
ELSEVIER

Pattern Recognition Letters 20 (1999) 791–806

Pattern Recognition Letters

www.elsevier.nl/locate/patrec

Recognition-based handwritten Chinese character segmentation using a probabilistic Viterbi algorithm

Yi-Hong Tseng, Hsi-Jian Lee *

Department of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu 30050, Taiwan, ROC

Received 9 December 1998; received in revised form 12 April 1999

Abstract

This paper presents a recognition-based character segmentation method for handwritten Chinese characters. Possible non-linear segmentation paths are initially located using a probabilistic Viterbi algorithm. Candidate segmentation paths are determined by verifying overlapping paths, between-character gaps, and adjacent-path distances. A segmentation graph is then constructed using candidate paths to represent nodes and two nodes with appropriate distances are connected by an arc. The cost in each arc is a function of character recognition distances, squareness of characters and internal gaps in characters. After the shortest path is detected from the segmentation graph, the nodes in the path represent optimal segmentation paths. In addition, 125 text-line images are collected from seven form documents. Cumulatively, these text-lines contain 1132 handwritten Chinese characters. The average segmentation rate in our experiments is 95.58%.

Moreover, the probabilistic Viterbi algorithm is modified slightly to extract text-lines from document pages by obtaining non-linear paths while gaps between text-lines are not obvious. This algorithm can also be modified to segment characters from printed text-line images by adjusting parameters used to represent costs of arcs in the segmentation graph.

Keywords: Chinese character segmentation; Recognition-based segmentation; Probabilistic Viterbi algorithm

1. Introduction

In an automatic document processing system, an optical character recognition (OCR) engine is used to recognize characters extracted from documents. The performance of character segmentation significantly affects the accuracy of character recognition. Character segmentation for handwritten characters is more complex than that for printed characters due to the diverse varieties of handwriting. In this paper, we present a novel means of segmenting characters from handwritten text-lines, which are characters located in horizontal or vertical blocks.

The previous literature offers several methods for character segmentation. Lu (1996) thoroughly reviewed pertinent literature related to character segmentation in printed documents. His review encompasses techniques for segmenting uniform or proportional fonts, broken and touching characters; techniques based on text image features and recognition results are addressed as well. Lu and Shridhar (1996) also reviewed the techniques used

* Corresponding author. Tel.: +886-3-5731-835; fax: +886-3-5724-176; e-mail: hjlee@csie.nctu.edu.tw

0167-8655/99/$ – see front matter
PII: S0167-8655(99)00043-4

to segment characters from handwritten words. Their review focused on handprinted word segmentation, handwritten numeral segmentation and cursive word segmentation. Casey and Lecolinet (1996) classified character segmentation methods into three strategies: (1) dissection methods, (2) holistic methods and (3) recognition-based methods. In the first strategy, text-line images are cut into meaningful components with predefined properties in character heights, character widths and gaps between characters. These properties, although generally helpful, are unstable in handwritings. Tseng and Chen (1998) proposed a method for segmenting handwritten Chinese characters. The strokes of Chinese characters were first extracted and represented by bounding boxes. Several knowledge-based merging operations were then used to merge stroke bounding boxes into candidate boxes. Finally, best segmentation boundaries were determined by applying a dynamic programming method and considering the aspect ratio of these candidate boxes. In the second strategy, a holistic process recognizes an entire word without segmenting the word. This method is only appropriate for alphanumeric characters. Since Chinese characters are non-alphanumeric, the holistic method is inappropriate for segmenting Chinese text. In the third strategy, candidate segmentation positions are detected and then verified by recognizing segmented images. Recognition-based techniques are more effective than other methods in segmenting characters from Chinese text-lines.

Two major approaches of locating possible segmentation positions are projection profile analysis and connected-component extraction. The former approach projects all black pixels in the X-axis (or Y-axis) and selects positions whose numbers of accumulative pixels are smaller than a threshold as possible segmentation positions. The latter one merges neighboring connected runs that consist of continuous black pixels to be connected components. The boundaries of these components are considered as possible segmentation positions. However, these methods are limited in that they can only detect "linear" segmentation paths. For handwritten text-lines, "non-linear" segmentation paths are preferred to separate characters which overlap other ones in the X (or Y) direction.

Wang and Jean (1994) derived a non-linear cutting path from touching characters by identifying the shortest path. While searching the path for horizontal text-lines, they considered only three types of moves: a downward one and two diagonal ones. A larger cost was assigned to a move that went towards a black pixel than through a white pixel. An additional penalty was applied to diagonal moves to encourage vertical cuts. Lee et al. (1996) initially determined character segmentation regions by analyzing the projection profiles and topographic features extracted from gray-scale images. Next, a multi-stage graph searching algorithm was used to obtain a non-linear segmentation path in each segmentation region. These two methods identified only one non-linear segmentation path from touching characters or a pre-determined segmentation region. Occasionally, touching characters consisting of more than two characters and segmentation regions may contain no segmentation path. Hence, the above investigations applied several heuristic rules to attain extra candidate segmentation paths and then used character recognition results to determine optimal segmentation paths.

The Viterbi algorithm is a dynamic programming technique used to derive an optimal path with linear time complexity on the length of the input sequence. In our recognition-based character segmentation system, a probabilistic Viterbi algorithm is applied to obtain non-linear possible segmentation paths from an entire text-line. To reduce the number of possible segmentation paths, three heuristic checking rules, i.e. path overlapping, between-character gaps and adjacent path distances, are employed to remove redundant segmentation paths. The remaining paths are regarded as candidate segmentation paths and used to construct a segmentation graph. Each graph node represents a candidate path and an edge connects two nodes whose distance is less than a threshold. The cost in each edge is a function of character recognition distances, squareness of characters and gaps in characters. After the shortest path is identified from the segmentation

graph, we locate optimal segmentation paths on the nodes of the shortest path.

Fig. 1 schematically depicts our system. Text-line images are initially divided into $m \times n$ grids, where the values of m and n are determined according to the approximate stroke widths in the images. Possible segmentation paths, which may be linear or non-linear, are detected using a probabilistic Viterbi algorithm. Based on properties of Chinese characters, several redundant paths are removed to determine candidate segmentation paths. After constructing a segmentation graph using the candidate paths and character recognition results, all nodes of the shortest path located in this graph are used to identify optimal segmentation paths.

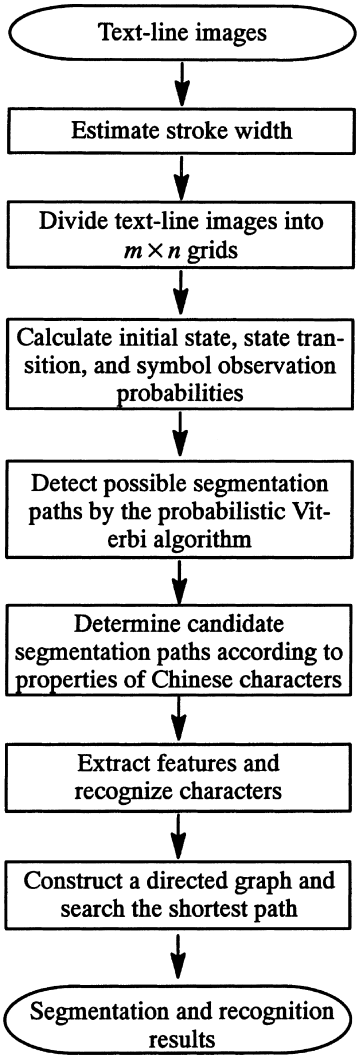


Fig. 1. Diagram of the recognition-based character segmentation system for handwritten Chinese text lines.

The rest of this paper is organized as follows. Section 2 explains the methods of stroke width estimation and the calculation of the four parameters used in the probabilistic Viterbi algorithm. Section 3 describes the probabilistic Viterbi algorithm and how to determine candidate segmentation paths. Next, Section 4 depicts the character recognition engine and the procedure to find the shortest path in a segmentation graph. Section 5 then summarizes the experimental results. Concluding remarks are finally made in Section 6.

2. Estimating stroke width and deriving the probabilistic Viterbi algorithm

To segment overlapping or touching characters by analyzing projection profiles or detecting connected-components is generally difficult. The system proposed herein searches for non-linear segmentation paths between overlapping or touching characters. For touching characters, a segmentation path must be allowed to pass through several black pixels. A *threshold value* T defines the permissible number of black pixels. Character segmentation must determine an appropriate threshold value. If the threshold value is too large, segmentation paths may pass through many black pixels and characters may be divided into several parts by incorrect segmentation paths. In contrast, if the threshold value is too small, touching characters cannot be segmented appropriately.

Our system estimates the approximate stroke width W of a text-line image and determines the threshold value T according to W. Let B_p denote the total number of black pixels in the text-line image and W_p denote the number of black pixels (x, y) whose three neighbors $(x, y+1)$, $(x+1, y)$ and $(x+1, y+1)$ in a 2×2 window have more than two black pixels. The approximate stroke width is defined as $W = B_p/(B_p - W_p)$. Finally, the threshold value is defined as $T = c\, f(W)$, where c is a constant and $f(W)$ is a function of W. Next,

non-linear segmentation paths are attained by constructing a multi-stage directed graph for each text-line image and then searching for all paths passing through black pixels whose number is less than the threshold T.

A multi-stage directed graph (Horowitz and Sahni, 1978) is a directed graph whose vertices are partitioned into $k(\geqslant 2)$ disjoint sets $V_i, 1 \leqslant i \leqslant k$. If $\langle u, v\rangle$ is an edge, then $u \in V_i$ and $v \in V_{i+1}$ for some $i, 1 \leqslant i \leqslant k$. Assume that a vertical input text-line image is divided into $m \times n$ grids, which are represented by the nodes $v_{1,1}, v_{1,2}, \ldots, v_{m,n}$ in the directed graph. Let IW and IH denote the image width and the image height, respectively. The values of m and n are calculated as $m = \lceil \mathrm{IW}/W \rceil$ and $n = \lceil \mathrm{IH}/W \rceil$, where W is the approximate stroke width. A probabilistic Viterbi algorithm is then used to detect *possible segmentation paths* from the entire text-line image rather than a path from pre-determined segmentation regions, which is employed in the method of Lee et al. (1996).

Fig. 2 illustrates a multi-stage directed graph used here, where we add a start node s at stage 0 and an end node t at stage $m+1$. Paths from s to t represent hypothetical character segmentation paths. To segment vertical Chinese text-lines, we consider three segmentation directions, a horizontal one and two diagonal ones. Hence, there are at most three edges from each node $v_{i,j}$ in the directed graph. They are edges $\langle v_{i,j}, v_{i+1,j-1}\rangle$, $\langle v_{i,j}, v_{i+1,j}\rangle$ and $\langle v_{i,j}, v_{i+1,j+1}\rangle$. The possible segmentation paths are those paths derived from this multi-stage directed graph which pass through black pixels and whose number is less than the threshold T.

According to Fig. 2, the multi-stage directed graph can be expressed by Hidden Markov Models (HMMs) (Rabiner and Juang, 1986; Kundu et al., 1989). Each state $v_{i,j}$ in the HMM defines a grid in the graph. The first-order discrete HMM can be described by the model $\Lambda = (\Pi, A, \Gamma, B)$, where $\Pi = \{\pi_i\}$, where $\pi_i = \Pr(v_{1,i}$ at stage 1) is the initial state probability, $A = \{a_{i,J}\}$, where $a_{i,j} = \Pr(v_{(t+1),j}$ at stage $t+1 | v_{t,i}$ at stage $t)$ is the state transition probability, $\Gamma = \{\gamma_j\}$, where $\gamma_i = \Pr(v_{m,j}$ at stage $m)$ is the final state probability, $B = \{b_k(j)\}$, where $b_k(j) = \Pr(v_{k,j}$ at stage $k)$ is the symbol observation probability.

The method to calculate the four probabilities for corresponding nodes and links in the graph is described as follows.

2.1. Symbol observation probability $b_k(j)$

After dividing text-line images, we obtain $m \times n$ grids to construct the multi-stage directed graph. The observation symbol probability $b_k(j)$ for the grid $v_{k,j}$ is calculated as

$$b_k(j) = 1 - \frac{\mathrm{Pixel}_{k,j}}{W \times W + 1},$$

where $\mathrm{Pixel}_{k,j}$ represents the number of black pixels in the grid $v_{k,j}$ and W is the approximate stroke width. Possible segmentation paths are the ones passing through the grids with high probabilities $b_k(j)$.

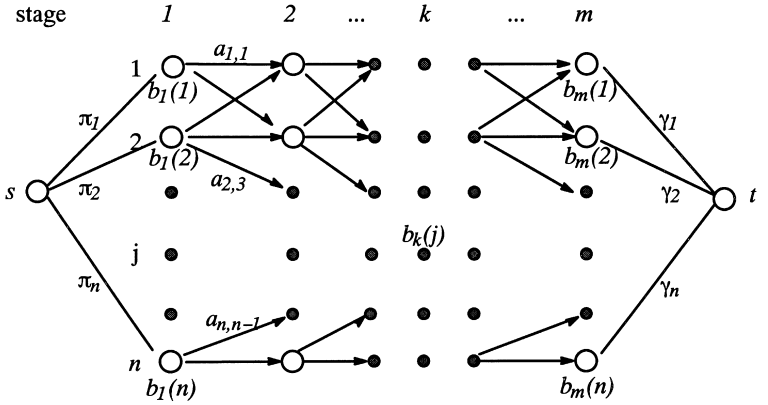


Fig. 2. A multi-stage directed graph for deriving possible segmentation paths.

2.2. State transition probability $a_{i,j}$

Although segmentation paths may be non-linear, linear paths can more feasibly extract characters from text-line images. To obtain segmentation paths, a higher probability is assigned to the straightforward transition than the diagonal transition. For vertical text-lines, the state transition probability $a_{i,j}$ is defined as follows:

$$a_{i,j} = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } j = i-1 \text{ (diagonal direction)}, \\ 1 & \text{if } j = i \quad \text{(horizontal direction)}, \\ \frac{1}{\sqrt{2}} & \text{if } j = i+1 \text{ (diagonal direction)}. \end{cases}$$

Herein, state transition probabilities are adopted to control the selection of cutting directions. Restated, it is desired to derive segmentation paths that move towards the right-hand side as much as possible.

2.3. Initial state probability π_i and final state probability γ_i

The initial state probability π_i and the final state probability γ_i represent the possibilities that the segmentation path starts from the node $v_{1,i}$ and terminates at the node $v_{m,i}$, respectively. Herein, appropriate initial states and final states are selected by accumulating the symbol observation probabilities $b_g(i)$ in the same horizontal position i and calculating the initial state probability π_i and the final state probability γ_i as follows:

$$\pi_i = \left(\sum_{g=1}^{m/3} b_g(i) \right) \Bigg/ \frac{m}{3},$$

$$\gamma_i = \left(\sum_{g=2m/3}^{m} b_g(i) \right) \Bigg/ \frac{m}{3}.$$

3. Deriving non-linear segmentation paths by the probabilistic Viterbi algorithm

In the multi-stage directed graph G, paths from the first stage to the final stage are hypothetical segmentation paths Path_i. Herein, we calculate the probabilities $\Pr(\mathrm{Path}_i; G)$ of all Path_i and select the paths with the maximal probabilities. This work adopts a formal technique for deriving these paths, called the *Viterbi algorithm* (Formey, 1973).

3.1. Possible segmentation paths detection using probabilistic Viterbi algorithm

The first step of the Viterbi algorithm calculates the probabilities of nodes at the first stage. Let $\delta_j(k)$ be the probability of node k at stage j, and $\psi_j(k)$ be the optimal node at stage $j-1$ to node k at stage j. The values of $\delta_1(k)$ and $\psi_1(k)$, where $1 \leqslant k \leqslant n$, are defined as follows:

$$\delta_1(k) = \pi_k \, b_1(k), \quad 1 \leqslant k \leqslant n, \quad \psi_1(k) = 0,$$

where $\delta_1(k)$ represents the probability that grid $v_{1,k}$ is the starting node of a segmentation path.

The second step computes the probabilities of all nodes at stage $j+1$ from nodes at stage j. That is, for $1 \leqslant j \leqslant m-2$ and $1 \leqslant k \leqslant n$

$$\delta_{j+i}(k) = \max_{\substack{k-1 \leqslant i \leqslant k+1 \\ i \geqslant 1, i \leqslant n}} \left[\delta_j(i) \, a_{i,k} \, b_{j+1}(k) \right],$$

$$\psi_{j+1}(k) = \operatorname*{argmax}_{\substack{k-1 \leqslant i \leqslant k+1 \\ i \geqslant 1, i \leqslant n}} \left[\delta_j(i) \, a_{i,k} \right].$$

At the final stage m, the final state probabilities γ_k are used for computing $\delta_m(k)$ and $\psi_m(k)$.

$$\delta_m(k) = \max_{\substack{k-1 \leqslant i \leqslant k+1 \\ i \geqslant 1, i \leqslant n}} [\delta_{m-1}(i) \, a_{i,k} \, b_m(k) \, \gamma_k],$$

$$\psi_m(k) = \operatorname*{argmax}_{\substack{k-1 \leqslant i \leqslant k+1 \\ i \geqslant 1, i \leqslant n}} [\delta_{m-1}(i) \, a_{i,k} \, \gamma_k].$$

In the conventional Viterbi algorithm, the terminal node with the maximum probability is chosen.

$$p^* = \max_{1 \leqslant i \leqslant n} \delta_m(i),$$

$$i_m^* = \operatorname*{argmax}_{1 \leqslant i \leqslant n} \delta_m(i).$$

The optimum segmentation path can then be recovered by backtracking from the array ψ, from the final stage to the first stage. For $j = m-1, m-2, \ldots, 1$, we retrieve the nodes belonging to this optimum segmentation path recursively. That is,

$$i_j^* = \psi_{j+1}(i_{j+1}^*).$$

In the proposed system, the number of possible segmentation paths is possibly more than one in text-line images. After the forward phase in the probabilistic Viterbi algorithm, each terminal node in the final stage stores the probability of a segmentation path. Therefore, a criterion must be established for determining the possible segmentation paths.

The higher probability of a segmentation path implies a lower number of black pixels it passes through and the more straight it is. A threshold value T used to determine all possible segmentation paths is defined as

$$T = \left(1 - \frac{c_1}{W \times W + 1}\right)^{c_2} \left(\frac{1}{\sqrt{2}}\right)^{c_3},$$

where parameter c_1 represents the allowed number of black pixels in the grid, whose size is $W \times W$, passed by a possible segmentation path. In addition, parameter c_2 denotes the permitted number of grids having more than c_1 black pixels in them, and parameter c_3 determines the frequency of diagonal directions that are selected in tracing a segmentation path. In the proposed system, the three parameters c_1, c_2 and c_3 are assigned as W^2, 3 and 4, respectively. Herein, we compute all probabilities $\delta_m(k), 1 \leqslant k \leqslant n$, in the final stage m and take the paths whose probabilities $\delta_m(k)$ are larger than T as possible segmentation paths. Fig. 3 illustrates several possible segmentation paths derived by the probabilistic Viterbi algorithm.

3.2. Candidate segmentation paths determination by removing redundant paths

As is well known, a recognition-based approach is a feasible means of segmenting characters. Although all correct segmentation paths are contained in those possible ones (Fig. 3), a large number of possible paths costs much matching

Fig. 3. Possible segmentation paths derived by the probabilistic Viterbi algorithm.

time to recognize all possible characters. Too many possible characters can be avoided by reducing the number of possible segmentation paths. Herein, we define *candidate segmentation paths* as intermediate results between possible paths and optimal ones. Candidate segmentation paths are obtained by removing redundant possible ones. Several properties of Chinese characters are considered and, then, three checking rules are employed to detect redundant paths.

Rule 1. Only one path is selected from partially overlapping paths.

Among all possible segmentation paths found by the probabilistic Viterbi algorithm, one path Path_i may be partially overlapped with another path Path_j, as shown in Fig. 4, where the node sequences of Path_i and Path_j are $\{i_1^*, i_2^*, \ldots, i_m^*\}$ and $\{j_1^*, j_2^*, \ldots, j_m^*\}$, respectively. The two accumulative probabilities, $\delta_m(i)$ and $\delta_m(j)$, are both larger than the threshold value T. According to Fig. 3, at least two non-overlapping paths could segment three consecutive characters. Hence, one of the two partially overlapping paths is redundant and could be removed. The proposed system removes the path with the smaller probability δ_m and keeps the other one. If the two partially overlapping paths have the same accumulative probabilities, any path can be kept without losing the validity. Restated, if the following two conditions are satisfied, the path Path_i is removed and Path_j is kept:

$$\{i_1^*, i_2^*, \ldots, i_m^*\} \bigcap \{j_1^*, j_2^*, \ldots, j_m^*\} \neq \phi$$
$$\text{and} \quad \delta_m(i) \leqslant \delta_m(j).$$

This rule markedly reduces the number of possible segmentation. Fig. 6(b) summarizes the results of removing redundant paths using this rule.

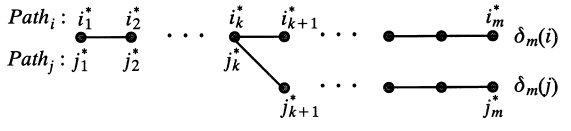


Fig. 4. Two paths, Path_i and Path_j, have k overlapped nodes and one of them is redundant.

Rule 2. Only one path is kept in the gap of two consecutive characters.

The gap between two characters contains several parallel possible paths, as illustrated in Fig. 5. Owing to the fact that they are straight and pass through no black pixels, these paths do not overlap each other and have the largest accumulative probability 1. Actually, any of these paths in the gap can be selected as a candidate segmentation path and the others can be removed. Assume that there are c consecutive possible segmentation paths Path_i, where $j+1 \leqslant i \leqslant j+c$, with the same accumulative probabilities $\delta_m(i) = 1$. Herein, we only select the middle path $\text{Path}_{\lfloor j+c/2 \rfloor}$ as the candidate path and omit the others. Fig. 6(c) summarizes the experimental results using the rule.

Rule 3. One of the two close neighboring paths is removed.

Different writers have a wide diversity of writing styles with respect to handwritten characters. Several properties of printed Chinese characters, e.g. the squareness of characters and gaps between characters, are inappropriate for handwritten Chinese characters. However, the variation of character size is generally not too much for characters written by the same writer. In the proposed system, those text-line images written by a specific person with the same pen are extracted from actual documents. Hence, we calculate the distance D between two neighboring segmentation paths and

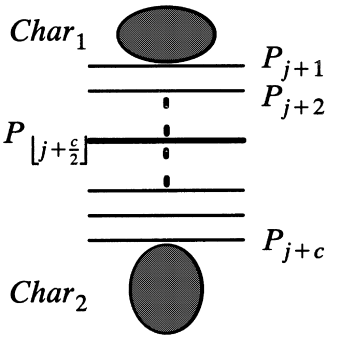


Fig. 5. Appearance of several consecutive segmentation paths in the gap between two characters.

Fig. 6. Results of redundant segmentation path removal: (a) image after detecting all possible segmentation paths; (b) image after removing overlapping paths; (c) image after eliminating redundant path in gaps; (d) image after discarding near paths.

determine the existence of the two paths according to D.

Let Path_i and Path_j be two neighboring segmentation paths, whose node sequences are denoted by $\{\text{grid}_i(0), \text{grid}_i(1), \ldots, \text{grid}_i(m)\}$ and $\{\text{grid}_j(0), \text{grid}_j(1), \ldots, \text{grid}_j(m)\}$, respectively, where $\text{grid}_i(k)$ denotes the kth grid position in the segmentation path P_i, as shown in Fig. 7. We first calculate all distances $d_{i,j}(k) = |\text{grid}_i(k) - \text{grid}_j(k)|$ between corresponding positions, where $1 \leqslant k \leqslant m$. In order to estimate the proper path distance $D_{i,j}$ between Path_i and Path_j, we calculate all path distances $d_{i,j}$, $1 \leqslant k \leqslant m$, and select the median value as the distance $D_{i,j}$. In addition, *adjacent path checking rule* is applied to determine which path should be removed and which should be kept.

If $D_{i,j} > 1.5 \times W$, then both Path_i and Path_j are reserved;
  else if $\delta_m(i) \leqslant \delta_m(j)$, then Path_i is removed and Path_j is kept;
    else Path_i is kept and Path_j is removed,

where W represents the estimated stroke width of the input text-line image and $\delta_m(\cdot)$ is the accumulative probability of a path in the last stage m.

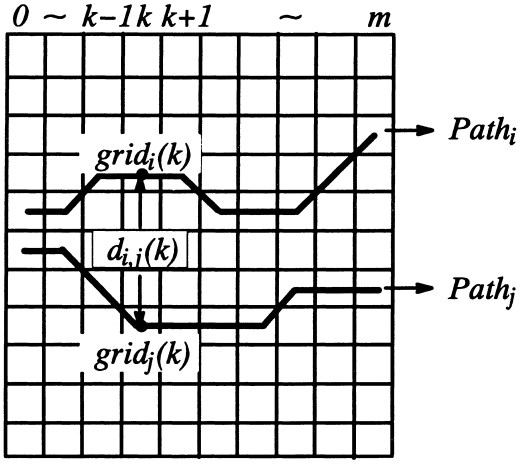


Fig. 7. Calculation of distances between two neighboring segmentation paths.

Fig. 6(d) summarizes the experimental results of applying this rule to Fig. 6(c). For touching characters, this rule can keep the most possible path and remove redundant paths which cannot be eliminated by the first two rules.

4. Identifying optimal segmentation paths according to character recognition results

Candidate segmentation paths are obtained by removing redundant segmentation paths according to the above three rules. Herein, optimal segmentation paths are determined from these candidate ones by applying a recognition-based technique in the proposed system.

4.1. Character extraction by finding approximate region

Character images can be extracted from text-lines according to two bounding segmentation paths. Let a character be bounded by two non-linear paths Path_i and Path_j (Fig. 8), whose node sequences are $\{\mathrm{grid}_i(0), \mathrm{grid}_i(1), \ldots, \mathrm{grid}_i(m)\}$ and $\{\mathrm{grid}_j(0), \mathrm{grid}_j(1), \ldots, \mathrm{grid}_j(m)\}$, respectively. Two point-sets, i.e. upper point-set (UPS) and lower point-set (LPS), are defined herein to record boundary positions of grids in the two non-linear paths.

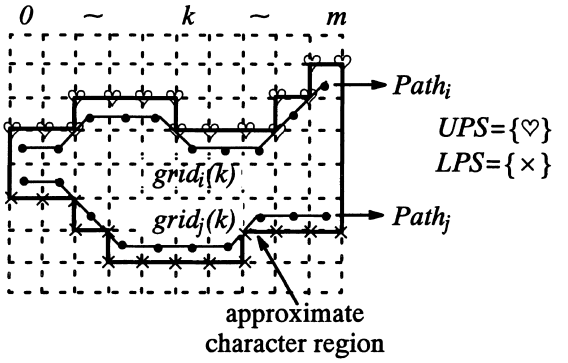


Fig. 8. An approximate character region bounded by two non-linear segmentation paths.

$$\mathrm{UPS} = \bigcup_{p=1}^{m} \{\mathrm{LT}(\mathrm{grid}_i(p)), \mathrm{RT}(\mathrm{grid}_i(p))\},$$

$$\mathrm{LPS} = \bigcup_{p=1}^{m} \{\mathrm{LB}(\mathrm{grid}_j(p)), \mathrm{RB}(\mathrm{grid}_j(p))\},$$

where $\mathrm{LT}(\cdot)$, $\mathrm{RT}(\cdot)$, $\mathrm{LB}(\cdot)$ and $\mathrm{RB}(\cdot)$ denote the left-top, right-top, left-bottom and right-bottom positions of a grid, respectively. The *approximate region* of this character is defined as a polygon composed of points in the *union point-set* $\mathrm{APS} = \mathrm{UPS} \cup \mathrm{LPS}$. Herein, the character boundary is obtained by finding a minimum rectangle to bound all black pixels in the appropriate region and the image in this rectangle is a segmented character image. Fig. 9 represents an illustrative example of segmenting two characters by three segmentation paths.

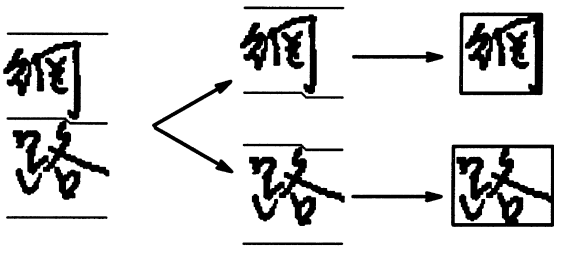


Fig. 9. An example of character segmentation by determining approximate regions and finding bounding rectangles.

4.2. Optimal segmentation paths determination using recognition results

The validity of candidate paths is verified by feeding extracted character images into an OCR engine and using recognition results to determine optimal segmentation paths. Two main techniques, i.e. *local decision* and *global decision*, are used to determine optimal segmentation paths according to recognition results. Let $S_{\text{cand}} = \{\text{Path}_1, \text{Path}_2, \ldots, \text{Path}_k\}$ be a set of k candidate segmentation paths and $S_{\text{final}} = \{\text{Path}_{c1}, \text{Path}_{c2}, \ldots, \text{Path}_{cl}\}$ a set of l optimal ones, where $S_{\text{final}} \subset S_{\text{cand}}$, as depicted in Fig. 10.

The local decision technique uses the recognition distance $\text{Dist}_{c_i,j}$ of character $C_{c_i,j}$ between Path_{c_i} and Path_j to determine the correctness of Path_j. If the distance $\text{Dist}_{c_i,j}$ is smaller than a predefined threshold T_{ocr}, then Path_j is regarded as the next optimal segmentation path $\text{Path}_{c_{i+1}}$.

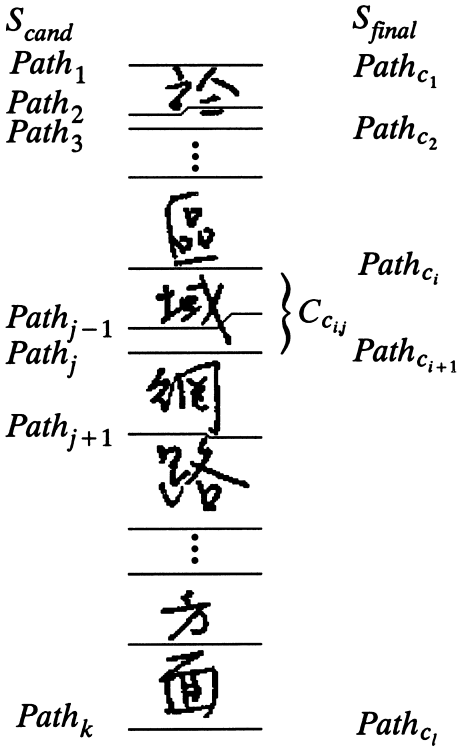


Fig. 10. Determination of c_l optimal segmentation paths from k candidate ones.

Otherwise, character $C_{c_i,j}$ is regarded as an incomplete character and path Path_j is discarded. Threshold T_{ocr} is generally determined by recognizing a large amount of training samples and calculating the distribution of recognition distances, as illustrated in Fig. 11.

Although threshold T_{ocr} plays an important role in the local decision technique, a stable T_{ocr} is difficult to determine for all OCR engines. On the other hand, Chinese characters may be composed of several parts, called *radicals*. Occasionally, these parts can also be characters. Only using local images, a radical image may be recognized as a character and other parts of character would be merged to the next character. Hence, if an error occurs when determining the validity of a candidate path, the validity determination of the next path usually causes error as well.

The global decision technique determines optimal segmentation paths after each character bounded by two candidate paths is recognized. This technique does not require defining a threshold for assessing the validity of segmentation paths. The *shortest path searching* algorithm is usually employed to derive the optimum solution in this technique. All candidate segmentation paths and character recognition distances are represented as nodes and costs in arcs of a graph. The nodes in the shortest path of the graph are taken as the optimal segmentation paths. In the proposed system, we construct a *segmentation graph* G_{seg} for each text-line image, where each node represents a candidate path and the cost of an arc is a function of recognition distances, squareness of characters, and internal gaps of characters.

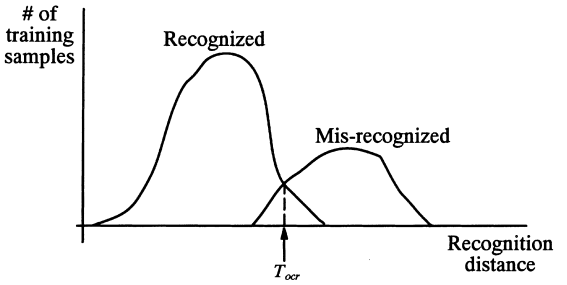


Fig. 11. Determination of threshold T_{ocr} according to the distribution of recognition distances.

4.2.1. Merging consecutive components as a character based on the aspect ratio

In general, characters in a text-line are written by the same writer and the writing characters have similar sizes. Herein, we calculate the average width (AW) and average height (AH) of all components bounded by two consecutive candidate segmentation paths. A character height threshold $H_{\text{thr}} = \text{AH} \times 1.2$ is then defined to determine whether or not consecutive components should be merged. Restated, if $C_1, C_2, \ldots, C_k$ are k consecutive components and the distance between the bottom of C_k and the top of C_1 is smaller than the height threshold H_{thr}, then these k components will be merged into a character.

4.2.2. Extra information

The shortest path searching algorithm derives an optimum path with the minimum accumulative cost in the segmentation graph. Occasionally, the recognition distance of a component C_3 consisting of two characters, C_1 and C_2, is smaller than the sum of the recognition distances from C_1 and C_2, as illustrated in Fig. 12. Hence, extra information of Chinese characters must be used to represent costs of arcs in the segmentation graph. In addition to the recognition distances, the squareness and the internal gaps of Chinese characters are used in our shortest path finding procedure. Restated, the cost of an arc connecting two paths, P_i and P_j, is represented by $\text{Cost}_{i,j} = k_{\text{rd}} \times \text{RD}_{i,j} + k_{\text{squ}} \times \text{SQU}_{i,j} + k_{\text{gap}} \times \text{GAP}_{i,j}$, where $\text{RD}_{i,j}$, $\text{SQU}_{i,j}$ and $\text{GAP}_{i,j}$ denote the recognition distance, squareness and internal gap of the character $C_{i,j}$ between P_i and P_j, respectively, the three constants, k_{rd}, k_{squ} and k_{gap}, are assigned to 5, 4 and 8, respectively, in our system.

Let CW and CH denote the character width and character height, respectively. The *squareness* of a character is defined as

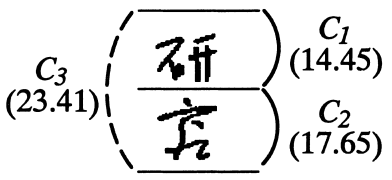


Fig. 12. Sum of recognition distances from C_1 and C_2 ($=32.10$) is larger than the recognition distance of C_3 ($=23.41$).

$$\text{SQU} = \frac{\min(\text{CW}, \text{CH})}{\max(\text{CW}, \text{CH})},$$

where $0 \leqslant \text{SQU} \leqslant 1$. Although Chinese characters may consist of several separated radicals, gaps between these radicals are generally smaller than gaps between consecutive characters. Hence, the larger number of gaps in a component implies a lower probability that the component is a character. Let $\text{Pix}(i,j)$ represent the value of pixel (i,j) in a binary image. The pixel is black while $\text{Pix}(i,j) = 1$; otherwise, it is white. In vertical text-lines, the *internal gaps* of a component is defined as the number of position j whose accumulative black pixel equals zero, that is, for each $j \in [1, \text{CH}], \sum_{i=1}^{\text{CW}} P(i,j) = 0$.

4.2.3. Normalization of three character evaluators

As mentioned earlier, the range of SQU is [0, 1] and GAP is [0, CH]. In our OCR engine (Tseng et al., 1998), two statistical features, i.e. contour-direction counts and crossing counts, were used. A 2-D character image was initially decomposed non-uniformly into eight sub-regions in both horizontal and vertical directions. The two features were then extracted from these sub-regions. Since only a small number of characters are recognized correctly with recognition distances larger than 30, we can set the range of RD as [0, 30]. Combining these three character evaluators, RD, SQU and GAP, involves normalizing these evaluators into the same range [0, 1].

Fig. 13 illustrates the normalization of the three evaluators. The larger normalized value implies a larger corresponding cost. According to this figure, those characters with a recognition distance larger than 30 are regarded as incorrect characters and their bounding segmentation paths are incorrect. The same assumption is employed in the normalization of GAP. Characters are incorrect when gaps are larger than CH/2. The arc cost, connecting two paths Path_i and Path_j, is represented by three normalized evaluators as follows:

$$\text{Cost}_{i,j} = k_{\text{rd}} \times \text{RD}'_{i,j} + k_{\text{squ}} \times \text{SQU}'_{i,j} + k_{\text{gap}} \times \text{GAP}'_{i,j}.$$

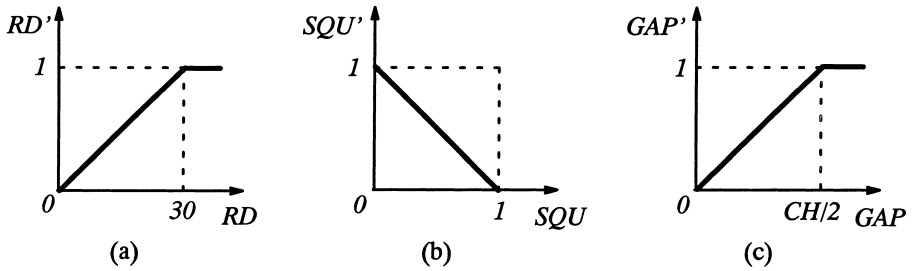


Fig. 13. Normalization of three evaluators: (a) recognition distance; (b) squareness of a character; (c) internal gap in a character.

Fig. 14(a) displays a text-line image after candidate segmentation paths are determined, while Fig. 14(b) illustrates the corresponding segmentation graph. Fig. 14(c) summarizes the results of optimal segmentation paths found from the segmentation graph.

4.3. Extension to segment printed Chinese characters and handwritten text-lines

To segment characters from printed Chinese text-line images, we must use another OCR engine which is trained by printed sample characters and

Fig. 14. (a) Candidate segmentation paths in a vertical text-line; (b) the corresponding segmentation graph and the shortest path of the graph; (c) the optimal segmentation paths determined according to the shortest path.

modify the three constants k_{rd}, k_{squ} and k_{gap} employed to determine optimal segmentation paths. The fact that the squareness of characters and the internal gaps in characters are more significant for printed characters than those for handwritten ones, accounts for why the weights k_{squ} and k_{gap} must be increased for segmenting printed characters. Fig. 15 summarizes the segmentation results of a printed Chinese text-line and the three parameters k_{rd}, k_{squ} and k_{gap} are assigned to 5, 8 and 8, respectively.

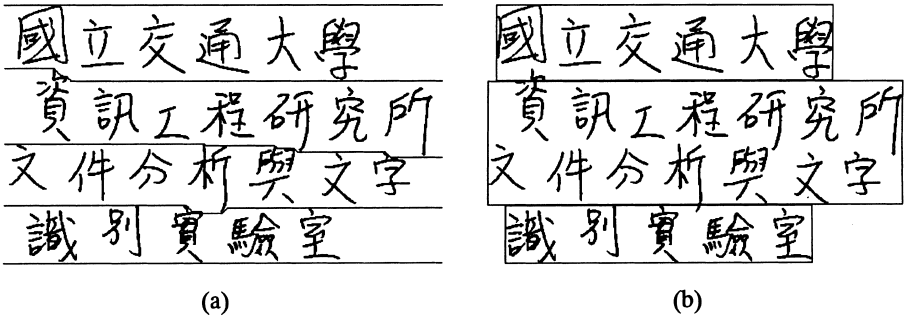
Extracting text-lines by analyzing projection profiles in handwritten documents would be relatively difficult when obvious gaps do not exist. This study attempted to extend the proposed method to detect non-linear segmentation paths, which resembled those used to segment characters from text-lines. While applying the process to segment characters, a test image was initially divided into $m \times n$ grids according to the approximate stroke width. After all possible segmentation paths are derived and redundant ones are removed, we obtain candidate segmentation paths. Owing to that the text-lines could not be directly recognized, these candidate segmentation paths are used to extract text-lines. Fig. 16 presents two results of text-line extraction using the probabilistic Viterbi algorithm and the projection-profile analysis approach.

Fig. 15. Segmentation result for a printed Chinese text-line.

5. Experimental results and analysis

This character segmentation system was implemented as a Windows-based application in Visual C++ with MFC (Microsoft Foundation Class) on a PentiumII-233 PC. All test documents were scanned using a Umax Vista-s8 flatbed scanner at a resolution of 300 dpi (dots per inch). The scanned images were saved as a one-bit, black-and-white, document in BMP format. Fig. 17 displays a test image.

125 text-lines were selected from seven form documents. The total number of characters in these text-lines was 1132. This study also evaluated the proposed system and compared the results between the connected-component-based approach



(a) (b)

Fig. 16. Text-line extraction by (a) the probabilistic Viteri algorithm and (b) projection-profile analysis approach.

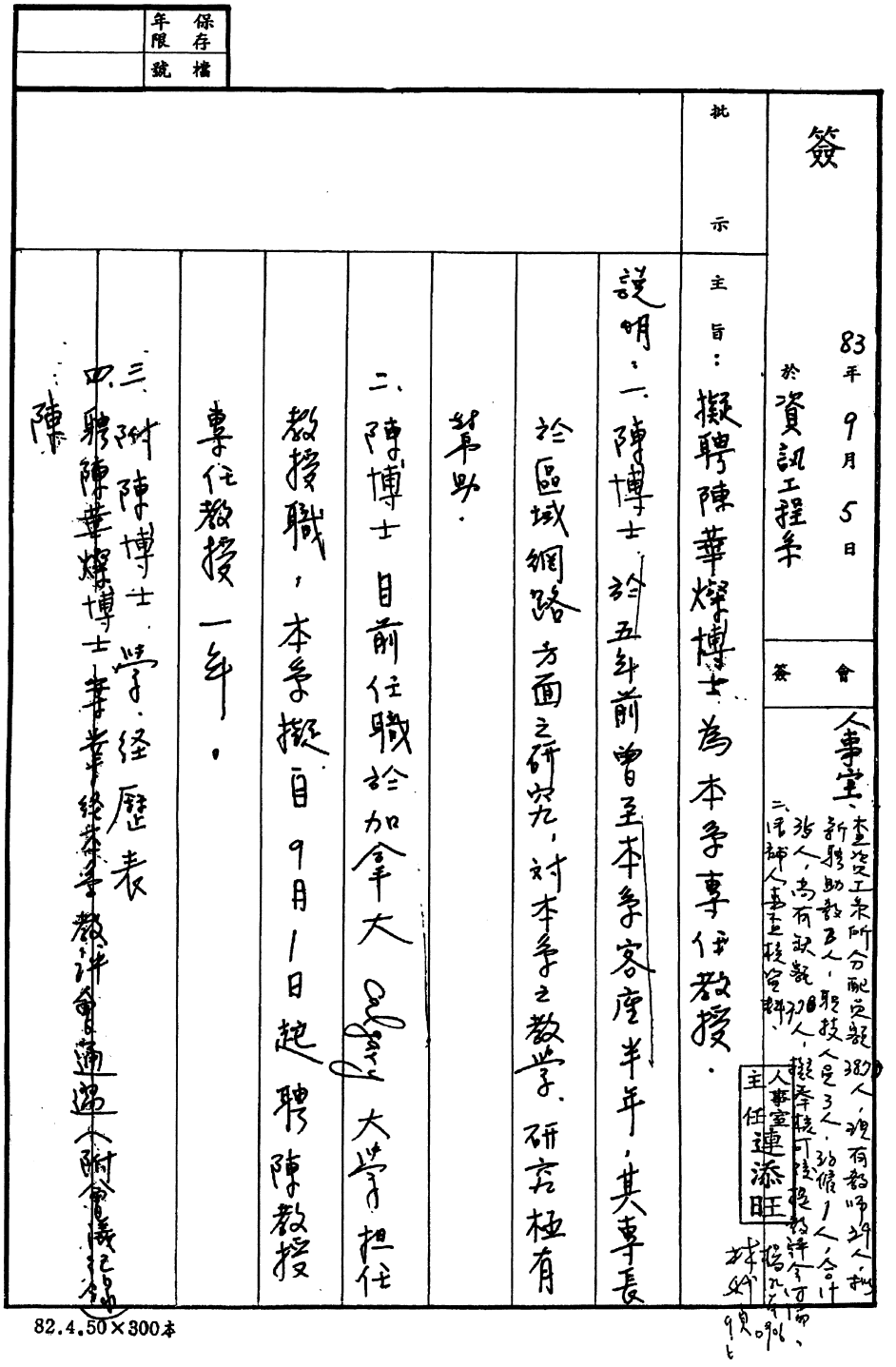


Fig. 17. Example of test form document.

Table 1
Experimental results of handwritten character segmentation

	Connected-component detection	Projection-profile analysis	Our method (RD only)	Our method (RD + SQU + GAP)
Document 1 (173 characters)	151	139	162	168
Document 2 (155 characters)	143	148	149	151
Document 3 (112 characters)	92	82	101	107
Document 4 (189 characters)	171	163	175	185
Document 5 (150 characters)	130	124	135	142
Document 6 (208 characters)	177	161	188	196
Document 7 (136 characters)	118	103	124	133
Average segment rate (%)	86.75%	81.27%	92.07%	95.58%

and the projection profile-based method. Estimation was also performed of the results while only recognition distances were used for detecting the shortest path from the segmentation graph. A *segmentation rate* is defined as the percentage of characters that are segmented precisely. Table 1 summarizes the experimental results. The segmentation rate is 86.75% by detecting connected-components and 81.27% by analyzing the projection profiles. When only recognition distances are used in our system, the segmentation rate is 92.07%. Notably, the segmentation rate is 95.58% if we use recognition distances, squareness of characters, and internal gaps in characters to determine the optimal segmentation paths.

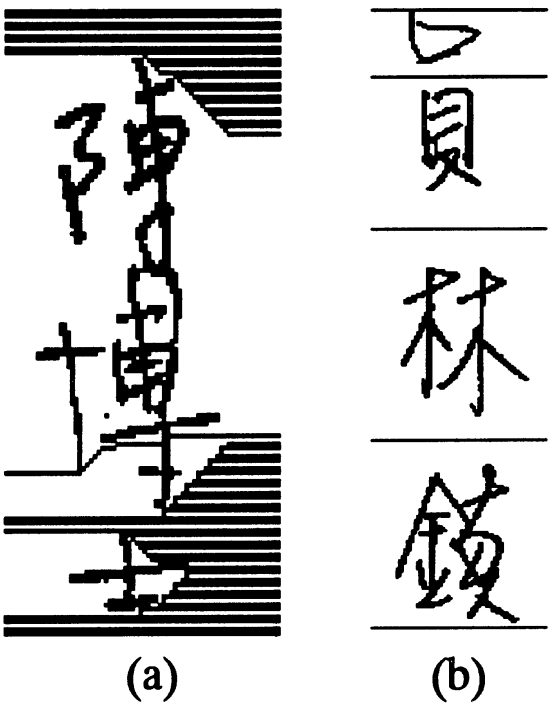


Fig. 18. Two main resons cause our character segmentation module failure: (a) touched character; (b) separated character.

In the proposed system, all possible segmentation paths are found from text-line images and then three rules are used to determine candidate paths. Finally, the shortest path finding algorithm was employed to select optimal segmentation paths. Incorrect segmenting of the characters can be attributed to two reasons. First, two characters touch each other to the extent that no possible segmentation paths were found at the touching position, as shown in Fig. 18(a). Second, a thin character was composed of separated radicals and large gaps existed between these radicals, as shown in Fig. 18(b). If the character height (CH) did not satisfy the constrain: $CH < 1.2 \times AH$, where AH denotes the average character height in the text-line, then these separated radicals could not be merged.

6. Conclusions

This work presents a novel means of extracting text-lines from text blocks and segmenting characters from text-lines. A probabilistic Viterbi algorithm is used to derive all possible segmentation paths which may be linear or non-linear. To reduce the computational cost, redundant paths

are removed by checking path overlapping, between-character gaps and adjacent-path distances. Those remaining paths, called candidate segmentation paths, are employed to construct a segmentation graph. The cost of an arc is a function of recognition distances, squareness of characters and internal gaps in characters. After the shortest path is obtained from the graph, the optimal segmentation paths are located on the nodes of the shortest path. We collect 125 text-line images from seven form documents. Cumulatively these text-lines contain 1132 characters, and the average segmentation rate is 95.58%.

Contextual information is generally applied in the post-processing of an OCR engine. For those mis-recognized characters, contextual information can be used to correct such characters. In our character segmentation module, characters touched seriously and separated obviously cannot be segmented correctly. These errors could be reduced using contextual information while most characters in the text-line are segmented and recognized. To enhance recognition accuracy, future research should focus on segmenting text-lines mixed by Chinese characters and alphanumerics, and correcting mis-segmented characters using context information.

References

Casey, R.G., Lecolinet, E., 1996. A survey of methods and strategies in character segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 18 (7), 690–706.

Formey Jr., G.D., 1973. The Viterbi algorithm. IEEE Proc. 61, 268–278.

Horowitz, E., Sahni, S., 1978. Fundamentals of Computer Algorithms. Computer Science Press, Rockville, MD.

Kundu, A., He, Y., Bahl, P., 1989. Recognition of handwritten word: First and second order hidden Markov model based approach. Pattern Recognition 22, 283–297.

Lee, S.W., Lee, D.J., Park, H.S., 1996. A new methodology for gray-scale character segmentation and recognition. IEEE Trans. Pattern Anal. Mach. Intell. 18 (10), 1045–1050.

Lu, Y., 1996. Machine printed character segmentation – an overview. Pattern Recognition 28 (1), 67–80.

Lu, Y., Shridhar, M., 1996. Character segmentation in handwritten words – an overview. Pattern Recognition 29 (1), 77–96.

Rabiner, L.R., Juang, B.H., 1986. An introduction to hidden Markov models. IEEE ASSP Mag. January, pp. 4–16.

Tseng, L.Y., Chen, R.C., 1998. Segmenting handwritten Chinese characters based on heuristic merging of stroke bounding boxes and dynamic programming. Pattern Recognition Letters 19 (10), 963–973.

Tseng, Y.H., Kuo, C.C., Lee, H.J., 1998. Speeding-up character recognition and in an automatic document reading system. Pattern Recognition 31 (11), 1601–1612.

Wang, J., Jean, J., 1994. Segmentation of merged characters by neural networks and shortest path. Pattern Recognition 27 (5), 649–658.