7.9	8.4
HCO_3	.078	.053			

A. From a 1-ft dug pit on west side of Salt Creek 600 ft upstream from W1-36a (see table 50 and pl. 2).
B. Salt Creek at W1-36a.

A series of samples along a 3,000-foot traverse into the northeast corner of Cottonball Basin, in the Salt Springs area, is illustrated on figure 39. This graph illustrates the panward increase in salts, but it also shows that the ground water is more saline than surface water at the same position in the pan. Later tests in this same area indicate that the uppermost layer of the ground water is more saline than deeper layers. In February 1958, a few days after a rain, the uppermost ground water, 3–4 inches below the surface, contained 4.3 percent dissolved solids, whereas ground water bailed from 3½ feet contained only 2.5 percent.

Springs used as a source of domestic water supplies in the Furnace Creek area are located along faults on the fans north of Furnace Creek Wash. Four major sources are Travertine and Nevares Springs, and springs in the bluffs near Cow Creek. Analyses of the waters are given in table 35, page B73.

The springs at Cow Creek probably are fed by ground water underflow from the Nevares Springs; the other springs are warm springs, and probably the high temperatures are due to circulation at no great depth but through warm rocks. The thermal gradient in the ground in this highly faulted area probably is higher than average, and an increase of only 30°F over the annual average temperature would heat the water to the maximum observed (104°F). This could be accomplished with circulation no deeper than about 1,500 feet (see p. B40).

The water is believed to be moving into Death Valley along faults in the Furnace Creek fault zone and derived ultimately from the Pahrump Valley (p. B38; see also Hunt and Robinson, 1960). This water is very different from that on the west side of Cottonball Basin, which probably moves in along faults from Mesquite Flat (p. B37; see also Hunt and Robinson, 1960).

The total dissolved solids in these spring waters is low compared to other Death Valley water; but by the

3 mm thick and about 30 mm long; an occasional one may be 40 mm long and only 1 mm thick. They are hollow tubes, and many contain a drop or so of water. Most of them lean or are bent in the direction of the last strong wind.

Fig. 22. Page image 5649-076 from DOE Group-5 sample.

We also constructed a special page image to test OCR-1. We composed the page with the ground truth of a real document page, but the character order within each word was reversed. This test page was printed by a good printer and scanned in at 300 dpi. Usually, OCR-1 has very low error rate for well-printed clean pages. But, for this page, we observed many additional errors due to the lexical or language analysis. For example, OCR-1 often recognized fo, which is the reverse of the word of, as to since fo is apparently not a dictionary word.

Sziranyi and Boroczki used a factor of 4 to scale the recognition errors (after spelling correction) to the errors before the spelling correction [19]. To estimate what the error rates would be with a dictionary in our experiment, we scale the results from WRAP and OCR-2 by dividing the error rates by a factor of 4 to approximate the possible lexical improvement. For WRAP, a factor of 4 is conservative since there is also no n-gram information used in WRAP. OCR-1 uses both n-gram and language analysis, so we do not scale its results.

We call the characters from the classes that we have templates *admissible characters*. The last column is the error rate scaled from the gross character error rate. The experimental results for six different source pages in Set-A, six pages from the two multipage documents (Set-B), and the overall results are shown in Table 1. For multipage documents in Set-B, WRAP was trained by the OCR-1 results from the first page of each document.

5.5 Discussion

We observe that OCR-1 has the lowest error rate on almost every test page. But, if we take the lexical and language analysis effects into account, the scaled results from WRAP are better than from both OCR-1 and OCR-2 on most of the test pages. In Fig. 23, we plot character error rates from WRAP and OCR-1. We can see that most of the scaled error rates from WRAP are lower than those of the initial OCR-1. So, training the OCR system for a specific page image, especially a degraded page image, can help to improve the recognition accuracy.

TABLE 1
Bootstrap Recognition Results (Summary)

Test Data				Results		
ISRI File	OCR ID	# adm. char.	# char.	adm. char. errors %	char. errors %	scaled errors %
Set-A	OCR-1	19712	20084	3.8	4.0	4.0
	WRAP	19712	20084	7.5	8.3	2.1
	OCR-2	19712	20084	12.7	12.9	3.2
Set-B	OCR-1	8008	8566	3.3	3.8	3.8
	WRAP	8008	8566	6.7	9.8	2.5
	OCR-2	8008	8566	13.7	15.4	3.6
Total 12 pages	OCR-1	27720	28650	3.7	3.9	3.9
	WRAP	27720	28650	7.3	8.6	2.2
	OCR-2	27720	28650	13.0	13.3	3.3

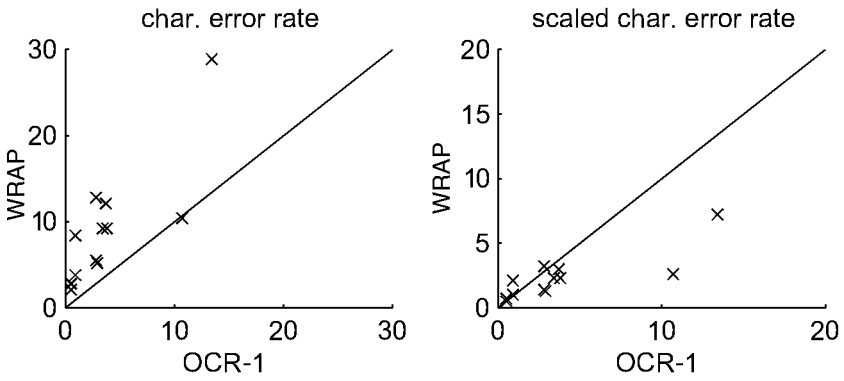


Fig. 23. Error rate comparison between WRAP and OCR-1.

The WRAP system also has better performance than OCR-2 (Fig. 24), although OCR-2 still uses n-grams in its recognition engine (but no lexicon).

We also observe that, on pages from Set-B (`ISRI-5842`), WRAP has a high error rate. As mentioned, we trained WRAP on the first page (`5842-061`) which contains only 587 characters. To show how WRAP can adapt itself to improve the recognition result when more samples are available, we add the second page (`5842-099`) with training samples and train WRAP with two pages. The results are shown in Table 2. There are more admissible characters and the errors are reduced considerably. The longer the document, the more and better templates we will get. We expect the results to eventually approach what we would obtain if we used the entire set for extracting the templates under supervised training. All three OCR systems have high error rates on the third page from this batch (`5842-285`). We observed that this page has many more broken characters than the other two, although they are all selected from the same article. WRAP achieves higher recognition accuracy than OCR-1 and OCR-2 on this page because WRAP is trained on a similar page image.

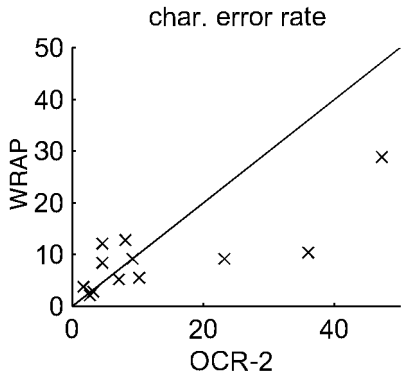


Fig. 24. Error rate comparison between WRAP and OCR-2.

TABLE 2
WRAP Recognition Results with Adaptation

Test Data		WRAP Results (trained on 1 page)			WRAP Results (trained on 2 pages)		
ISRI File	# char.	# adm. char.	char. errors %	adm. char. errors	# adm. char.	char. errors %	adm. char. errors
5842-061	587	482	12.1	3.5	523	8.3	3.4
5842-099	1063	978	8.4	5.3	1027	5.0	3.5
5842-285	767	620	28.9	24.4	672	21.8	18.6

6 Conclusion

We have confirmed that OCR accuracy can be improved through recognition based on representative character prototypes and we have demonstrated a robust method to automatically extract the required prototype bitmaps from each new page or document. The method depends on the availability of a partially correct transcript, which was obtained here from a commercial omnifont OCR system. We may extend our method to use WRAP's high confidence character decisions to retrain the classifier for better recognition results. The recursive approach proved valuable in [14], [2], but we would need a faster implementation.

We attribute the increased accuracy to capturing both the shape associated with the new typeface and the characteristics of the degradation. The underlying idea is similar to that of page recognition based on the Hidden Markov Model, but our character-based model leads to conceptual and implementational simplification compared to HMM.

In order to exploit this method in an operational environment, it must be augmented with a reliable font-change detector. Such a detector may be as simple as a threshold on the cumulative mismatch between the word bitmaps and the best template sequences. Also required for most applications is some form of contextual correction based on linguistic properties.

Acknowledgments

The authors gratefully acknowledge the support of the Central Research Laboratory, Hitachi, Ltd., and of the Nortel Educational Research Network. This work has been conducted in the New York State Center for Automation Technology (CAT), which is partially funded by a block grant from the New York State Science and Technology Foundation.

References

[1] H.S. Baird, "The Skew Angle of Printed Documents," *Proc. Conf. Photographic Scientists and Engineers,* pp. 14-21, 1987.
[2] H.S. Baird and G. Nagy, "A Self-Correcting 100-Font Classifier," *Proc. SPIE,* vol. 2,181, pp. 106-115, 1994.
[3] R. Duda and P.E. Hart, *Pattern Classification and Scene Analysis.* John Wiley & Sons, 1973.
[4] R. Esposito, D. Malerba, and G. Semeraro, "An Experimental Page Layout Recognition System for Office Document Automatic Classification: An Integrated Approach for Inductive Generalization," *Proc. 10th Int'l Conf. Pattern Recognition (ICPR),* pp. 557-562, 1990.
[5] A. El-Nasan, "InkLink—An Unconstrainted-Handwriting Recognition Engine," technical report, DocLab, RPI, 1998.
[6] H.S. Heaps, *Information Retrieval: Computational and Theoretical Aspects.* New York: Academic Press, 1978.
[7] T. Hong and J. Hull, "Character Segmentation Using Visual Inter-Word Constraints in a Text Page," *SPIE,* vol. 2,422, pp. 15-25, 1995.
[8] R. Ingold, "Structure de Documents et Lecture Optique: Une Nouvell Approche," doctoral dissertation, Ecole Polytechnique Federale de Lausanne, Presses Polytechniques Romandes, Lausanne, Switzerland, 1994.
[9] G.E. Kopec and P.A. Chou, "Document Image Decoding Using Markov Source Models," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, no. 6, pp. 602-617, June 1994.
[10] G.E. Kopec and M. Lomelin, "Document-Specific Character Template Estimation," *Proc. SPIE,* vol. 2660, pp. 14-26, 1996.
[11] G.E. Kopec, "Least-Squares Font Metric Estimation from Images," *IEEE Trans. Image Processing,* vol. 2, no. 4, pp. 510-519, 1993.
[12] H. Kucera and W.N. Francis, *Computational Analysis of Present-Day American English.* Brown Univ. Press, 1967.
[13] C.L. Lawson and R.J. Hanson, *Solving Least Squares Problems.* Prentice Hall, 1974.
[14] G. Nagy and G.L. Shelton, "Self-Corrective Character Recognition System," *IEEE Trans. Information Theory,* vol. 12, no. 2, pp. 215-222, Apr. 1966.
[15] G. Nagy and Y. Xu, "Priming the Recognizer," *Proc. IAPR Workshop Document Analysis Systems,* pp. 263-281, 1996.
[16] G. Nagy and Y. Xu, "Automatic Prototype Extraction for Adaptive OCR," *Proc. Fourth Int'l Conf. Document Analysis and Recognition,* pp. 278-282, 1997.
[17] G. Nagy and Y. Xu, "Bayesian Subsequence Matching and Segmentation," *Pattern Recognition Letters,* vol. 18, pp. 1,117-1,124, 1997.
[18] S.V. Rice, F.R. Jenkins, and T.A. Nartker, "The Fifth Annual Test of OCR Accuracy," *UNLV Information Science Research Inst. 1996 Ann. Report,* 1993-1996.
[19] T. Sziranyi and A. Boroczki, "Overall Picture Degradation Error for Scanned Images and the Efficiency of Character Recognition," *Optical Eng.,* vol. 30, no. 12, pp. 1,878-1,885, 1991.
[20] L. Spitz, "An OCR Based on Character Shape Codes and Lexical Information," *Proc. Third Int'l Conf. Document Analysis and Recognition,* pp. 723-728, 1995.
[21] R. Valdes, "Finding String Distances," *Dr. Dobb's J.,* pp. 56-62, Apr. 1992.
[22] I.H. Witten, A. Moffat, and T.C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images.* Van Nostrand Reinhold, 1994.

Yihong Xu received the BS degree in electrical engineering from Xi'an Jiaotong University, Xi'an, China, the MS degree in computer science from the University of Science and Technology of China, Hefei, China, and the PhD degree in computer engineering from Rensselaer Polytechnic Institute, Troy, New York. She is currently a member of the technical staff at Hewlett-Packard Laboratories. Her research interests include document image analysis, pattern recognition, document management, internet imaging, image processing, and multimedia security. She is a member of the IEEE and the IEEE Computer Society.

George Nagy received the BEng and MEng degrees from McGill University and the PhD degree in electrical engineering from Cornell University in 1962 (on neural networks). For the next 10 years, he conducted research on pattern recognition and OCR at the IBM T.J. Watson Research Center in Yorktown Heights, New York. From 1972 to 1985, he was a professor of computer science at the University of Nebraska-Lincoln and worked on remote sensing applications, geographic information systems, computational geometry, and human-computer computer interfaces. Since 1985, he has been a professor of computer engineering at Rensselaer Polytechnic Institute, Troy, New York. He has held numerous visiting appointments in the U.S. and abroad. In addition to document image analysis and character recognition, his interests include spatial modeling, finite-precision spatial computation, and computer vision. He is coauthor of the 1999 monograph, *OPTICAL CHARACTER RECOGNITION: An Illustrated Guide to the Frontier*. He is a senior member of the IEEE and a member of the IEEE Computer Society.