Optical character recognition for cursive handwriting

Article *in* IEEE Transactions on Pattern Analysis and Machine Intelligence · July 2002

DOI: 10.1109/TPAMI.2002.1008386 · Source: IEEE Xplore

2 authors:

Nafiz Arica
Bahçeşehir University
74 PUBLICATIONS **1,218** CITATIONS
SEE PROFILE

Fatos Tunay Yarman Vural
Middle East Technical University
190 PUBLICATIONS **2,849** CITATIONS
SEE PROFILE

Optical Character Recognition for Cursive Handwriting

Nafiz Arica, *Student Member*, *IEEE*, and Fatos T. Yarman-Vural, *Senior Member*, *IEEE*

Abstract—In this paper, a new analytic scheme, which uses a sequence of segmentation and recognition algorithms, is proposed for offline cursive handwriting recognition problem. First, some global parameters, such as slant angle, baselines, and stroke width and height are estimated. Second, a segmentation method finds character segmentation paths by combining gray scale and binary information. Third, Hidden Markov Model (HMM) is employed for shape recognition to label and rank the character candidates. For this purpose, a string of codes is extracted from each segment to represent the character candidates. The estimation of feature space parameters is embedded in HMM training stage together with the estimation of the HMM model parameters. Finally, the lexicon information and HMM ranks are combined in a graph optimization problem for word-level recognition. This method corrects most of the errors produced by segmentation and HMM ranking stages by maximizing an information measure in an efficient graph search algorithm. The experiments in dicate higher recognition rates compared to the available methods reported in the literature.

Index Terms—Handwritten word recognition, preprocessing, segmentation, optical character recognition, cursive handwriting, hidden Markov model, search, graph, lexicon matching.

1 Introduction

The most difficult problem in the field of Optical Character Recognition (OCR) is the recognition of unconstrained cursive handwriting. The present tools for modeling almost infinitely many variations of human handwriting are not yet sufficient. The similarities of distinct character shapes, the overlaps, and interconnection of the neighboring characters further complicate the problem. Additionally, when observed in isolation, characters are often ambiguous and require context information to reduce the classification error. Thus, current research aims at developing constrained systems for limited domain applications such as postal address reading [21], check sorting [8], tax reading [20], and office automation for text entry [7]. A well-defined lexicon plus a well-constrained syntax help provide a feasible solution to the problem [11].

Handwritten Word Recognition techniques use either holistic or analytic strategies for training and recognition stages. Holistic strategies employ top-down approaches for recognizing the whole word, thus eliminating the segmentation problem [9]. In this strategy, global features, extracted from the entire word image, are used in recognition of limited-size lexicon. As the size of the lexicon gets larger, the complexity of algorithms increase linearly due to the need for a larger search space and a more complex pattern representation. Additionally, the recognition rates decrease rapidly due to the decrease in between-class-variances in the feature space.

The analytic strategies, on the other hand, employ bottom-up approaches, starting from stroke or character-level and going towards producing a meaningful text. Explicit [23] or implicit [16] segmentation of word into characters or strokes is required for this strategy. With the cooperation of segmentation stage, the problem is reduced to the recognition of simple isolated characters or strokes, which can be handled for unlimited vocabulary. However, there is no segmentation algorithm available in the literature for correctly extracting the characters from a given word image. The popular techniques are based on over-segmenting the words and applying a search algorithm for grouping segments to make up characters [14], [10]. If a lexicon of limited size is given, dynamic programming is used to rank every word in the lexicon. The word with the highest rank is chosen as the recognition hypothesis. The complexity of search process for this strategy also increases linearly with the lexicon size, if the flat representation of lexicon is used. More efficient representations such as trie and hash tables can be used in order to reduce the search space.

Application of the preprocessing techniques to a given image, may introduce unexpected distortion (closing loops, breaking character, spurious branches etc.) to the data, which may cause unrecoverable errors in the recognition system. Most of the existing character recognition systems threshold the gray-level image and normalize the slant angle and baseline skew in the preprocessing stage. Then, they employ the normalized binary image in the segmentation and recognition stages [10], [16], [3]. However, in some cases, normalization may severely deform the writing, generating improper character shapes. Furthermore, through the binarization of the gray scale document image, useful information is lost. In order to avoid the limitation of binary image, some recent methods use gray-level image [13]. There, however, the insignificant details suppress important shape information.

The scheme developed in this study, employs an analytic approach on gray-level image, which is supported by binary image and a set of global features. Document image is not

• *The authors are with the Computer Engineering Department, Middle East Technical University, Ankara, Turkey.*
E-mail: {nafiz, vural}@ceng.metu.edu.tr.

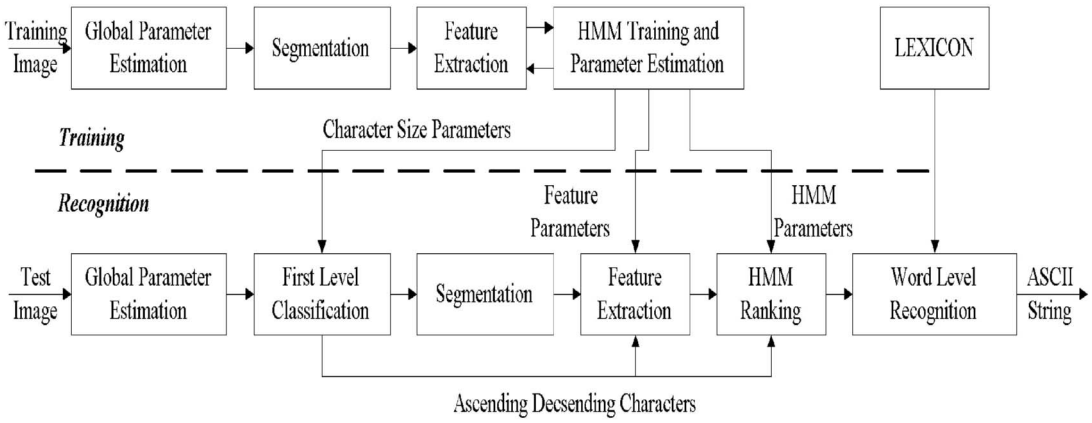

Fig. 1. System overview.

preprocessed for noise reduction and normalization. However, global parameters, such as lower-upper baseline and slant angle are estimated and then incorporated to improve the accuracy of the segmentation and recognition stages. The scheme makes concurrent use of binary and gray-level image in a mixed way to extract the maximum amount of information for both segmentation and recognition.

The segmentation algorithm, proposed in this study, segments the whole word into strokes, each of which corresponds mostly to a character or rarely to a portion of a character. Recognition of each segment is accomplished in three stages: In the first stage, characters are labeled in three classes as ascending, descending, and normal characters. In the second stage, Hidden Markov Model (HMM) is employed for shape recognition. The features extracted from the strokes of each segment are fed to a left-right HMM. The parameters of the feature space are also estimated in the training stage of HMM. Finally, an efficient word-level recognition algorithm resolves handwriting strings by combining lexicon information and the HMM probabilities.

2 System Overview

The proposed system receives the gray-level word image as input, assuming the segmentation of input image into individual words is performed. Although the system is designed for cursive handwriting, methodologies used in the system are easily applicable to machine or hand-printed characters.

System overview is summarized by the block diagram representation in Fig. 1. Global parameter estimation, segmentation, and feature extraction stages employ both gray-level and binary images. The parameters for HMM and feature space are estimated by using the correctly segmented character images in training. These parameters are then used in feature extraction and HMM ranking of character segments. Finally, the word-level recognition algorithm maximizes an information measure, using the HMM probabilities and lexicon information, resulting with ASCII strings. If the input image consists of isolated characters, the segmentation stage is omitted.

Global Parameter Estimation. The output of the global parameter estimation stage is the word-level features, such as average stroke width/height, baselines, skew, and slant angles (see Section 3).

First Level Character Classification. The baselines and character size information estimated in HMM training stage are used to decide on the ascending and descending character thresholds in a given word image. The character size information contains the height-to-width ratios of ascending, descending, and normal characters (see Section 4).

Segmentation. Initially, the word image is divided into segmentation regions each of which contains a segmentation path. Then, a search process finds the segmentation path in each region in order to split the connected characters. The algorithm performs the search process by combining the characteristics of gray scale and binary images. The proposed method slightly over-segments the word image (see Section 5).

Feature Extraction and HMM Training. Since HMM is most successful in the recognition of one-dimensional string of codes, it is critical to represent the two-dimensional information of character images as one dimensional strings. A feature extraction scheme proposed by the authors of this study [1] is employed in this stage, where a set of directional skeletons is extracted by scanning a fixed size window in various directions (see Section 6.1). HMM training is performed on the selected output of the segmentation stage for the estimation of both HMM parameters and the parameters of feature space. These parameters are composed of the character window size, number of scanning directions, and number of regions in each scanning direction. The parameters, which give the maximum recognition rate for the training set, are then used to form the feature space of recognition stage (see Section 6.2).

HMM Ranking. Each string of codes extracted from a character segment is fed to the HMM recognizer. The output is a ranked list of character labels with the associated HMM probabilities for one or more sequential segments (see Section 6.3).

Word Level Recognition. The goal of this stage is to recognize the unknown word by using the candidate characters of the HMM recognizer and the lexicon information. For this purpose, the candidate characters and the associated HMM probabilities are represented in a word graph. Then, dynamic programming is invoked for finding the best path in the word graph, which corresponds to a valid word in the lexicon (see Section 7).

(a) (b)

Fig. 2. Slant angle detection of the word "eighteen." (a) Near vertical contours. (b) Slant of each contour.

3 Global Parameter Estimation

In this stage, first, the input image is binarized by using the optimal thresholding method in [22]. Then, the eight-neighbor boundary extraction algorithm [2] finds the closed contours in the binary word image. An averaging mask smoothes the character contours.

Second, average stroke width/height estimation, slant angle detection, and baselines extraction are performed on the binarized image. These parameters are essential in the segmentation algorithm. Gray-scale and binary-word images are employed together with the global parameters in the subsequent stages of the proposed scheme. Note that, this stage does not perform any noise reduction or normalization on the gray-level word image.

3.1 Stroke Width/Height Estimation

A simple program is developed to estimate the average stroke width, which is then used to decide whether the character segmentation regions are spurious or not. This is a two-scan procedure. The first scan on each row of the binary image calculates the stroke width histogram by counting the black pixel runs in horizontal direction. Then, the mean width, estimated over all of the rows, is taken as the upper bound (*maximum width*) for the run length of the strokes. The second scan on the stroke width histogram discards those strokes whose run length is greater than *maximum width*. Finally, the stroke width of the input-word image is estimated as the average width of the strokes in the second scan.

In order to estimate the stroke height, which is assumed to be average height of the vertical strokes in writing, a similar algorithm is used. However, the scanning procedure is applied in vertical direction. *Minimum height* is estimated instead of *maximum width*. In the second scan, those pixels whose run lengths are smaller than *minimum height* are discarded. Estimated stroke height is used in slant angle detection and upper baseline extraction, assuming small slant. It is slightly less then the actual one due to the slant.

3.2 Slant Angle Detection

Slant is the deviation of the strokes from the vertical direction, depending on writing style. In many handwriting recognition studies, slant correction is applied before segmentation and recognition stages. However, this correction produces serious deformation of characters, which may cause important information loss. In this study, we did not make any slant correction, but we used the slant angle in the segmentation stage by moving from top to the bottom of a word image in the slant direction.

Previous studies in the literature involve Hough transform [12], slanted histogram approach [9], and word contour analyzing [15]. In this study, the slant of a word is estimated by finding the average angle of near-vertical strokes extracted by three-by-three masks, which allows one-pixel deviation from the vertical line. This is calculated by extracting a chain of connected pixels representing the edges of strokes whose lengths are longer than the estimated stroke height. The mode orientation of those edges close to the vertical direction is used as an overall slant estimate for a word image (Fig. 2).

3.3 Baselines Extraction

Proper computation of baselines, which delimit the main body of the word, has significant meaning, especially, in holistic approaches. Locations of upper and lower baselines determine the existence of ascending and descending characters in a given word image. Baseline information can also be used in segmentation in order to avoid problems introduced by ascending and descending portions of the characters.

Most studies in the literature, first detect and correct the skew angle, then estimate the baselines using some heuristics, based on horizontal projection profile [18], [9]. One commonly used method for skew detection is to fit a line through a set of points assumed to lie on the baseline by minimizing the sum of least squares or absolute values distances. Local minima of the word image are, generally, chosen as the restricted set of points for detection of lower baseline. Therefore, some local minima need to be pruned to remove the ones that do not lie on the baseline such as the minima on the descending part of the characters. Most of the time the minimum of a descending character is relatively lower than the minimum of a normal character.

However, it is practically impossible to find a heuristic for correctly eliminating those minima, which spoil the baseline skew. Therefore, a pruning procedure does not work properly in many cases, resulting in an abnormal skew angle. In order to prevent this undesirable effect, a method, which gives a weight to each minimum with respect to the curvature of the stroke at the local minima, is proposed in [5], assuming a lower curvature on the minima of a descending character compared to that of a normal character. However, these curvatures depend highly on the writing style.

In this study, we propose a new baseline extraction algorithm, which uses a weighting approach based on the angles between the local minima in a least square estimation scheme. First, a preliminary centerline for each word image is determined by finding the horizontal line with the highest number of black pixel runs. Then, the local minima below the preliminary baseline are identified eliminating the ones on the ascending part (Fig. 3a). The goal is to find the best fit to the local minima with a high contribution from the normal characters and low contribution from descending characters. A weight is computed for each minimum by considering the average angle between that minimum and the rest of the minima. This approach assumes relatively small average angles among the minima of normal characters compared to the average angle between a descending minimum and normal minima,

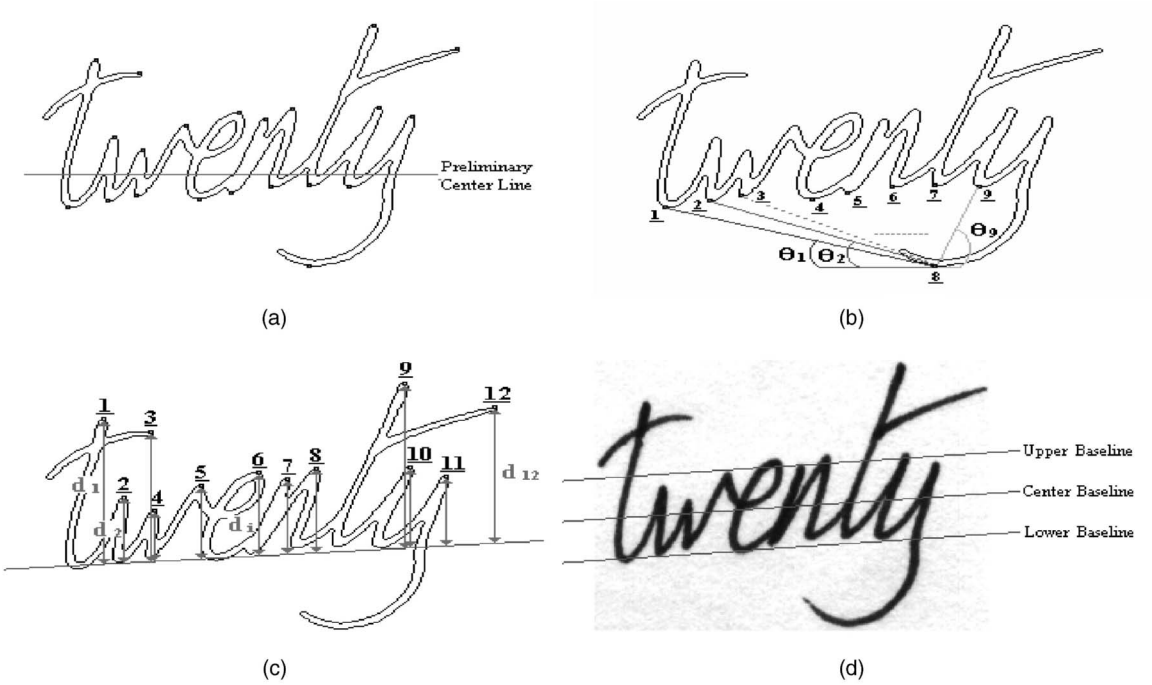

Fig. 3. Baselines extraction. (a) Local maxima and minima above and below the preliminary center line, respectively. (b) Angles between the minimum eight and the other minima. (c) Distances of local maxima from the lower baseline. (d) Upper, center, and lower baselines.

independent of the writing style. Finally, a line-fitting algorithm is performed over the weighted local minima. After the detection of lower baseline, upper baseline is estimated using the local maxima above the lower baseline. The following algorithm gives the steps used for the lower and upper baselines extraction method:

Baseline Extraction

1. Scan the binary word image horizontally and count the black pixel runs. Find the horizontal line, called preliminary center line, which contains the highest number of black pixel runs (Fig. 3a).

Lower Baseline Extraction:

2. Identify the local minima below the preliminary center line. For each minimum *i*, construct a list of the angles (θ_j) between that minimum and the other minima (Fig. 3b).
3. For each minimum *i*,
   a) Cluster θ_j into two classes C_1 and C_2 by using C-Means algorithm, where $1 \leq j \leq M$, $i \neq j$ and M is the number of local minima.
   b) Find the class C_k and its mean μ_k with the highest number of elements.
4. Find the lower baseline by fitting a line that minimizes the sum of weighted square distances:

$$\mathrm{E} = \sum_{j=1}^{M} \left(1 \Big/ \mu_j^2\right) (y_j - ax_j - b)^2, \qquad (1)$$

where *a* and *b* are the parameters to be estimated for the lower baseline.

Upper Baseline Extraction:

5. Identify the local maxima above the lower baseline and calculate the vertical distances (d_i) of each maximum from the lower baseline (Fig. 3c).
6. Prune the ones whose distance is lower than the estimated stroke height, in order to eliminate the spurious local maxima which belong to the main body of the writing.
7. Cluster the local maxima in two classes according to the distances (d_i) from the lower baseline.
8. Take the mean value of the class, which includes the local maxima with smaller distances and draw a parallel line to the lower baseline, which passes from that mean.

Center Baseline Extraction:

9. Find the center baseline as the parallel line with equal distance to the upper and lower baseline.

4 First Level Character Classification

In this stage, the proposed system decides on the existence of ascending and descending characters in a given word image. Although this information is not crucial for the system, it improves the accuracy of segmentation and recognition algorithms a great deal.

In most of the holistic approaches, ascending and descending characters are identified by using an empirical threshold as a percentage of the main body [9]. However, this threshold is subject to change for different data sets. In this study, the thresholds are determined with respect to the character size parameters estimated in the HMM training stage.

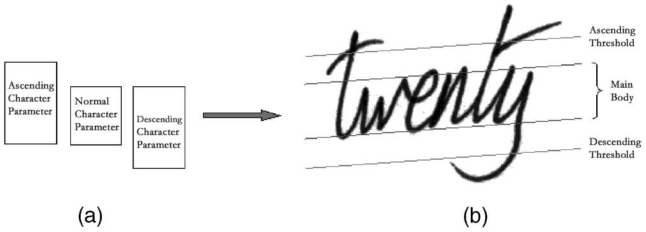

Fig. 4. First-level character classification. (a) Character size parameters estimated in HMM training. (b) Ascending and descending character detection.

The distance between the upper and lower baseline is taken as the height of the normal characters. Since the character size parameters contain the height-to-width ratios of normal, ascending, and descending characters, we can easily calculate the heights of the ascending and descending characters, assuming equal width for all the characters in a given word. Finally, the ascending/descending characters are detected by using threshold values, computed as the half way between the height of normal and ascending/descending characters. (Fig. 4).

5 Segmentation

Segmentation is the most crucial part of the cursive handwritten recognition problem, when an analytic approach is used for the recognition [4]. Small modifications on the segmentation algorithms result in serious improvements in the recognition rates.

The segmentation method proposed in this study, is motivated from the work of Lee et al. [13], where the character segmentation problem is defined as the problem of finding the shortest path in a graph defined over a segmentation region, minimizing the accumulated intensity. In [13], the segmentation regions are identified from the peaks of the horizontal projection profile in the gray-level image assuming machine printed characters. On the other hand, our aim is to develop a system for the handwritten characters, which may be both slanted and skewed. For this reason, the proposed method performs the segmentation by combining the characteristics of gray-scale and binary images. First, the segmentation regions are determined in the binary word image by using the contour of the writing. Then, an improved search process is applied to the segmentation regions on the gray-scale word image for determining the segmentation boundaries between characters.

5.1 Determination of Segmentation Regions

The segmentation regions carry the potential segmentation boundaries between the connected characters. Our first task is to partition each word image into stripes along the slant angle direction, each of which contains a potential segmentation boundary. Two rules are applied on the binary word image for identifying the segmentation regions:

Rule 1. A single maximum above the center baseline is an indication of a single character or a portion of a character.

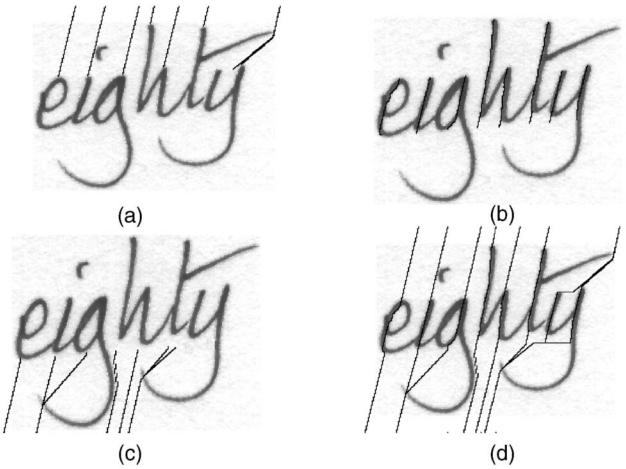

Fig. 5. Extraction of segmentation regions. (a) From character peaks to top. (b) From character peaks to lower baseline. (c) From lower baseline to bottom. (d) The segmentation regions for the overall word.

Rule 2. When there is more than one local maxima, the region between the two adjacent local maxima carries a potential segmentation boundary.

Although, the above rules are valid for most of the cases, they may yield false segmentation boundaries in the search algorithm mentioned below. The error is corrected in the final recognition of the whole word. Determination of the segmentation regions in each word image is accomplished in three steps:

Step 1. In this step, we attempt to draw a straight line in the slant angle direction from each local maximum until the top of the word image. However, there may be an ascender character on this path. In order to avoid a cut in the ascender character, this path is forced to pass from an end point of the ascender character. While going upward in slant direction, if any contour pixel is hit, this contour is followed until the slope of the contour changes to the opposite direction. An abrupt change in the slope of the contour indicates an end point. The direction of the contour following is selected as the opposite of the relative position of the local maximum with respect to the first contour pixel hit by the slanted straight line. After reaching the end point of the ascending portion of the character, a line is drawn from the maximum to the end point and path continues to go upward in slant direction until the top of the word image (Fig. 5a). If the algorithm does not reach any end point as in the case of connected "t"s in the word "tot", the slanted straight line breaks the connected letters at the point of the hit and reaches to the top of the word image.

Step 2. In this step, a path in the slant direction from each maximum to the lower baseline, is drawn. However, the algorithm avoids passing from the white pixels by selecting a black pixel, as long as there is one in either left or right neighborhood of the white pixels (Fig. 5b).

Step 3. The same process as in the first step is performed in order to determine the path from lower baseline to the bottom of the word image. In this case, our aim is to find the path, which does not cut any part of the descended character (Fig. 5c).

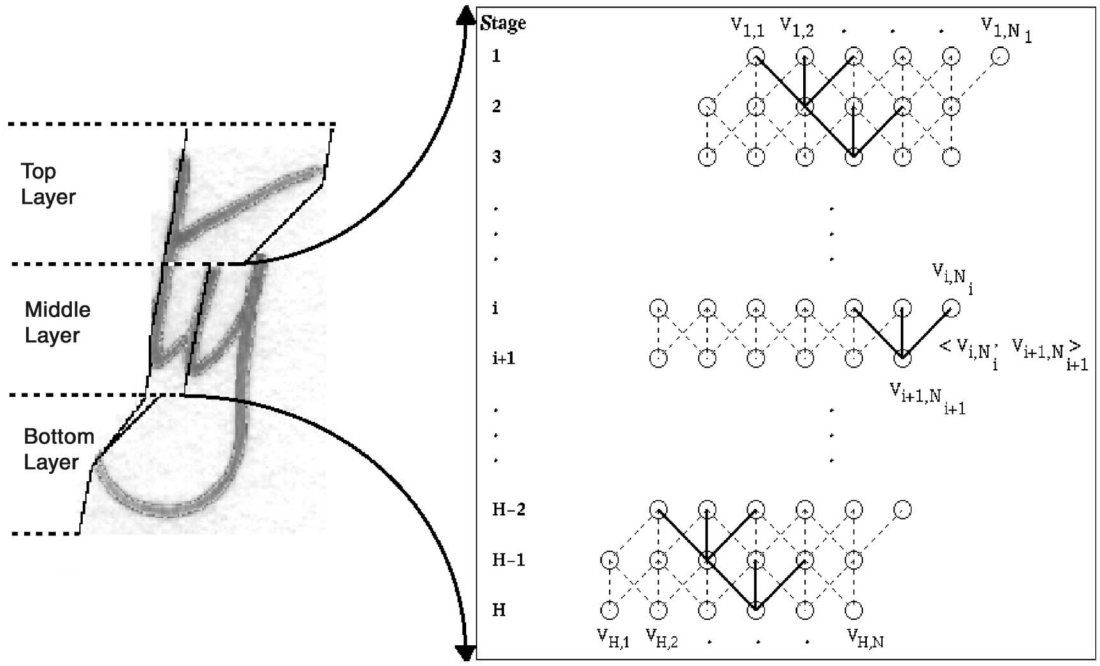

Fig. 6. Graph representation of a layer in a segmentation region.

Fig. 5d indicates the segmentation regions obtained by combining the results of Step 1, 2, and 3.

5.2 Search for Nonlinear Segmentation Paths

A search algorithm finds the shortest path from the top to the bottom of a word image, which separates the connected characters in the segmentation regions. The algorithm performs the segmentation on the gray-scale word image by combining the gray-scale and binary information.

Recall that the ascenders and descenders are labeled in the first-level classification stage. In case of an ascending or descending stroke, a segmentation region is partitioned into three almost horizontal layers (See Fig. 6) as follows:

- The top layer is above the piecewise linear line obtained by connecting the maxima of consecutive normal characters in the word image.
- The middle layer lies in between the lower baseline and the piecewise linear line, obtained above.
- The bottom layer is the region below the lower baseline.

Search for the optimum segmentation path is accomplished in three layers, separately.

Given a layer of the segmentation region of height H, let $G(V, A)$ be a multistage di-graph, where the vertices, $\mathrm{v_{ij}} \in \mathrm{V}_i$ represent the pixel coordinates and $V_i, \mathrm{i} = 1, \ldots \mathrm{H}$, corresponds to the set of vertices at ith row of the word image.

There are $\mathrm{N_i}$ vertices in each $\mathrm{V_i}$. $\mathrm{N_i}$ differs from row to row according to the distance between two successive segmentation region borders. The edges in G are of the form $< \mathrm{v_{ij}}, \mathrm{v_{i+1,k}} >$ for all $\mathrm{j} - 1 \leq \mathrm{k} \leq \mathrm{j} + 1$ and $1 \leq \mathrm{i} \leq \mathrm{H}$. Then, a layer in a segmentation region can be represented by a multistage di-graph as shown in Fig. 6.

The edges indicate the cost of a path between the vertices $\mathrm{v_{ij}} \in \mathrm{V_i}$ and $\mathrm{v_{i+1,k}} \in \mathrm{V_{i+1}}$ for some j and k. The cost of an edge $< \mathrm{v_{ij}}, \mathrm{v_{i+1,k}} >$ is determined by using the information extracted from both gray-level and binary image and is defined as follows:

$$a_{(i,j)(i+1,k)} = \left(\frac{H + (H - i)}{H}\right) \times I_{i+1,k} + SW \times C_{i+1,k}, \qquad 1 \leq i \leq H, \qquad j - 1 \leq k \leq j + 1, \tag{2}$$

where I_{ij} and C_{ij} are the gray values of the pixel (i, j) in gray-level and boundary image respectively, with zero corresponding to white and one corresponding to black. SW is the estimated stroke width, *i* is the *y* coordinate and j is the x coordinate of a pixel.

Note that, the above cost function forces the shortest path to pass through the white pixels without cutting any boundary of the characters if possible. The intensity values in gray-level are weighted by the y coordinate of the pixel. Between two pixels with the same intensity values, the pixel whose coordinate is lower than the other is given higher weight. If the path is required to cut any stroke in the segmentation region, it cuts the stroke, which is closest to the bottom (see Fig. 7a). The character contours in the boundary image are represented by black pixels and weighted by the estimated stroke width. The weight given to each boundary pixel enforces the path to cut the minimum number of strokes (see Fig. 7b). Therefore, the segmentation path is optimal when it goes through the common stroke, thus separating two joined characters. Note also that we constrain the possible vertices that can be reached from $\mathrm{v_{ij}}$ to $v_{i+1,j-1}$, $v_{i+1,j}$, and $v_{i+1,j+1}$. This avoids the cuts in the horizontal direction and the moves in the opposite directions. A dynamic programming algorithm then searches for the shortest path (segmentation path) from top to bottom rows. See also Fig. 8.

The goal is to find the minimum cost from the top row to the bottom row. Therefore, the problem of segmentation can

(a) (b)

Fig. 7. Correction of the segmentation errors by the weights given (a) in the gray-level image and (b) in the boundary image.

be represented as finding the shortest path, which minimizes the cumulative cost function,

$$COST = \sum_{1 \le i < H} a_{(i,j)(i+1,k)} \qquad j-1 \le k \le j+1. \tag{3}$$

Let $f_i(j)$ be the minimum accumulated cost at vertex v_{ij} in row i, and $Q_i(j)$ be the column of the vertex in V_{i-1} on the shortest path to the vertex v_{ij} in V_i. We define the path with the minimum accumulated cost as the nonlinear character segmentation path, searched by the following algorithm:

Segmentation Path Search Algorithm
Initialization: *for* $1 \le j \le N_1$,
$$f_1(j) = I_{1,j},$$
Recursion: *for* $2 \le i \le H$,

$$f_i(j) = \min_{j-1 \le k \le j+1} \{a_{(i-1,k)(i,j)} + f_{i-1}\}, \qquad 1 \le j \le N_i, \tag{4}$$

$$Q_i(j) = \arg \min_{j-1 \le k \le j+1} \{a_{(i-1,k)(i,j)} + f_{i-1}(k)\} \qquad 1 \le j \le N_i, \tag{5}$$

Terminaiton: $f^* = \min_{1 \le j \le N_H} \{f_H(j)\}$,
$m_H^* = \arg\min_{1 \le j \le N_H} \{f_H(j)\}$,
Backtracking: *for* $i = H-1, H-2, \ldots, 1$
$m_i^* = Q_{i+1}(m_{i+1}^*)$.

In the first step, $f_1(j)$ is initialized with the intensity of pixel $(1, j)$. Then, the accumulated distance $f_1(j)$ can be recursively evaluated at each stage. In the final row, when $i = H$, we have N_H accumulated distances, $f_H(j), j = 1, 2, \ldots, N_H$. The minimum accumulated distance f^* is the candidate for the shortest path. Now, the final task is to backtrack from m_H^* to the initial vertex, following Q_i. The complexity of this algorithm is $O(N)$, where N is the number of vertices $\sum_i^H N_i$, and H is the number of stages in the graph.

The segmentation paths obtained for each layer are appended to yield the overall segmentation path from top to bottom of the segmentation region. Note that, there is only one path in each segmentation region and this path is not allowed to leave the region.

6 HMM Shape Recognizer for Character Ranking

Most of the time, the segmentation algorithm proposed in the previous section either correctly segments a single character or over segments a character and yields a false partition. Under segmentation is very rare. In order to correct the fragmented characters, single and multiple

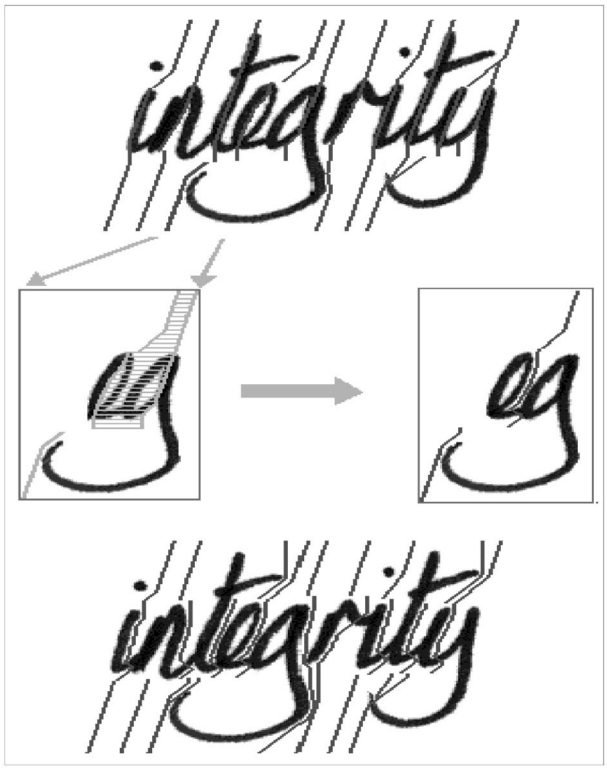

Fig. 8. Illustration of the segmentation process.

segments are considered as character candidates. First, a feature extraction scheme is employed and the character candidates are represented by one-dimensional string of codes. Then, discrete density HMM is used to label and rank the character candidates according to their HMM probabilities.

6.1 Feature Extraction

Feature extraction stage employs the binarized character candidates, extracted from the segmentation stage. Each candidate character is represented by a fixed size string of codes proposed by the authors of this study [1]. The representation is based on scanning the candidate characters in various directions and finding the median (center pixel) of the black pixel runs in each scan line. The scan lines are divided into several regions, which are coded by the powers of 2 (see Fig. 9). Each median is labeled by the code of the region where it belongs. Note that, the median of a black run indicates a point on the skeleton of the character for a particular direction. Each scan line is represented by summing the codes of the medians on that scan line. The observation sequence is then obtained by concatenating the string of codes for the scan lines in all directions. Fig. 9a, Fig. 9b, Fig. 9c, and Fig. 9d indicate horizontal, vertical, left and right diagonal scan lines, and the resulting observation sequences in each direction, respectively. The number of scan lines (n) in diagonal directions is equal to that of the ones in horizontal direction when n is even; otherwise, it is taken as $n-1$ to keep the symmetry with respect to the diagonal axis. Dots indicate the medians of the black runs in each scanning direction.

The above feature extraction method yields an observation sequence of fixed length, which is equal to the total

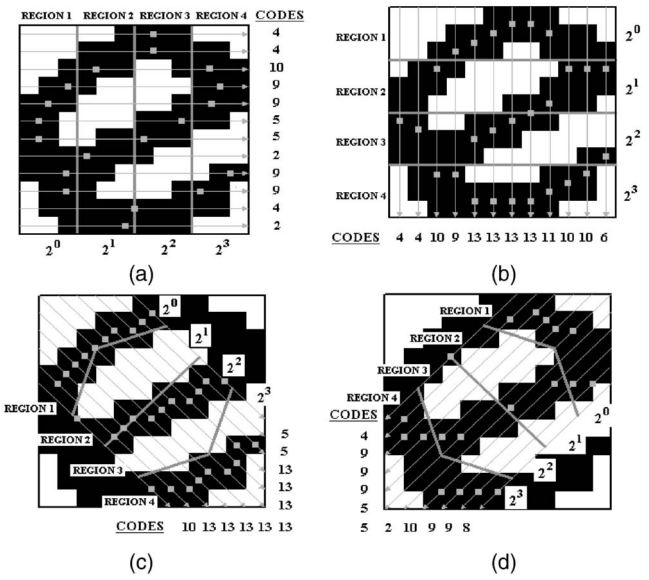

Fig. 9. Feature extraction of a sample letter "e" (a) horizontal, (b) vertical, (c) left diagonal, (d) right diagonal medians, and their corresponding codes. The observation sequence for this example is {4 4 10 9 9 5 5 2 9 9 4 2 4 4 10 9 13 13 13 11 10 10 6 8 13 13 13 13 13 13 13 5 5 4 9 9 9 5 5 2 10 9 9 8}.

number of scan lines in all directions. The observation sequence consists of the codes of an alphabet $\mathrm{V} = \{0, 1, \ldots, 2^{\mathrm{R}} - 1\}$ consisting of the integers obtained by adding all possible combinations of the codes of the regions.

The crucial part of the feature extraction stage is the estimation of the parameters, F, of the feature space, where $\mathrm{F} = (\mathrm{H}, \mathrm{W}, \mathrm{D}, \mathrm{R})$ is a four tuple, where, in turn, H is the height and W is the width of the character window, D is the number of scanning directions, and R is the number of regions in each scanning direction. In order to estimate the optimal parameter set, training data, which are formed by selecting the correctly segmented characters, are represented by the instances of F in a practical range as explained in Section 8.2. Then, for each instance, discrete density HMM described in the following section is used to train the character shapes in three training datasets for ascending, descending, and normal characters. The instance with the highest recognition rate on the training data is selected as the optimal parameter set.

6.2 HMM Training

In this study, the discrete density HMM is used to train the shapes of the characters and to rank the possible character labels. There are three basic categories for the HMM topologies for ascending, descending, and normal characters. Left-right HMM topologies are designed to suit the nature of the feature set. The number of states and the state transitions are selected by the method described in [1].

Given the initial class $j = 1, 2, 3$, for ascending, descending, and normal characters, the character class i is modeled by a four tuple as follows:

$$\lambda_{i/j} = (\mathrm{A_{i/j}}, \mathrm{B_{i/j}}, \mathrm{S_j}, \mathrm{F_j}),$$

where $\mathrm{A_{i/j}}$ and $\mathrm{B_{i/j}}$ are the state transition and observation probability matrices, respectively, $\mathrm{S_j}$ is the number of states and $\mathrm{F_j}$ is the feature parameter set for ascending, descending, and normal characters. The number of states are monotonically increased with the window size $H \times W$. Baum-Welch method is used in training [17]. For each instance of F, the optimal parameters, $\mathrm{A_{i/j}}$ and $\mathrm{B_{i/j}}$ are estimated. The parameter set $\mathrm{F_j}$, which maximize the recognition rate of the training set is fixed for the next stage of ranking.

Note that the training stage yields three different feature parameter sets for ascending, descending, and normal characters together with the HMM parameters. Note also that the optimal window size $H \times W$ changes by the size of writing, the resolution of the document image, and the type (ascending, descending, and normal) of the character.

6.3 HMM Ranking

In this stage, candidate characters, obtained by combining the segmented strokes are labeled by HMM. A ranked list of character labels with associated HMM probabilities are then generated for each combination of the segments in a word image.

Let a word image consist of K segments and assume that a character is divided into at most N segments. Then, for each segment $k = 1, \ldots, K$, a candidate character may be formed by merging $n = 1, \ldots, N$ consecutive segments. Our goal is to estimate the conditional probabilities $\mathrm{P}(O_{k,n}|\lambda_i)$, where $O_{k,n}$ represents the observation sequence of the candidate character obtained by merging n segments starting from the kth segment and λ_i represents the HMM model for character class i. (For simplicity, the indice j is dropped in the ranking stage since this information is not needed any more.)

In the ranking stage, the probability $\mathrm{P}(O_{k,n}|\lambda_i)$ of observing the coded segment $O_{k,n}$, given the character model, λ_i for every HMM class $i = 1, \ldots, C$, is calculated for the segments $k = 1, \ldots, K$ and $n = 1, \ldots, N$. For an unknown observation sequence, the output of HMM classifier gives us a ranked list of character labels with their associated probabilities $\mathrm{P}(O_{k,n}|\lambda_i)$, from the most probable character to the least probable one. This ordered list is used in the next stage of the word-level recognition. However, the complete list has no practical value. For this reason, the list is truncated after the recognition probabilities become smaller than a threshold, which is estimated in the training stage for each character class. The length, $\Gamma_{k,n}$ of the truncated list varies for each character candidate and it is much smaller than the number of distinct character classes, C. Fig. 10 indicates the ranked list of HMM probabilities for some of the segments in the word "eighteen." The table is constructed by the information on $-\log \mathrm{P}(O_{k,n}|\lambda_i)$. For this particular example $\Gamma_{k,n}$ is fixed to 3 for all the candidate characters. Note that the HMM classifier is likely to find relatively low recognition probabilities for the falsely segmented character candidates since the candidates do not exist in the training set.

Forward-backward method [17] is used for ranking the character labels. For each character class, this algorithm requires on the order of $S^2 O_L$ calculations, where S is the number of states and O_L is the length of the observation sequence. The fixed character size estimated in the training stage makes the recognition system size invariant. The fixed size observation sequence enables us to find an appropriate HMM topology, which provides a consistent comparison platform for HMM probabilities for all classes.

(a)

Segment		Character	Normalized	1st rank		2nd rank		3rd rank	
k	n	Image	Image	λ_i	$-\log P(O_{kn}\|\lambda_i)$	λ_i	$-\log P(O_{kn}\|\lambda_i)$	λ_i	$-\log P(O_{kn}\|\lambda_i)$
1	1			e	105	a	124	o	135
1	2			e	232	u	243	w	251
1	3			w	407	g	418	y	425
2	1			i	94	j	133	l	142
2	2			y	303	g	329	f	341
2	3			g	493	y	524	f	531
3	1			g	123	y	132	t	156
3	2			f	254	g	276	w	280
3	3			h	448	g	468	w	479
4	1			l	130	t	132	f	154
4	2			h	118	t	148	f	156
⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮	⋮

(b)

Fig. 10. Examples of character ranking of the word "eighteen." (a) Segmentation result. (b) Ranked list of character labels with associated HMM probabilities for $\Gamma_{k,n} = 3$.

7 Word-Level Recognition

The goal of this stage is to finalize the recognition in word-level by using a graph representing the whole word alternatives. This goal is achieved by computing the most probable word which exists in the lexicon. The system can also extend the lexicon if the probability of an unknown word exceeds a certain confidence level. In other words, the recognition is not restricted to the words in the lexicon.

Given a segmented word, let $G(V, A)$ be a multistage word graph, where the vertices represent the candidate segmentation boundaries between the characters and the edges represent the candidate character labels with the corresponding HMM probability measures. The word graph has the set of vertices $v_k \in V, k = 1, \ldots, K+1$, where K is the total number of segments obtained from the segmentation algorithm. The source v_1 and the sink v_{K+1} represent the left and right word boundaries, respectively. The edges in G are of the form $< v_k, v_{k+n} >_j$ and their associated costs are denoted as

$$a_{k,n}(j) = -\log P(O_{k,n}|\lambda_i), \qquad 1 \le j \le \Gamma_{k,n}, \qquad 1 \le n \le N, \tag{6}$$

and the character labels are denoted as

$$\arg(a_{k,n}(j)) = i, \qquad 1 \le i \le C, \tag{7}$$

where $\Gamma_{k,n}$ is the total number of edges between vertices v_k and v_{k+n} which is calculated in the HMM ranking stage and C is the number of characters in the alphabet.

Fig. 11b indicates the word graph of Fig. 11a, with the most probable candidates, taken from the list $(0 \le \Gamma_{k,n} \le 2, \ for\ 1 \le k \le K)$ for $N = 3$ and $K = 10$. The shortest path from source to sink can be obtained by minimizing the cumulative $-logP(O_{k,n}|\lambda_i)$ measure.

However, the sequence of characters lying on the edge of the shortest path may not form a valid word in the lexicon. At this point, lexicon information is necessary for achieving further improvements in word-level recognition rates.

For this purpose, most of the available methods match the unknown word graph against the words in the lexicon using dynamic programming technique and ranking every word in the lexicon [14], [10]. Then, the word with the highest rank is chosen as the recognition result. Recognition rates are quite satisfactory for small size vocabularies. However, as the size gets larger, the highest rate may not always correspond to the correct word. The time complexity of these algorithms is $O(K \times £)$ to match K segments to a lexicon of size £. In some studies, the lexicon is stored in a hash table [6] or in a trie structure [14]. If it is represented as trie, the complexity is of order $O(K \times \Lambda)$, where K is the number of segments and Λ is the number of nodes in the trie.

In this study, the lexicon is also stored in a trie data structure. However, the search process does not traverse the

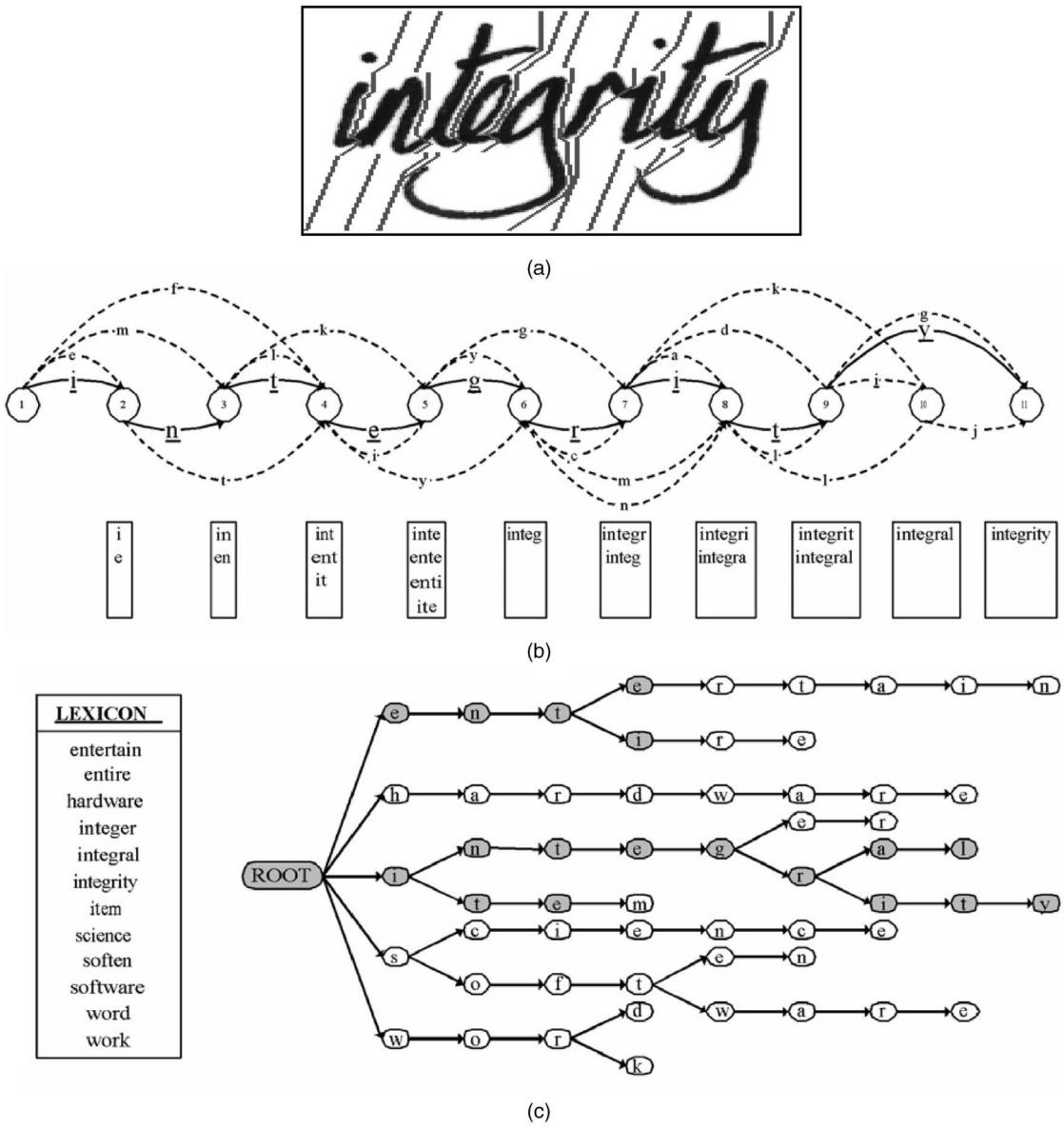

Fig. 11. Word-level recognition. (a) Segmentation paths of the word "integrity." (b) Word graph and the ordered list for each vertex. (c) The sample lexicon and its trie representation (shades indicate the valid paths in the word graph).

trie, but traverses the word graph, while maintaining pointers to the nodes of the subtrie, which is defined by the ranked list of HMM. For this purpose, at each vertex of the word graph an ordered list of partial results is stored with the corresponding pointers to the subtrie. Note that the size of the subtrie is very small compared to the size of the full trie because of the restriction imposed by HMM ranked list, which eliminates most of the word alternatives in the lexicon. (see Fig. 11c). Therefore, rather than selecting the sequence of characters with the shortest path, we select the shortest path, which correspond to a valid word in the lexicon. The algorithm avoids matching each entry in the lexicon against the word graph and reduces the complexity of the search process.

Formally speaking, let us define each node of the trie by the following structure,

$$struct(char\_index,\ parent\_pnt,\ child\_pnts(i)),\ \ 1 \le i \le C, \tag{8}$$

where *char_index* denotes the character assigned to each node, *parent_pnt* is the pointer to parent node, and *child_pnts* contains the list of pointers to the child nodes each of which is indexed by a character in the alphabet. Let us also define an ordered list at each vertex of word graph for storing the partial results as,

$$List_k = \{L_k(1), L_k(2), \ldots L_k(\Pi)\} \qquad 1 \le k \le K+1, \tag{9}$$

where $L_k(j)$ is the jth minimum accumulated cost in vertex k and Π is the size of the list. Each entry of $List_k$ also contains a pointer $(pnt(L_k(j)))$ to a node in the trie. Then, the minimum cost of the path, from source to vertex t, which corresponds to a valid subword, can be defined as

$$L_t(1) = \min_{p_J} \left(\sum_{<v_k, v_{k+n}> \in E(p_j), 1 \le i \le \Gamma_{k,n}} a_{k,n}(i) \right) \quad (10)$$
$$if\ \arg(p_j) \in LEX,$$

where p_j is any path from the source to vertex t and $E(p_j)$ is the set of edges on the path p_j, $arg(p_j)$ is the character sequence constructed by the arguments of $E(p_j)$, and LEX represents the lexicon. For $t = K + 1$, the cost of the best path, which corresponds to a valid word in the lexicon is computed as $L_{K+1}(1)$. The search process is summarized by the following algorithm:

Word Graph Search Algorithm

Initialization: $L_1(1) = 0$,
$pnt(L_1(1)) = ROOT$,

Recursion:

for $2 \le k \le K + 1$
  for $1 \le n \le N$
    for $1 \le t \le \Pi$
      temp_cost = $\min_{1 \le i \le \Gamma_{k-n,n}, 1 \le j \le \Pi} \{L_k - n(j) + a_{k-n,k}(i)\}$, (11)
      temp_pnt = $pnt(L_{k-n}(j)) \rightarrow child\_pnts (\arg(a_{k-n,k}(i)))$, (12)
      if temp_pnt <> NULL
        $temp\_list_n(t) = temp\_cost$ (13)
        $pnt(temp\_list_n(t)) = temp\_pnt$, (14)
  for $1 \le t \le \Pi$,
    $L_k(t) = \min_{1 \le n \le N}\{temp\_list_n(j)\}$, $1 \le j \le \Pi$, (15)
    $pnt(L_k(t)) = pnt(\min_{1 \le n \le N}\{temp\_list_n(j)\})$, $1 \le j \le \Pi$, (16)

Backtracking:

$temp\_pnt = pnt(L_{k+1}(1))$,
$while\ (temp\_pnt <> ROOT)$
  write_in_reverse_order (temp_pnt→ char_index),
  temp_pnt = temp_pnt→ parent_pnt.

Initially, $List_1$ is empty in the left boundary of the word (source) and at the root of the trie. The ordered list, $List_k$ is constructed recursively by using a set of temporary lists at each vertex k. There is a total of N temporary lists (recall that N is the maximum number of segments to make up a character) for each k, having size Π. The temporary list, $temp\_list_n$ consists of the paths until the vertex v_{k-n} plus the edges between the vertices v_{k-n} and v_k. In each iteration, the path with minimum cost is computed in (11) and the argument of the edge $< v_{k-n}, v_k >$ is searched in the next level of the trie by (12). If there is a hit, the cost and the argument of the path are inserted to the temporary list defined in (13) and (14). Otherwise, this subword is eliminated. Since both the lists $List_k$ $k = 1 \ldots K$ and the edges $a_{k,n}(j), j = 1 \ldots \Gamma_{k,n}$, are ordered with respect to their costs, the minimum cost of the sum $\{L_{k-n}(j) + a_{k-n,k}(i)\}$, can be evaluated sequentially.

After the creation of temporary lists, Π paths among the entries of N temporary lists with minimum costs are inserted to the ordered list of the corresponding vertex by the equations (15) and (16). At the sink, when $k = K + 1$, the first entry of the list $List_{K+1}$, which points to a leaf node in the trie, contains the best path. The unknown word is then found by backtracking the trie and writing the character index of each node in reverse order.

At each vertex, a temporary list is constructed at most in $\Gamma_{k,n} \times \Pi$ iterations and N temporary lists are used in order to store the partial results in the ordered list of the corresponding vertex. Considering the small size of N and $\Gamma_{k,n}$, the total complexity of the word-level recognition process is $O(K \times \Pi)$. Note that, the complexity of the search process is independent of the size of the lexicon.

8 Performance Test

The proposed scheme is implemented in UNIX environment with C programming language. Three sets of experiments are done to test the performance of

- Segmentation,
- HMM ranking, and
- Whole word recognition.

Unfortunately, there is not a standard database for cursive handwriting and it is very difficult to generate a complete set for testing the full performance of the proposed method. In this study, we employ handwritten database of Lancester-Oslo/Bergen (LOB) corpus used in [18], which contains single author cursive handwriting. In this dataset, 1,000 words with lower-case letters are segmented and used for HMM training and another disjoint set of 2,000 words are used for testing performance of the proposed system.

8.1 Segmentation

Given a word image, the result of the segmentation stage is assumed to be correct if there exist a path for that word from source to sink in the word graph. The correct segmentation does not always mean the extraction of a single character. It may divide a character like "u" and "w" into more than one segment, but it is not supposed to leave two distinct characters together. For example, a split in "u" is considered as correct segmentation since the split in characters that have more than one local maximum is assumed to be correct segmentation. These two regions are expected to be merged in the final stage of word-level recognition.

It is observed that correct segmentation depends mostly on the determination of the segmentation regions. Using the slant angle as we move from top to bottom of a word image contributes a great deal in finding the segmentation regions, which contain the correct segmentation path. In order to get rid of the spurious local maxima, a smoothing mask is used on the character boundaries. It is also observed that the smoothing has significant impact on finding the correct local maxima. There are three basic sources of segmentation error. Two percent comes from the extraction of segmentation regions, which do not carry a correct segmentation path. Less then 1 percent comes from the missing local maxima, which yields under segmentation. Finally, given a segmentation region, finding the false nonlinear segmentation path is less than 2 percent. The overall segmentation algorithm yields correct result with the rate of 95.5 percent. Approximately 70 percent of the segments consist of full characters, which means 30 percent oversegmentation.

8.2 HMM Ranking

Correctly segmented samples varying between 20 and 100 for each character class are extracted and used for

TABLE 1
Trie Size for the Lexicon with 40,000 Words

Trie Levels	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of nodes	396	2851	9787	16367	18868	17664	14580	10708	7184	4385	2458	1264	567	257	99

TABLE 2
Recognition Rates in Word Level (in Percent)

	TEST DATA SIZE	LEXICON SIZE			
		50	1000	30000	40000
LOB Dataset	2000	92.3	90.8	89.1	88.8

estimation of both feature extraction and HMM parameters. The optimal parameters of feature space F is obtained by varying $10 \leq \mathrm{H} \leq 30,\ 10 \leq W \leq 30,\ 2 \leq D \leq 4,\ 3 \leq R \leq 8$, and computing the highest recognition rate in the training data. The upper and lower bounds for search space is found empirically. For this particular training data, HMM estimates the optimum height and width lengths as 20×25, 20×30, and 20×30 for normal, ascending, and descending characters, respectively. Optimum number of scanning directions is estimated as 4 and each scan line is divided into five regions, for extracting the features. The number of HMM states for ascending, descending, and normal characters are taken as 30. The state transition matrix has only three nonzero diagonals.

The scheme for cursive handwritten recognition, presented in this study, embeds the isolated character recognition scheme of [1], based on the HMM recognizer. The recognition rates of the HMM recognizer with the proposed set of features are above 95 percent for various data sets of NIST Special Database 19. The detailed test results are reported in [1].

The performance of HMM ranking can be measured by the existence of correct characters in the ranked list. For each unknown candidate character, if the alternatives in the rank list contains the correct character, the HMM result is assumed to be correct. In order to test the performance of the HMM recognizer, correctly segmented characters are used. If the segmentation algorithm splits a character into more then one region due to the existence of more than one local maximum, we manually merge the regions to cover the whole character. Ninety-three percent of the characters are recognized correctly and take the first place in the HMM ranked list; whereas, 99 percent of them take among the first five in the ranked list. These results indicate the power of the segmentation and HMM shape recognizer proposed in this study.

8.3 Whole Word Recognition

In this study, we assume that each segmentation region contains at most one character and each character can be segmented into at most three segments which corresponds to $\mathrm{N} = 3$. These assumptions can be easily relaxed by defining more complicated word graphs. However, the tests on the data indicate that these assumptions are valid for the data set we use. Experiments indicate that HMM ranked list has the length of at most five characters, in all cases $(\Gamma_{k,n} \leq 5)$. The maximum size of the ordered lists, constructed at each vertex, is 100 for storing the partial results $(\Pi = 100)$.

In word-level recognition, the proposed system is tested on lexicons with various sizes. The lexicons are constructed by the randomly chosen words among the vocabulary used in [18]. The lexicon of each experiment is stored in a trie data structure at the beginning of the test process. Table 1 shows the number of nodes in each level of trie data structure until 16th level, which corresponds to the words with 16 characters.

The overall recognition rate of the whole system on word basis for various lexicon sizes is shown in Table 2. This result is better than the result reported by Senior and Robinson, which achieves 88.3 percent recognition rate with a lexicon of size 30,000. Another lexicon driven study [10] reports 73.8 percent recognition rate for 3,000 writer independent postal words with a lexicon of size 1,000.

9 Conclusion

The problem of the cursive handwriting is made complex by the fact that the writing is inherently ambiguous as the letters in a word are generally linked together, poorly written, and may even be missing. As a consequence, cursive script recognition requires sophisticated techniques, which uses a large amount of shape information and which compensates for the ambiguity by the use of contextual information.

The recognition strategies heavily depend on the nature of the data to be recognized. Small modifications and sensitive selection of the parameters in the laboratory environment provide high recognition rates for a given data set. However, a successful system reported in the literature may fail in real-life problems due to unpredicted change in data and violation of the initial assumptions in real life environment.

In this study, we try to maximize the amount of information to be retained in the data to a certain extent. The proposed method avoids most of the preprocessing operations, which causes loss of important information. However, the information, extracted in the global parameter estimation stage, is used on top of the original image as needed. We used the slant and skew angle information, but we did not make slant or skew angle correction. We used the binary information, but we did not rely only on the binary information. The ambiguities in segmentation and HMM recognition results are carried until we resolve the word graph, where we maximize the cumulative probability measures of HMM ranked list and we consider all possible segmentation alternatives.

One of the major contributions of this study is the development of a powerful segmentation algorithm. Utilization of the character boundaries, local maxima and minima, slant angle, upper and lower baselines, stroke height and width, andascenders and descenders improves the search algorithm of the optimal segmentation path, applied on a gray-scale image. This approach decreases the over-segmentation, a great deal.

Another contribution is the use of HMM training, not only for the estimation of model parameters, but also for the estimation of some global and feature space parameters. Also, HMM probabilities are used to measure the shape information and rank the candidate characters. One-dimensional representation of a two-dimensional character image increases the power of the HMM shape recognizer.

The proposed word-level recognition method checks the validity of the optimal path in the word graph and yields the

most probable valid word with an efficient and robust algorithm. Experiments indicate that increasing the size of the lexicon causes only a slight decrease in the recognition rate. The complexity of the algorithm is independent from the lexicon size. Also, adding or deleting a word from the lexicon does not require any training or rearranging the OCR system.

Combining the graph search and lexicon search procedures through the trie data structure increases the efficiency of the word-level recognition. This is a critical issue, especially for the large size lexicons.

In conclusion, the intensive research effort of the last couple of decades is slowly approaching to the end on the Optical Character Recognition era. We think that this is one of the studies, which fine tunes and completes the approaches presented in the previous studies to a certain extent, thus providing some improvements in the algorithms and recognition rates.

ACKNOWLEDGMENTS

The authors would like to thank Dr. I. Hakki Toroslu for his helpful comments and the anonymous reviewers for their constructive suggestions. In memory of Cemile Zehra Turker-Yarman and Ifakat Arica.

REFERENCES

[1] N. Arica and F.T. Yarman-Vural, "One Dimensional Representation Of Two Dimensional Information For HMM Based Handwritten Recognition," *Pattern Recognition Letters,* vol. 21, pp. 583-592, 2000.

[2] N. Arica and F.T. Yarman-Vural, "A New Scheme for Off-Line Handwritten Connected Digit Recognition," *Proc. Int'l Conf. Pattern Recognition,* pp. 1127-1131, 1998.

[3] A. Atici and F.T. Yarman-Vural, "A Heuristic Method for Arabic Character Recognition," *J. Signal Processing,* vol. 62, pp. 87-99, 1997.

[4] R.G. Casey and E. Lecolinet, "Strategies in Character Segmentation: A Survey," *Proc. Third Int'l Conf. Document Analysis and Recognition,* pp. 1028-1033, 1995.

[5] T. Caesar, J.M. Gloger, and E. Mandler, "Estimating The Baseline For Written Material," *Proc. Third Int'l Conf. Document Analysis and Recognition,* pp. 382-385, 1995.

[6] A. Dengel, R. Hoch, F. Hones, T. Jager, M. Malburg, and A. Weigel, "Techniques For Improving OCR Results," *Handbook of Character Recognition and Document Image Analysis,* H. Bunke and P.S.P. Wang, eds., pp. 227-254, 1997.

[7] S. Gopisetty, R. Lorie, J. Mao, M. Mohiuddin, A. Sorin, and E. Yair, "Automated Forms-Processing Software and Services," *IBM J. Research and Development,* vol. 40, no. 2, pp. 211-230, 1996.

[8] N. Gorski, V. Anisimov, E. Augustin, O. Baret, D. Price, and J.-C. Simon, "A2iA Check Reader: A Family of Bank Check Recognition Systems," *Proc. Fifth Int'l Conf. Document Analysis and Recognition,* pp. 523-526, 1999.

[9] D. Guillevic and C.Y. Suen, "Recognition of Legal Amounts on Bank Cheques," *Pattern Analysis and Applications,* vol. 1, no. 1, pp. 28-41, 1998.

[10] G. Kim and V. Govindaraju, "A Lexicon Driven Approach to Handwritten Word Recognition for Real-Time Applications," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 4, pp. 366-379, 1997.

[11] G. Kim, V. Govindaraju, and S.N. Srihari, "An Architecture For Handwritten Text Recognition Systems," *Int'l J. Document Analysis and Recognition,* vol. 2, no. 1, pp. 37-44, 1999.

[12] A. Kornai, K.M. Mohiuddin, and S.D. Connell, "Recognition of Cursive Writing on Personal Checks," *Proc. Int'l Workshop Frontiers in Handwriting Recognition,* pp. 373-378, 1996.

[13] W. Lee, D.J. Lee, and H.S. Park, "A New Methodology for Gray Scale Character Segmentation and Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 10, pp. 1045-1050, Oct. 1996.

[14] J. Mao, P. Sinha, and K. Mohiuddin, "A System For Cursive Handwritten Address Recognition," *Proc. Int'l Conf. Pattern Recognition,* pp. 1285-1287, 1998.

[15] S. Madhvanath, G. Kim, and V. Govindaraju, "Chaincode Contour Processing for Handwritten Word Recognition," *IEEE Trans. Pattern Recognition and Machine Intelligence,* vol. 21, no. 9, pp. 928-932, Sept. 1999.

[16] M. Mohamed and P. Gader, "Handwritten Word Recognition Using Segmentation-Free Hidden Markov Modeling and Segmentation Based Dynamic Programming Techniques," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 5, pp. 548-554, May 1996.

[17] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE,* vol. 77, pp. 257-286, 1989.

[18] A.W. Senior and A.J. Robinson, "An Off-Line Cursive Handwriting Recognition System," *IEEE Trans. Pattern Recognition and Machine Intelligence,* vol. 20, no. 3, pp. 309-322, 1998.

[19] M. Shridhar, G. Houle, and F. Kimura, "Handwritten Word Recognition Using Lexicon Free and Lexicon Directed Word Recognition Algorithms," *Proc. Fourth Int'l Conf. Document Analysis and Recognition,* pp. 861-866, 1997.

[20] S.N. Srihari, Y.C. Shin, Y. Ramanaprasad, and D.S. Lee, "A System To Read Names and Adresses on Tax Forms," *Proc. IEEE,* vol. 84, no. 7, pp. 1038-1049, 1996.

[21] S.N. Srihari and E.J. Keubert, "Integration of Handwritten Address Interpretation Technology into the United states Postal service Remote Computer Reader System," *Proc. Fourth Int'l Conf. Document Analysis and Recognition,* pp. 892-896, 1997.

[22] Ø.D. Trier and A.K. Jain, "Goal Directed Evaluation of Binarization Methods," *IEEE Trans. Pattern Recognition and Machine Intelligence,* vol. 17, no. 12, pp. 1191-1201, 1995.

[23] J. Wang and J. Jean, "Segmentation of Merged Characters by Neural Networks and Shortest Path," *Pattern Recognition,* vol. 27, no. 5, pp. 649, 1994.

Nafiz Arica received the BSc degree from the Turkish Naval Academy in 1991. He worked for the Navy as communications and combat officer for four years. In 1995, he joined theMiddle East Technical University (METU) where he received the MSc degree in computer engineering. His thesis was awarded the thesis of the year in 1998 at METU. He is currently a PhD candidate in the Computer Engineering Department and also a lieutenant in the Navy. His research interests include character recognition, content-based image representation. He is a student member of the IEEE.

Fatos T. Yarman-Vural received the BSc degree with honors in electrical engineering from the Technical University of Istanbul in 1973, the MSc degree in electrical engineering from Bogazici University in 1975, and the PhD degree in electrical engineering and computer science from Princeton University in 1981. From 1981 to 1983, she was a research scientist in Marmara Research Institute in Turkey. From 1983 to 1985, she was a visiting professor at Drexel University. From 1985 to 1992, she was a technical and deputy manager at Yapitel Inc. Since 1992, she has been a professor in the Computer Engineering Department of Middle East Technical University (METU). Her research area covers computer vision, image processing, pattern recognition, and artificial intelligence. She has been involved in teaching, consulting, and organizing conferences in these areas. She was the chair woman in the Computer Engineering Department between 1996-2000. Currently, she is assistant to the president at METU. She is a senior member of IEEE and a member of Turkish Informatics Foundation, Turkish Information Association, and chamber of Electrical Engineers in Turkey.