Recognition of Merged Characters Based on Forepart Prediction, Necessity-Sufficiency Matching, and Character-Adaptive Masking

Jiqiang Song, *Member, IEEE*, Zuo Li, Michael R. Lyu, *Fellow, IEEE*, and Shijie Cai

***Abstract*—Merged characters are the major cause of recognition errors. We classify the merging relationship between two involved characters into three types: "*linear*," "*nonlinear*," and "*overlapped*." Most segmentation methods handle the first type well, however, their capabilities of handling the other two types are limited. The weakness of handling the *nonlinear* and *overlapped* types results from character segmentation by linear, usually vertical, cuts assumed in these methods. This paper proposes a novel merged character segmentation and recognition method based on forepart prediction, necessity-sufficiency matching and character-adaptive masking. This method utilizes the information obtained from the forepart of merged characters to predict candidates for the leftmost character, and then applies character-adaptive masking and character recognition to verifying the prediction. Therefore, the arbitrary-shaped cutting path will follow the right shape of the leftmost character so as to preserve the shape of the next character. This method handles the first two types well and greatly improves the segmentation accuracy of the *overlapped* type. The experimental results and the performance comparisons with other methods demonstrate the effectiveness of the proposed method.**

***Index Terms*—Character-adaptive masking, character feature extraction, character segmentation, forepart prediction, merged character recognition, necessity-sufficiency matching.**

I. Introduction

OPTICAL character recognition (OCR) is one of the most successful applications of automatic pattern recognition. Nowadays, the research interests of OCR focuses on the recognition of degraded, multiple-font machine-printed, and handwritten text [1]. For printed text, the main difficulty comes from the severely merged or degraded characters. Until now, incorrect recognition of merged characters is still one of the main causes for recognition errors. The well-known tests of commercial printed-text OCR systems by the University of Nevada, Las Vegas [2], show that even when perfect patterns were recognized, commercial systems experience 0.5% spacing errors in average. This is essentially a segmentation error by a process that attempts to isolate word images [3]. Furthermore, since segmentation errors often raise chain effects, the performance of segmentation is crucial for the whole OCR process.

This paper proposes an effective merged-character recognition method, which consists of a novel segmentation scheme based on forepart prediction (FP), character-adaptive masking (CAM), and a necessity-sufficiency matching (NSM) algorithm for single character recognition. First, the method predicts the candidates for the foremost (for horizontally aligned text, "foremost" equals "leftmost") character of merged characters based on a set of forepart features and then cuts it from the image using its individual mask, namely, character-adaptive masking. Next, the recognizer determines the probability of each candidate and decides which candidates are acceptable, and each accepted candidate will start a new processing branch for the latter segmentation; consequently, there will be multiple segmentation solutions. Finally, the solution with the highest holistic recognition probability will be accepted. Compared with existing merged-character recognition approaches, the advantages of the proposed method are the following:

1) forepart prediction is verified by reliable character matching;
2) NSM algorithm utilizes a concise feature-row-based character model to distinguish similar characters efficiently;
3) character masks for the segmentation are adaptive to character shapes so as to avoid damaging the character image.

The rest of this paper is organized as follows. Section II classifies the merging types and reviews the related work. Section III describes the proposed algorithms in detail. The experimental results and comparisons with other methods are reported in Section IV. Finally, Section V draws our conclusions.

II. Related Work

The recognition methods of merged characters can first be divided into two classes: one with segmentation, and one without. The methods without segmentation recognize characters from a text image directly. Most segmentation-free methods are lexicon-based. They treat the word as a single, indivisible entity, and attempt to recognize it using features of the word as a whole. Thus, they are usually called holistic methods. Madhvanath and Govindaraju [4] conduct a survey on holistic methods and summarize "their treatment of lexicon words as distinct pattern classes has traditionally limited their application to recognition scenarios involving small, static

Manuscript received October 19, 2003; revised May 7, 2004. This work was supported in part by the Yaulee Construction Co. Ltd., Hong Kong, and the Research Grants Council of the Hong Kong Special Administrative Region, China under Grant CUHK 4182/03E. This paper was recommended by Guest Editor V. Govindaraju.

J. Song and M. R. Lyu are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: songjq@ieee.org; lyu@cse.cuhk.edu.hk).

Z. Li and S. Cai are with the State Kay Laboratory of Novel Software Technology, Nanjing University, Nanjing 210093, China (e-mail: lizuo@graphics.nju.edu.cn; sjcai@netra.nju.edu.cn).

Digital Object Identifier 10.1109/TSMCB.2004.837588

lexicons." Rocha and Pavlidis [5] propose an interesting segmentation-free approach, which converts a text image into a feature graph and then attempts to match subgraphs of features with predefined character prototypes. Different alternatives are represented by a net whose nodes correspond to the matched subgraphs. A final search for the optimal path in the net gives the best interpretation of the text image. Martin *et al.* [6] engage in an input window to scan horizontally over the text image, and train a neural network to recognize whether the window is centered over a single character (if so, the character is recognized) or between characters. Lee and Kim [7] improve this idea by employing the cascade neural network to retain the spatial relationship of connected characters.

Generally, most character recognition methods [8]–[23] include a segmentation process, i.e., first segmenting a given text image into character images and then recognizing each character image separately. OCR methods for the single character recognition have been well studied in the literature [24]. The current directions to improve the OCR performance include designing better classifiers [25], [26], integrating several existing classifiers to utilize various kinds of features [27], and extracting more robust features [28]–[30]. This paper proposes a new feature extraction and matching approach based on FP and NSM. Jung *et al.* [19] also explores the leftmost and rightmost features to recognize merged characters; however, they use linear segmentation, while the proposed method conducts nonlinear segmentation.

Prior to discussing the segmentation approaches, we define the three types of merging relationships: "*linear*," "*nonlinear*," and "*overlapped*" (Fig. 1). In Fig. 1(a), the two merged characters can be completely separated by a linear segmentation path, defined as the "*linear*" type. Here the "completely separated" means the entirety of both characters is maintained. In Fig. 1(b), the two merged characters cannot be completely separated by any linear path, but can be done by a nonlinear path, defined as the "*nonlinear*" type. In Fig. 1(c), the two merged characters overlap each other; therefore, they cannot be completely separated by either a linear path or a nonlinear path, defined as the "*overlapped*" type. Following the definition of merging types, we classify a character segmentation approach into either a linear segmentation approach or a nonlinear segmentation approach, according to the shape of its segmentation path.

Linear segmentation approaches separate merged characters by linear cuts, mostly vertical cuts. They depend on a variety of clues to locate the cutting positions. The most simple and fast way is to analyze the vertical projection of a black region. The representative methods are [8], [9], which locate the valleys of the projection profile as the cutting positions. However, they cannot handle the *linear* type with long vertical connected length [Fig. 2(a)], degraded characters, or italic characters. Tsujimoto and Asada [10] proposed a filter to produce a deeper valley at merging positions using an "AND" operation of adjacent columns, but the improvement is limited. Another popular way is recognition-based segmentation, which generates multiple segmentation hypotheses and then chooses the best one by recognition. The approach of Casey and Nagy [14] recognizes the subimage in a shrinking window whose width decreases leftward [Fig. 2(b)]. Once a character is matched, its corresponding block will be cut and the remainders will be processed recursively. This strategy is better than projection-based methods because it selects cutting positions dynamically, but it also cannot handle the *nonlinear* type due to the vertical cutting. Other recognition-based segmentation approaches [15], [19] propose different criteria to select cutting positions, however, they still perform vertical cutting. To decrease the segmentation errors caused by vertical cuts, Bayer *et al.* [11], [12] place a cut classifier before cutting, and perform additional connectivity analysis after cutting to correct the errors. Some connected-component based approaches [13], [18], [22] employ a splitting-and-merging scheme. They first separate the text image into primitive blocks and then combine consecutive blocks into candidate characters using certain criteria, such as knowledge-based dynamic programming [18] and lexicon matching [22]. Recently, Garain and Chaudhuri [23] propose a new approach applying fuzzy multifactorial analysis to identify touching parts and segment merged characters by vertical cuts.

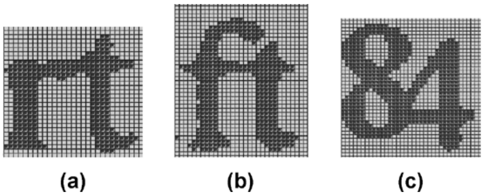


Fig. 1. Three merging types. (a) Linear. (b) Nonlinear. (c) Overlapped.

Nonlinear segmentation approaches [16], [17], [20], [21] attempt to separate merged characters with various merging types by the optimal nonlinear cutting paths. Wang and Jean [16] propose an approach called "shortest path segmentation." By predefining the "distance (cost)" needed for selecting the path of each hypothetic cut and then finding the "shortest" one, this method selects the optimal segmentation from all "legal" ones. This approach partially overcomes the problems raised by vertical cutting, and in a way, can adjust the path from vertical cutting. The shortest path consists of several vertical monochromic runs, as shown in Fig. 3. However, since the adjustment depends only on the local connectivity analysis, it cannot guarantee that the shortest path preserves the character's entirety well. Lee *et al.* [17] extend the shortest path search to the grayscale image space. Arica and Yarman-Vural [21] define a novel cost function using the information extracted from both grayscale image and binary image, and employ a dynamic programming algorithm to search the shortest path. The above three approaches share the same concept, i.e., searching the shortest path based on the pixel-wise cost accumulation. Chang and Chen [31] introduce the convex-hull information to improve the accuracy of shortest path location. Chen and Wang [20] further propose a different approach which first performs thinning on both foreground and background regions for the feature points extraction on the foreground and background skeletons, then constructs several segmentation paths from these feature points, and finally decides the best segmentation path or rejects the segmentation by the mixture Gaussian probability function.

In summary, existing nonlinear segmentation approaches all depend on local image features to determine segmentation paths, giving no guarantee to maintaining characters' entirety.

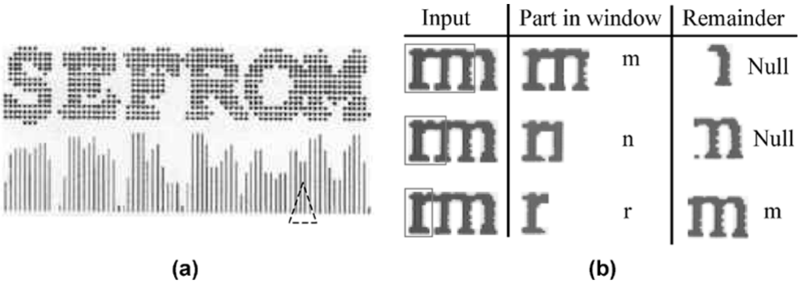


Fig. 2. Various segmentation methods. (a) Projection-based segmentation. (b) Recognition-based segmentation.

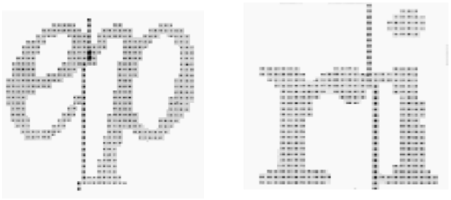


Fig. 3. Shortest path segmentation.

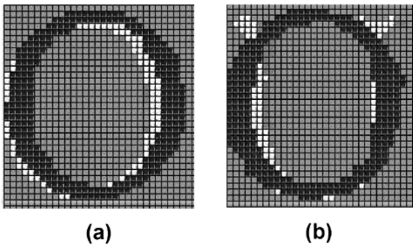


Fig. 4. Examples of *DP*s. (a) *Edge_DP* $M =$ "O;" $X =$ "O;" (b) *Stroke_DP* $M =$ "O;" $X =$ "U."

The proposed method differs from existing approaches in that the nonlinear segmentation path is formed by character-adaptive masking. We describe our approach in the next section in detail.

III. Algorithms

Correct recognition of merged characters doubtlessly relies on the accuracy of segmentation, which in turn needs image analysis and recognition to avoid the blindness of segmentation. We believe that the segmentation integrating both image analysis and recognition is the most promising way. Considering a horizontal string of merged characters, its leftmost part, i.e., the forepart of the first character, keeps the original shape. Thus, if we can 1) predict a group of candidates by analyzing the forepart, 2) verify the candidates by character recognition, and finally 3) cut out the subimage of the recognized character without damaging the succeeding character, then the entire string of merged characters can be recognized recursively. This is the fundamental idea of our proposed method.

The three conditions reveal the three key techniques of our method: FP-based prediction, NSM-based character recognition, and character-adaptive masking. This section will explain our method in a bottom-up order. Since the character recognition is an elementary process, it will be described first. The prediction and masking algorithms are presented next. Finally, we illustrate the overall paradigm of the merged characters recognition.

A. Character Recognition

Being an elementary process, the character recognition will be invoked very frequently; therefore, it should be time-efficient and accurate. The traditional character bitmap model cannot satisfy this requirement. Letting the prototype binary bitmap of a character be M and the input binary bitmap of a character be X, the resulting binary bitmap of matching M and X is calculated by $R = M \oplus X$. The nonzero points in R are called *different points* (*DP*). A *DP* caused by degradation or displacement of the same stroke is called *Edge_DP*, as shown in Fig. 4(a). A *DP* caused by different strokes is called *Stroke_DP*, as shown in Fig. 4(b). Since the character recognition is normally to detect *Stroke_DP*s and tolerate *Edge_DP*s, judging only by the quantity of *DP* is not robust for degradation and displacement; for example, Fig. 4(a) and Fig. 4(b) contain a similar number of *DP*s. On the other hand, distinguishing *Edge_DP* from *Stroke_DP* requires more complex global analysis. Therefore, the entire bitmap model is not a good choice. Moreover, the entire bitmap model contains much redundancy whose contribution to the recognition is less than the noise it introduces. Thus, we design a new compact feature extraction and matching algorithm.

1) Feature matching algorithm: To reduce the processing time and to achieve the maximum marginal utility of character features, the number of features should be the minimal that can distinguish English characters and punctuations. Since the bitmap data is organized row by row, computation in the row direction is fast. Therefore, we define the character feature as a group of rows in the prototype bitmap of a character, called *feature rows*, denoted by f.

Theoretically, the feature matching function should check whether each feature row and its corresponding row in an input bitmap are identical, as described in Definitions 1 and 2.

Definition 1: Let $\mathbf{MATCH}(f, X)$ be the matching function of f and an input bitmap X. If they match, the function returns 1; otherwise, it returns 0.

Definition 2: Let M_i be a row in the prototype bitmap M of a character, where i is the row index. Let M' be the prototype bitmap of another character. A group of rows of M, denoted by $f_{\mathrm{m}} = \{M_i | i = r_1, r_2 \ldots r_k\}$, can be the feature rows of this character on the condition that

$$\mathrm{MATCH}(f_m, M') = \begin{cases} 1, & \text{if and only if } M = M' \\ 0, & M \neq M'. \end{cases} \quad (1)$$

However, since the scanned image often contains degradation or displacement, the condition in (1) is too strict. Consequently, we propose the necessity-sufficiency matching algorithm based on *dilated feature rows* and *contracted feature rows*.

Definition 3: Let $f_t(t = 1 \ldots k)$ be a feature row consisting of several consecutive black and white runs. Then, the dilated feature row of f_t, denoted by f_t^d, is formed by adding two more black points to both the head and the tail of each run. Similarly, the contracted feature row of f_t, denoted by f_t^c, is formed by removing two head points and two tail points from each run; however, each run has to retain at least one point. Thus, $f = \{(f_t, f_t^c, f_t^d) | t = 1 \ldots k\}$.

With Definition 3, if X resembles M, X should cover $f_t^c(t = 1 \ldots k)$ completely and be covered by $f_t^d(t = 1 \ldots k)$ completely. Suppose that for each character, its prototype bitmap and the input bitmap have been normalized to the same width (W) and height (H). The necessity matching condition is defined as (2), where $row(t)$ returns the row index of t in the bitmap

$$N(f,X) = \sum_t N_t(f_t, X) = 0$$
$$\text{where } N_t(f_t,X) = \sum_{j=1}^{W} \left[\left(f_{tj}^c \wedge X_{\mathrm{row(t)j}} \right) \oplus f_{tj}^c \right]. \quad (2)$$

And, the sufficiency matching condition is defined as (3)

$$S(f,X) = \sum_t S_t(f_t, X) = 0$$
$$\text{where } S_t(f_t,X) = \sum_{j=1}^{W} \left[\left(X_{\mathrm{row(t)j}} \wedge f_{tj}^d \right) \oplus X_{\mathrm{row(t)j}} \right]. \quad (3)$$

Thus, we can define the necessity-sufficiency matching function as follows:

$$\mathrm{MATCH}(f,X) = \begin{cases} 1, & \text{if } N(f,X) = 0 \text{ and } S(f,X) = 0 \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

Equation (4) judges whether X matches the feature rows of a character. The binary output is adequate for selecting feature rows based on Definition 2, but it cannot indicate to which extent X resembles the character. Considering this information is important for the segmentation to make decision when multiple solutions exist, we modify the above equations to yield the probability of X being the specific character. Equations (2)–(4) are modified to be (5)–(7), respectively.

$$N'(f,X) = \prod_t \left[1 - N'_t(f_t,X)\right], \quad \text{where}$$
$$N'_t(f_t,X) = \frac{1}{W} \cdot \sum_{j=1}^{W} \left[\left(f_{tj}^c \wedge X_{row(t)j} \right) \oplus f_{tj}^c \right] \quad (5)$$
$$S'(f,X) = \prod_t \left[1 - S'_t(f_t,X)\right], \quad \text{where}$$
$$S'_t(f_t,X) = \frac{1}{W} \cdot \sum_{j=1}^{W} \left[\left(X_{row(t)j} \wedge f_{tj}^d \right) \oplus X_{row(t)j} \right] \quad (6)$$
$$\mathbf{MATCH}'(f,X) = N'(f,X) \times S'(f,X),$$
$$0 \le N'(f,X) \le 1 \text{ and}$$
$$0 \le S'(f,X) \le 1. \quad (7)$$

Thus, (2)–(4) form the NSM algorithm for selecting feature rows, whereas (5)–(7) form the NSM algorithm for character recognition. An input bitmap X can be recognized as the character with the maximum matching probability.

The normal, dilated and contracted feature rows can tolerate horizontal displacement, but they are font-dependent. This design is based on the consideration that it will enhance the capability of distinguishing similar characters.

2) Selection of feature rows: The process of selecting feature rows for a character consists of two steps 1) preparing the prototype bitmap for this character, and 2) automatically generating candidate feature.

The feature rows are independent from font size since both the prototype bitmap and the input bitmap are normalized. Since they are dependent on font style, we generate separate feature libraries for ten commonly-used fonts. This fashion is suitable for our application—tenders, which require specific fonts in specific parts, so we can preload the feature library of the specific font by tender layout analysis. In each feature library, the prototype bitmap of every character has three variants for regular, bold, and italic styles, respectively.

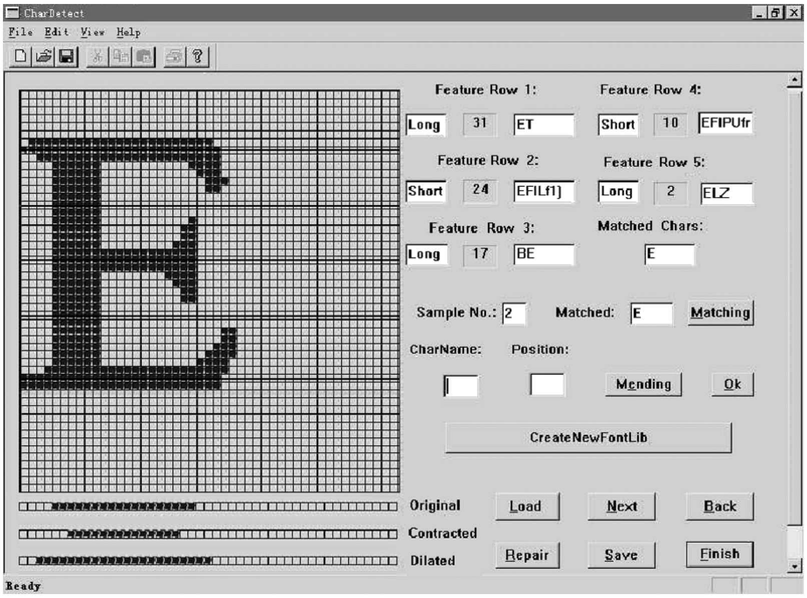
With the prototype bitmaps of all characters ready, the feature rows of each character are automatically selected by the following four steps. For the current character, denote the prototype bitmap by M, the target feature row set by f_r, and the candidate row set by C_r.

1) Initially, $f_r = \{\}$, and C_r contains all rows in M;
2) For each row r in C_r, first make $f'_r(r) = f_r + \{r\}$, then count the times of f'_r matching other prototype bitmaps than M, denoted by M', in the same feature library: $M\_\mathrm{SUM}(\mathrm{r}) = \Sigma\mathbf{MATCH}(\mathrm{f}'_\mathrm{r}(\mathrm{r}), \mathrm{M}')$, finally select the $r_{\min}$ with the minimum $M\_\mathrm{SUM}(r)$. If more than one r reach a tie for the $r_{\min}$, select the one with the maximum vertical distance to the existing feature rows in f_r to distribute the feature rows evenly;
3) Move $r_{\min}$ from C_r to f_r;
4) If $M\_SUM(r_{\min}) = 0$, the current f_r is the feature row set of M, then terminate. Otherwise, go to (2).

The automatic feature row selection is offline and one-off; therefore, the time efficiency is not critical. The number of feature rows for each character is different. According to our experiment, the number is at most five. Most characters have four or five feature rows.

Fig. 5 shows a tool for monitoring the feature rows of a selected character. The left side shows the prototype bitmap and the feature rows. The current feature row together with its contracted and dilated feature rows is drawn at the bottom. The right side displays the matched characters of each feature row, and the box "matched chars" gives the characters matching all feature rows. Only the ground-truth character should appear in this box.

An important issue before matching feature rows is the alignment between the input bitmap and the feature rows, especially the vertical alignment since the NSM algorithm can tolerate small horizontal offsets. There are two common alignment methods: bounding-box alignment and gravity-center alignment. The former is simple but sensitive to noise, while the latter is antinoise but computation-expensive. We utilize the



Feature Row 1: Long 31 ET

Feature Row 4: Short 10 EFIPUfr

Feature Row 2: Short 24 EFILf1]

Feature Row 5: Long 2 ELZ

Feature Row 3: Long 17 BE

Matched Chars: E

Sample No.: 2 Matched: E Matching

CharName: Position:

Fig. 5. Feature rows.

baseline of printed text line (Fig. 6) for alignment. The baseline is defined at the bottom of upper case characters. Since the baseline is obtained from a text line, it is more antinoise than the gravity center, and it is fast. Therefore, the baseline position of each character is also labeled in the feature library.

B. FP-based segmentation

The FP-based segmentation algorithm consists of two steps: forepart prediction and character-adaptive masking. The former selects a group of candidates by forepart analysis, while the latter screens out the bitmap using the nonrectangular mask adaptive to each candidate and then recognizes the segmented bitmap by the NSM algorithm. Therefore, the merged characters can be segmented with the highest recognition probability.

1) Forepart prediction: After horizontally scanning a document image, we obtain the baseline position and the height of each text line, which is the vertical distance between the baseline and the top of the text line. Denoting the height of text line by H_l, the term "forepart" means the leftmost $H_l/4$ wide part of the input image of merged characters. For a prototype bitmap, it is nearly the left half of the bitmap. The forepart predication is based on three reliable forepart features, i.e., baseline-related feature, forepart height feature, and forepart boundary feature.

The baseline-related feature indicates whether the forepart of a character has components under the baseline, which takes value "*true*" or "*false*." The forepart height feature indicates whether the forepart occupies the full height above the baseline, which also takes value "*true*" or "*false*." The forepart boundary feature is a little more complex, defined as follows:

$$B = \{B(i) | i \text{ is the row index of a bitmap} \\ \text{and baseline} \leq i \leq \text{baseline} + H_l\}. \quad (8)$$

$B(i)$ is the number of white points before the first black point in the row i. Considering the input scanned bitmap may contain noises, it can be revised to be the number of points before the first black segment whose width is not less than the minimum stroke width. The difference between two forepart boundary features B_1 and B_2 is defined as $|B_1 - B_2| = \underset{\text{baseline} \leq \text{i} \leq \text{baseline}+\text{H}_\text{l}}{\text{MAX}} [|B_1(i) - B_2(i)|]$.

Billing system
Baseline

Fig. 6. Baseline of a text line.

These three features of each character are obtained from its prototype bitmap and then stored in the feature library. Denote the baseline-related feature by L, the forepart height feature by G, and the forepart boundary feature by B. Correspondingly, these three features obtained from the forepart of an input bitmap are denoted by L', G' and B', respectively. Letting the initial candidate set C_0 containing all models in a feature library, we can select candidates by the following operations:

$$C_1 = \{c \in C_0 | L = L'\}$$
$$C_2 = \{c \in C_1 | G = G'\}$$
$$C_3 = \{c \in C_2 | |B - B'| < \varepsilon\}.$$

The dimension of C_3 is much smaller than that of C_0, however, candidates like "P" and "K" cannot be distinguished. To further reduce the number of candidates, for each candidate in C_3, we perform a quick necessity matching by (2) to form C_4, which is the final candidate set produced by the forepart prediction.

$$C_4 = \{c \in C_3 | N(f_c, X) = 0\}.$$

2) Character-adaptive masking: The forepart prediction produces a small number of candidates. The best way to verify each candidate is to screen out the bitmap of the leftmost character and recognize it. To avoid the disadvantages of vertical cutting, we propose a character-adaptive masking algorithm, which uses an individual mask for each character. The mask is generated from the prototype bitmap of the character. Before defining the mask, we first define the right boundary of the character. Recall the width and height of the prototype bitmap are W and H, respectively. The right boundary is defined as follows:

$$RB = \{RB(i) \mid i \text{ is the row index of a bitmap, and } 0 \leq i \leq H\}$$

where $RB(i)$ is the horizontal coordinate of the rightmost black point on the row i. If the entire row is white, $RB(i) = 0$. Then, the real width of the character $W_c = \underset{0 \leq i \leq H}{MAX}[RB(i)]$. Now we can define the character mask.

Definition 4: The mask is a binary matrix whose row index is the same as that of the prototype bitmap and whose width is W_c. The value of each point in the mask is defined by (9), also as shown in Fig. 7.

$$\text{Mask}(i,j) = \begin{cases} 1, & 0 < j \leq \text{MAX}\left(RB(i), \frac{W_c}{2}\right) \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

Notice that the right bound "MAX(RB(i),W_c/2)" is important for the sufficiency matching. For example, when the leftmost character is "B," the candidates may include "B," "P," an "L." If simply using $RB(i)$ as the right bound, the masked bitmap of "P" or "L" will also be recognized perfectly. Using the mask of a candidate (denoted by c) to "AND" with the input bitmap pixel by pixel, we can segment the bitmap of the candidate (X_c) from the remainder along the path determined by the right shape of the candidate. Character-adaptive masking is very effective in segmenting *linear* and *nonlinear* merging types. For the *overlapped* type, at least the shape of the left character can be preserved.

C. Merged Character Recognition Paradigm

Compute the recognition probability of c by (7), i.e., $P_c = \mathbf{MATCH}'(f_c, X_c)$. If only accepting the candidate with the highest P_c, the segmentation will have one solution; however, it also has a higher risk of rejection or misrecongition of the entire string of merged characters. On the other hand, if accepting every candidate whose P_c is high enough, the segmentation may yield multiple solutions, and the best solution can be selected by the holistic recognition probability. For the best performance, we choose the latter way.

The recognition of a scanned document image first separates the image into several areas by layout analysis, then divides each area into subimages of text lines by horizontal projection, and finally cuts each text line into character bitmaps at zero-height positions of vertical projection profile. The final bitmap may contain either single character or merged characters, which are distinguished by checking the aspect ratio and the side profile [19]. The bitmap of single character is recognized by the NSM algorithm directly, while the bitmap of merged characters becomes the input bitmap X of our recognition paradigm shown in Fig. 8.

Since this paradigm allows multiple solutions, we need to use a stack to store the branchpoint status for the recursive process. The structure of stack element consists of three variables, as shown in the double-bordered box at the top-left corner of Fig. 8. P_h is the holistic recognition probability up to the branchpoint. "String" contains a list of recognized characters. "Bitmap" is the current remainder of bitmap to be recognized. Initially, the feature library is ready, and the stack contains only one element, whose P_h is 1, "String" is empty, and "Bitmap" is the input bitmap of merged characters.

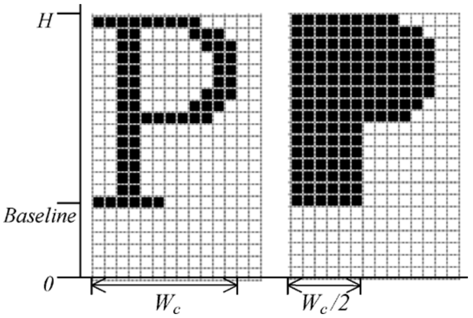


Fig. 7. Mask of "P."

The recursive process begins with popping up the top element of the stack. Then, the forepart prediction takes place to select a small number of candidates into C_4 by four procedures. Next, for each candidate c in C_4, (1) the character-adaptive masking screens out the bitmap of this candidate, denoted by X_c, correctly. (2) The NSM-based character recognition is performed to obtain the recognition probability P_c. (3) Check whether $P_c > \xi$, where ξ is a predefined threshold for accepting the recognition probability. If $P_c \leq \xi$, this candidate is discarded and the following steps are skipped. (4) Check whether the remainder of the input bitmap, i.e., "bitmap"- X_c, is empty. If so, all merged characters are processed. Therefore, a solution, consisting of two variables: P_h and "String," is added to the solution array and the following step is skipped. (5) Push the current recognition status into the stack. The value of the stack element representing the current recognition status is assigned as follows: $P_h = P_h \times P_c$, "string" is appended with the current candidate c, and "bitmap" is the remainder after masking. Once all candidates in C_4 have been processed, check whether the stack is empty. If so, the recognition finishes. Otherwise, perform the recursive process again.

After the recursive process finishes, we get n solutions. If $n = 0$, the rejection happens. If $n = 1$, the only solution becomes the final solution. Otherwise, there exist multiple solutions. Usually, the one with the highest holistic recognition probability (P_h) will become the final solution. When the vocabulary of the application is limited, spelling check is also useful.

IV. Performance Analysis and Experiments

A. Performance Analysis

Table I presents a performance comparison on segmentation between the proposed method and other representative methods in terms of general characteristics, segmentation accuracy, adaptability to three merging types, and speed. The evaluation is based on the experimental results of implementing the projection-based segmentation approach [8], recognition-based segmentation approach [14], and shortest-path segmentation approach [16].

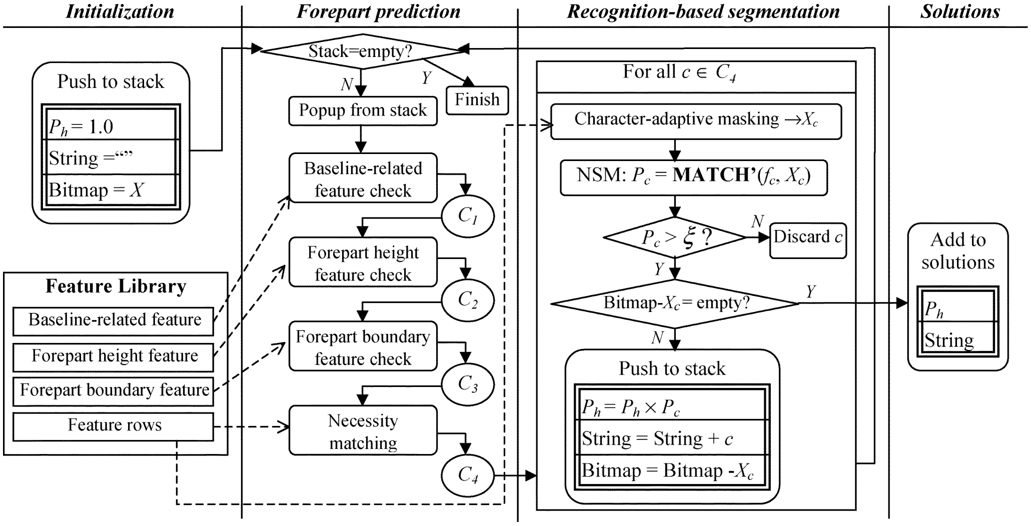


Fig. 8. Merged character recognition paradigm.

TABLE I
PERFORMANCE COMPARISON BETWEEN FOUR SEGMENTATION METHODS

Method / Item	**Projection-based Segmentation** [8]	**Recognition-Based Segmentation** [14]	**Shortest Path Segmentation** [16]	**Our method**
General Characteristics	Static, vertical cutting path.	Dynamic, vertical cutting path.	Dynamic, nonlinear cutting path.	Dynamic, character-adaptive cutting path.
Accuracy of segmentation	Low	Medium	Relatively high	High
Adaptability	Poor	Low	Medium	High
Linear	Medium	Good	Good	Good
Nonlinear	No	No	Medium	Good
Overlapped	No	No	Limited	Medium
Speed	Fast. Projection analysis requires little computation.	Slow. Trying all possible cutting positions.	Fast, the algorithm of searching path is not complicated.	Medium, most of the time was spent on the forepart prediction.

Table I shows that the segmentation quality of our proposed method is the best among the four methods. The advantages of our method come from the forepart prediction and the character-adaptive masking. For the merged characters of the *linear* and *nonlinear* types, our method can segment them correctly. for those of the *overlapped* type, our method can correctly segment at least the left character, while the other methods cannot guarantee segmenting any of them correctly.

B. Experiments

We implemented the proposed method in an experimental system—VHTender, which processes scanned tenders for construction companies. To evaluate the performance of the proposed algorithms in detail, we divide the experiment into two parts: NSM-based character recognition and FP-based merged character segmentation.

1) Result of character recognition: The proposed NSM-based character recognition algorithm is compared with a commercial OCR system—OmniPage Pro 10 (OPP10). We choose OPP10 for two reasons: 1) OPP is a well-known OCR software with a long development history. Its developer, ScanSoft company, claims that OPP10 is one of the best OCR system in the world, 2) The testing documents in our experiment are construction tenders, which contain a lot of underlined texts. Many OCR systems cannot handle underlined texts well, especially when text touches its underline. However, OPP10 is not very sensitive to it.

The testing data are the scanned images of 12 tender pages, containing 12 786 characters. Table II shows the recognition results of two systems obtained in a PC (PIII500/256M).

The number of errors includes both false recognition errors and rejection errors. We analyze the cause of each error and report the error distributions of two systems in Table III.

The experimental results show that both systems achieve high recognition rates. They have similar performances on degraded characters and merged characters; however, VHTender has a much better ability in distinguishing characters that are similar in shape, e.g., "1-l-I," which makes VHTender achieve a higher recognition rate. This advantage of VHTender comes from that the feature rows of the similar characters are automatically defined at the location of the maximum difference and the NSM algorithm is sensitive to the difference of even one feature row. The overall processing time (**T_all**) of VHTender is longer than

TABLE II
PERFORMANCE COMPARISON BETWEEN VHTENDER AND OPP10

Item System	Number of errors	Recognition rate	T_all (sec/page)	T_OCR (sec/page)	Matching function speed (sec)
OPP 10	88	99.31%	0.94	N.A	N.A
VHTender	37	99.71%	1.21	0.24	0.89×10^{-4}

TABLE III
RECOGNITION ERROR DISTRIBUTIONS OF TWO SYSTEMS

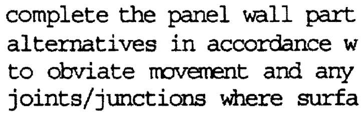
Reason System	Similar in shape	Wrong segmentation	Degradation	Unknown	Total
OPP 10	57	12	7	12	88
VHTender	16	10	11	0	37

that of OPP10. Since VHTender is a system designed for automatic tender analysis, **T_all** includes the time for layout analysis and tender understanding. The pure time for OCR (**T_OCR**) is only one fifth of **T_all**.

2) Result of character segmentation: We select another 20 pages containing heavily merged characters to test the FP-based segmentation algorithm. Fig. 9 shows the nature of the testing image. The total number of merged characters and the distribution of merging types are given in Table IV.

Our FP-based segmentation algorithm (abbreviated to FPS) is compared with the shortest path segmentation algorithm [16] (abbreviated to SPS). For easy understanding, we describe the SPS algorithm briefly. A candidate cutting path P_i starts from each column in the bottom (or top) row of an input bitmap, then selects the least cost for P_i to move to the next row upward (or downward), and finally stops when reaching the top (or bottom) row. At any row, P_i can only move to one of its left-diagonal, vertical, or right-diagonal neighbors in the next row. The cost for a move is defined as follows: a vertical move to a white pixel costs 0, a vertical move to a black pixel costs 10, a diagonal move to a white pixel costs 1, and a diagonal move to a black pixel costs $10\sqrt{2}$. The SPS algorithm uses a one–dimensional (1D) cost array $\{C_i\}$ to record the accumulative cost for each P_i and a two-dimensional (2-D) path array to store the route of each P_i. After obtaining the costs of all candidate paths, the path with the least cost, i.e., the shortest path, will be considered as the cutting path.

Some examples of the segmentation results of the two algorithms are shown in Fig. 10, where the cutting paths are drawn in gray. The cutting path of the SPS algorithm consists of 8-connected vertical lines, whereas that of the FPS algorithm follows the right boundary of the character mask. The left four examples are all with the *Linear* merging type and the vertical length of connected parts are very short. Both two algorithms segment them well with small difference, such as the bottom-left corner of "a" in the example "na," which demonstrates the FPS algorithm can preserve the shape of character better than the SPS algorithm. For the *nonlinear* merging type, e.g. "ob" and "on" in the right examples, the FPS algorithm does a better clean cut than the SPS algorithm. The abilities of two algorithms differs much in handling the *overlapped* merging types. in the example "omp," "om" is with the *nonlinear* type and "mp" is with the *overlapped* type. Since the lengths of connected parts are long, the cost of cutting through the black areas becomes very high. Consequently, the SPS algorithm mis-segments "omp" into "cnp." On the contrary, the FPS algorithm segments three characters correctly by using forepart prediction and character-adaptive masking. The only imperfection of the FPS result is that "p" loses those pixels overlapping with "m."



complete the panel wall part
alternatives in accordance w
to obviate movement and any
joints/junctions where surfa

Fig. 9. Fraction of testing image.

The analysis on the least cost profile of SPS (Fig. 11) reveals the three following theoretical disadvantages of the SPS algorithm:

1) definition of cost prevents the cutting path from passing many black pixels, which, however, is a must to segment long connected parts;
2) usually, the shortest path is not the best cutting path. Sometimes, many paths have similar costs. Therefore, the SPS algorithm has to invoke the character recognition to select the best path, which somewhat diminishes the advantage of "shortest path.;"
3) least cost profiles generated by bottom-up accumulation and top-down accumulation are not consistent.

For example, the top-down cost profile provides a better path to segment "ve" rather than the bottom-up profile. However, the SPS algorithm did not consider this inconsistence. Compared to the "shortest path" concept, the segmentation based on forepart prediction is more straightforward and goal-directed.

Table IV reports the quantitative experimental results, which demonstrates that the FP-based segmentation achieves remarkable improvement when handling the *nonlinear* and *overlapped* merging types. The improvement attributes to three factors: the effective forepart prediction, the character-adaptive masking and the accurate NSM-based character recognition.

V. CONCLUSION

This paper classifies the merging relationship between two involved characters into three types: *linear*, *nonlinear*, and *overlapped*. Existing segmentation methods usually handle the *linear* type well; however, their capabilities of handling

TABLE IV
EXPERIMENTAL RESULT OF FP-BASED SEGMENTATION

Item \ Type		*Linear*	*Nonlinear*	*Overlapped*	Total
Number of characters		805	834	14	1653
FP-based segmentation	**Error_Num**	10	12	6	28
	Accuracy	98.8%	98.6%	57.1%	98.3%
Shortest path segmentation	**Error_Num**	13	161	10	184
	Accuracy	98.4%	80.7%	28.6%	88.9%

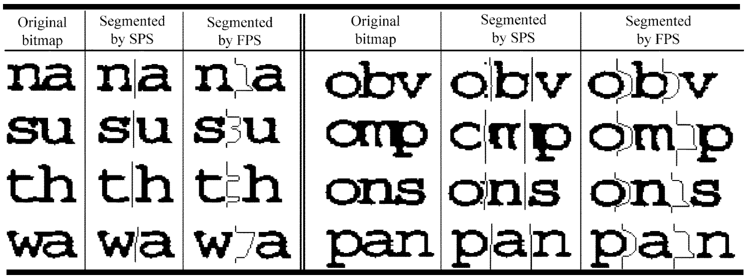


Fig. 10. Segmentation comparison between SPS and FPS.

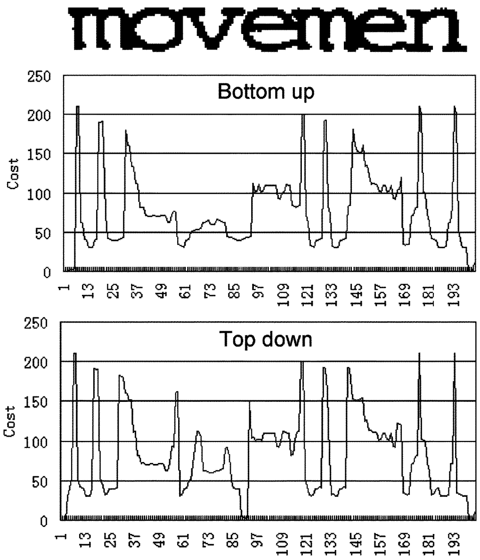


Fig. 11. Least cost profile of SPS.

the other two types are rather limited. This paper proposes a novel merged-character recognition method based on forepart prediction, NSM-based recognition, and character-adaptive masking. This method handles both the *linear* and the *nonlinear* merging types well. For the *overlapped* type, the left character is segmented correctly at the cost of possibly breaking the entirety of the right character. In summary, this paper makes two main contributions. The first one is the NSM-based character recognition. This algorithm utilizes feature rows to reduce the redundancy of character feature such that the feature rows are selected to maximize the difference among similar characters; therefore, this algorithm is fast and accurate. Due to the use of dilated and contracted feature rows, it is also antinoise and able to tolerate one- or two-pixel horizontal displacement between the character model and the character bitmap to be matched. The second contribution is the FP-based segmentation using forepart prediction and character-adaptive masking. The effective forepart prediction quickly eliminates most irrelevant candidates, and the character-adaptive masking, making this segmentation algorithm the first one engaging character-dependent cutting path, is very important to preserve the shapes of merged characters. The experimental results and the performance comparisons with other methods demonstrate that the proposed method achieves remarkable improvement over existing methods.

REFERENCES

[1] O. D. Trier, A. K. Jain, and T. Taxt, "Feature extraction methods for character recognition—a survey," *Pattern Recognit.*, vol. 29, no. 4, pp. 641–662, 1996.

[2] T. Nartker, "ISRI Annual Report," Univ. Nevada, Las Vegas, NV, 1993.

[3] R. G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, pp. 690–706, Jul. 1996.

[4] S. Madhvanath and V. Govindaraju, "The role of holistic paradigms in handwritten word recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 2, pp. 149–164, Feb. 2001.

[5] J. Rocha and T. Pavlidis, "Character recognition without segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 9, pp. 903–909, Sep. 1995.

[6] G. L. Martin, M. Rashid, and J. A. Pittman, "Integrated segmentation and recognition through exhaustive scans or learned saccadic jumps," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 7, no. 4, pp. 831–847, 1993.

[7] S.-W. Lee and S.-Y. Kim, "Integrated segmentation and recognition of handwritten numerals with cascade neural network," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 29, no. 2, pp. 285–290, May 1999.

[8] Y. Lu, "On the segmentation of touching characters," in *Proc. Int. Conf. Document Analysis Recognition (ICDAR)*, Tsukuba City, Japan, Oct. 1993, pp. 440–443.

[9] S. Liang, M. Ahmadi, and M. Shridhar, "Segmentation of touching characters in printed document recognition," in *Proc. Int. Conf. Document Analysis Recognition (ICDAR)*, Tsukuba City, Japan, Oct. 1993, pp. 569–572.

[10] S. Tsujimoto and H. Asada, "Major components of a complete text reading system," *Proc. IEEE*, vol. 80, no. 7, pp. 1133–1149, July 1992.

[11] T. Bayer, U. Krebel, and M. Hammelsbeck, "Segmenting merged characters," in *Proc. 11th Int. Conf. Pattern Recognition (ICPR)*, 1992, pp. 346–349.

[12] T. Bayer and U. Krebel, "Cut classification for segmentation," in *Proc. Int. Conf. Document Analysis Recognition (ICDAR)*, 1993, pp. 565–568.

[13] G. Seni and E. Cohen, "External word segmentation of off_line handwritten text lines," *Pattern Recognit.*, vol. 27, no. 1, pp. 41–52, 1994.

[14] R. G. Casey and G. Nagy, "Recusrive segmentation and classification of composite patterns," in *Proc. 6th Int. Conf. Pattern Recognition (ICPR)*, 1982, pp. 1023–1026.

[15] H. Fujisawa, Y. Nakano, and K. Kurino, "Segmentation methods for character recognition: from segmentation to document structure analysis," *Proc. IEEE*, vol. 80, no. 7, pp. 1079–1092, Jul. 1992.

[16] J. Wang and J. Jean, "Segmentation of merged characters by neural networks and shortest path," *Pattern Recognit.*, vol. 27, no. 5, pp. 649–658, 1994.

[17] S.-W. Lee, D.-J. Lee, and H.-S. Park, "A new methodology for gray-scale character segmentation and recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 10, pp. 1045–1050, Oct. 1996.

[18] L. Y. Tseng and R.-C. Chen, "A new method for segmenting handwritten Chinese characters," in *Proc. Int. Conf. Document Analysis Recognition (ICDAR)*, 1997, pp. 568–571.

[19] M.-C. Jung, Y.-C. Shin, and S. N. Srihari, "Machine printed character segmentation method using side profiles," in *Proc. IEEE Int. Conf. Systems, Man, Cybernetics (SMC)*, vol. 6, 1999, pp. 863–867.

[20] Y.-K. Chen and J.-F. Wang, "Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1304–1317, Nov. 2000.

[21] N. Arica and F. T. Yarman-Vural, "Optical character recognition for cursive handwriting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 6, pp. 801–813, Jun. 2002.

[22] C.-L. Liu, M. Koga, and H. Fujisawa, "Lexicon-driven segmentation and recognition of handwritten character strings for Japanese address reading," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 11, pp. 1425–1437, Nov. 2002.

[23] U. Garain and B. Chaudhuri, "Segmentation of touching characters in printed Devnagari and Bangla scripts using fuzzy multifactorial analysis," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 32, no. 4, pp. 449–459, Nov. 2002.

[24] S. Mori, C. Y. Suen, and K. Yamamoto, "Historical review of OCR research and development," *Proc. IEEE*, vol. 80, pp. 1029–1058, Jul. 1992.

[25] S.-B. Cho, "Recognition of unconstrained handwritten numerals by double self-organizing neural network," in *Proc. 13th Int. Conf. Pattern Recognition*, Vienna, Austria, 1996, pp. 426–430.

[26] C.-T. Chuang and L. Y. Tseng, "A heuristic algorithm for the recognition of printed Chinese characters," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, no. 4, pp. 710–717, Apr. 1995.

[27] T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier system," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 1, pp. 66–75, Jan. 1994.

[28] I.-S. Oh, J.-S. Lee, and C. Y. Suen, "Analysis of class separation and combination of class-dependent features for handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 10, pp. 1089–1094, Oct. 1999.

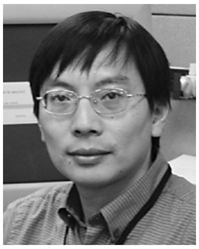
[29] I. S. Oh and C. Y. Suen, "A feature for character recognition base on directional distance distribution," in *Proc. 4th Int. Conf. Document Analysis Recognition*, 1997, pp. 288–292.

[30] K. Yamada, "Non-uniformly sampled feature extraction method for Kanji character recognition," in *Proc. 4th Int. Conf. Document Analysis Recognition*, 1997, pp. 200–205.

[31] T.-C. Chang and S.-Y. Chen, "Character segmentation using convex-hull techniques," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 13, no. 6, pp. 833–858, 1999.

Jiqiang Song (S'99–M'02) received the B.S. degree in computer science and the Ph.D. degree in computer science and application from Nanjing University, Nanjing, China., in 1996 and 2001, respectively.

Currently, he is a Postdoctoral Fellow at the Department of Computer Science and Engineering, the Chinese University of Hong Kong, Shatin. His research interests include graphics recognition, automatic interpretation of engineering drawings, image/video processing, and video compression. He has published over 30 papers in these areas.



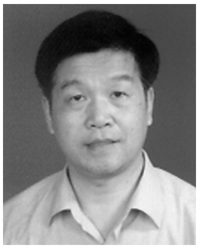
Zuo Li received the B.S. degree in physics from Petroleum University, Beijing, China, in 1988 and the Ph.D. .degree in computer science and application from Nanjing University, Nanjing, China, in 2001.

Currently, he is a Member of the Technical Staff of Bell Labs, Lucent Technologies, Beijing, China. His research interests include graphics recognition and mobility intelligent network solution.

Michael R. Lyu (S'84–M'88–SM'97–F'04) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1981, the M.S. degree in computer engineering from University of California, Santa Barbara, in 1985, and the Ph.D. degree in computer science from University of California, Los Angeles, in 1988.

He is currently a Professor in the Department of Computer Science and Engineering, The Chinese University of Hong Kong. He was with the Jet Propulsion Laboratory, Pasadena, CA, as a Technical Staff Member from 1988 to 1990. From 1990 to 1992, he was with the Department of Electrical and Computer Engineering, The University of Iowa, Iowa City, as an Assistant Professor. From 1992 to 1995, he was Member of the Technical Staff in the applied research area of Bell Communications Research/Bellcore, Morristown, NJ. From 1995 to 1997, he was Research Member of the Technical Staff at Bell Laboratories, Murray Hill, NJ. His research interests include software reliability engineering, distributed systems, fault-tolerant computing, mobile networks, Web technologies, multimedia information processing, and E-commerce systems. He has published over 190 refereed journal and conference papers in these areas. He has participated in more than 30 industrial projects, and helped to develop many commercial systems and software tools. He was the editor of two book volumes: *Software Fault Tolerance* (New York: Wiley, 1995) and *The Handbook of Software Reliability Engineering* (Piscataway, NJ: IEEE and New York: McGraw-Hill, 1996). He is Associate Editor for the *Journal of Information Science and Engineering.*

Dr. Lyu received the Best Paper Awards ISSRE in 1998 and 2003, respectively. He served on the Editorial Board of IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, and has been an Associate Editor of IEEE TRANSACTIONS ON RELIABILITY. He initiated the First International Symposium on Software Reliability Engineering (ISSRE) in 1990. He was the program chair for ISSRE'96, and has served in program committees for many conferences, including ISSRE, SRDS, HASE, ICECCS, ISIT, FTCS, DSN, ICDSN, EUROMICRO, APSEC, PRDC, PSAM, ICCCN, ISESE, and WWW. He was the General Chair for ISSRE2001, and the WWW10 Program Co-Chair. He has been frequently invited as a keynote or tutorial speaker to conferences and workshops in U.S., Europe, and Asia.



Shijie Cai received the B.S. degree in mathematics, from Nanjing University, Nanjing, China., in 1967.

He is a Professor at the Department of Computer Science and Technology, Nanjing University. His research interests include computer graphics, graphics recognition, image processing, and document analysis and recognition.