IEICE TRANS. INF. & SYST., VOL.E92–D, NO.5 MAY 2009

PAPER

A Lexicon-Driven Handwritten City-Name Recognition Scheme for Indian Postal Automation

Umapada PAL[†a)], **Kaushik ROY**[††], *Nonmembers*, *and* **Fumitaka KIMURA**[†††], *Member*

SUMMARY A lexicon-driven segmentation-recognition scheme on Bangla handwritten city-name recognition is proposed for Indian postal automation. In the proposed scheme, at first, binarization of the input document is done and then to take care of slanted handwriting of different individuals a slant correction technique is performed. Next, due to the script characteristics of Bangla, a water reservoir concept is applied to pre-segment the slant corrected city-names into possible *primitive components* (characters or its parts). Pre-segmented components of a city-name are then merged into possible characters to get the best city-name using the lexicon information. In order to merge these primitive components into characters and to find optimum character segmentation, dynamic programming (DP) is applied using total likelihood of the characters of a city-name as an objective function. To compute the likelihood of a character, Modified Quadratic Discriminant Function (MQDF) is used. The features used in the MQDF are mainly based on the directional features of the contour points of the components. We tested our system on 84 different Bangla city-names and 94.08% accuracy was obtained from the proposed system.

key words: *handwriting recognition, cursive script recognition, modified quadratic discriminant function, Indian script, postal automation*

1. Introduction

Recognition of handwritten words has been a popular research area for many years because of its various potential applications. Some of its potential application areas are postal automation, bank cheque processing, automatic data entry, etc. There are many research works towards handwritten recognition of Roman [1], [2], Japanese/Chinese [3] and Arabic scripts [4], [5]. Mainly two approaches (segmentation approach [6] and holistic approach [7]) are used for the purpose. A combination of these two approaches has also been used for word recognition [8]. Neural networks [1], hidden Markov model [9], MQDF [2], [10], [11], graph-based method [12] etc. are extensively used by researchers for handwritten word recognition [6].

Postal automation has been a topic of research interest for the last two decades and many published articles are available towards automation of non-Indian language documents [13]–[16]. At present systems are available for postal automation in several countries like USA, UK, Canada, Japan, Germany etc. But no postal automation system is available for India. System development towards postal automation for a country like India is more difficult than other countries because of its multi-lingual and multi-script behavior. In India there are 19 official languages, and 11 different scripts are used to write these languages. An Indian postal document may be written in any of these official languages.

Although many investigations have been made towards the recognition of isolated handwritten character and numeral of Indian scripts [17], [18], [27], to the best of our knowledge there exists only a few studies towards handwritten city-name recognition of Indian scripts [19], [20], [28]. In this present paper, we propose a lexicon-driven method for unconstrained handwritten city-name recognition of Bangla, the second most popular language in India and the fifth most popular language in the world.

The city-name recognition approach proposed in this paper is as follows. At first, binarization of a city-name is done and the slant correction of the individual city-name is performed. In Bangla most of the characters have a horizontal line at the upper part. When two or more characters sit side by side to form a word in Bangla, these horizontal lines generally touch and generate big regions (spaces). For example see Fig. 1. Here two characters form a word and they touch each other in the word. Because of this touching behavior, we use water reservoir based concept [21] for the segmentation of Bangla city-names into primitives. Each primitive ideally consists of a single character or a sub-image of a single character. A lexicon-

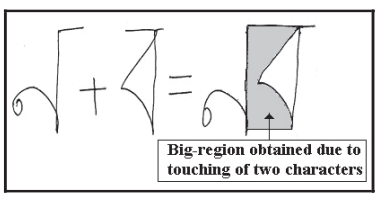


Fig. 1 Example of character touching in a Bangla word.

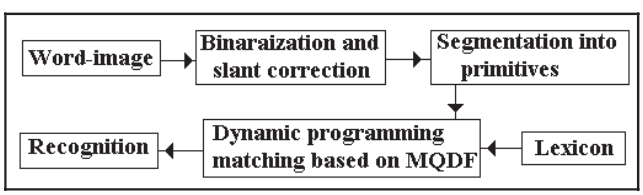


Fig. 2 Block diagram of the proposed system.

Manuscript received September 2, 2008.
Manuscript revised December 17, 2008.

[†]The author is with Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata–108, India.
[††]The author is with Dept. of Computer Science, West Bengal University of Technology, Kolkata–64, India.
[†††]The author is with Faculty of Engineering, Mie University, Tsu-shi, 514–8507 Japan.
a) E-mail: umapada@isical.ac.in

DOI: 10.1587/transinf.E92.D.1146

driven segmentation-recognition scheme is applied here for our recognition purpose. In order to merge these primitive components into characters and to find optimum character segmentation of a city-name, dynamic programming (DP) is applied using the total likelihood of characters as the objective function. To compute the likelihood of a character, a MQDF based on the directional features of the contour points of the components is used. Block diagram of the proposed system is shown in Fig. 2.

The organization of the rest of the paper is as follows. In Sect. 2, properties of Bangla characters are described. Principle of water reservoir concept is discussed in Sect. 3. Section 4 deals with slant correction and character presegmentation from Bangla city-names. Feature extraction is described in Sect. 5. Section 6 deals with segmentation-recognition using dynamic programming technique. Results and discussion are given in Sect. 7. Finally, conclusion is given in Sect. 8.

2. Properties of Bangla Script

Bangla, the secondmost popular language in India and the fifthmost popular language in the world, originated from Brahmi script. About 200 million people in the eastern part of Indian subcontinent speak in this language. Except Bangla, other languages like Assamese, Manipuri etc. are also written in Bangla script. Bangla is also the national language of Bangladesh. The alphabet of the modern Bangla script consists of 11 vowels and 39 consonants. These characters are called *basic characters*. The basic characters of Bangla script are shown in Fig. 3. The concept of upper/lower case is absent in this script. Writing style in Bangla script is from left to right. From the Fig. 3 it can be seen that most of the characters in Bangla script have a horizontal line at the upper part. We call this line as *head-line*.

In Bangla script a vowel following a consonant takes a modified shape. Depending on the vowel, its modified shape is placed at the left, right (or both) or bottom of the consonant. These modified shapes are called *modified characters*. Examples of Bangla vowel modified shapes are shown in Fig. 4. A consonant or vowel following a consonant sometimes takes a new orthographic shape, which we call as *compound character*. Examples of some types of Bangla compound character formations are shown in Fig. 5. Occasionally in Bangla, combination of two basic characters forms a new shape as shown in the first two rows of Fig. 5. In the third and fourth rows of Fig. 5 one of the constituent characters of the compound character retains its shape and the other constituent character reduces its size in the compound character. In the compound character shown in fifth row, two characters sit side by side in compounding, but the size of the first character is slightly reduced. Compound characters depicted in the sixth and seventh rows are formed by three basic characters, and shape of none of its constituent basic characters can be found in these compound characters. Compounding of four characters also exists in the Bangla script. Since the compound characters have different shapes, it is very difficult to recognize them. Because of such modified and compound characters there are about 280 characters in Bangla script [22].

A Bangla text line can be partitioned into three zones. The upper-zone denotes the portion above the head-line, the middle zone covers the portion between head-line and *base line*, the lower zone is the portion below base line. Different zones of a Bangla text line are shown in Fig. 6. Mostly a modified or a part of a modified character sits in the upper zone of a Bangla line. In the lower zone, mostly three vowel modifiers (shown in the 4th, 5th and 6th columns of Fig. 4) of Bangla script sit. We use the positional informa-

Fig. 3 Basic characters of Bangla alphabet (first eleven are vowels and rest is consonant).

Consituent Basic characters		Compound charcters
ক + র	=	ক্র
ঙ + গ	=	ঙ্গ
জ + ব	=	জ্ব
ন + ট	=	ন্ট
চ + ছ	=	চ্ছ
ক + ষ + ম	=	ক্ষ্ম
স + ত + র	=	স্ত্র

Fig. 5 Illustration of compound character formation in Bangla script.

Vowel	আ	ই	ঈ	উ	ঊ	ঋ	এ	ঐ	ও	ঔ
Modified shape	া	ি	ী	ু	ূ	ৃ	ে	ৈ	ো	ৌ
When attached to consonant ক	কা	কি	কী	কু	কূ	কৃ	কে	কৈ	কো	কৌ

Fig. 4 Bangla vowels and their corresponding modified characters. Shapes of these modifiers when attached with a consonant 'ক' are shown in the last row.

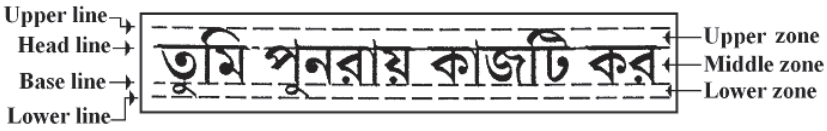


Fig. 6 Different zones of a Bangla text line.

tion of different characters and modifiers in our recognition scheme.

Because of the larger character set as well as complex-shaped structure of Bangla text, development of Bangla handwritten city-name recognition system is more difficult than Roman script.

3. Water Reservoir Principle

Water reservoir is a metaphor to illustrate the cavity region of a component. The water reservoir principle [21] is as follows. If water is poured from a side of a component, the cavity regions of the background portion of the component where water will be stored are considered as reservoirs of the component. These reservoirs are very useful for the segmentation of the city-names into primitives. Many researchers have used contour analysis or thinning for segmentation. Simple contour analysis and thinning based method will not work well because of touching behaviour and script characteristics of Bangla. Hence we use water reservoir concept for segmentation.

Some of the water reservoir principle based features used here are as follows.

Top (Bottom) reservoir: By top (bottom) reservoirs of a component we mean the reservoirs obtained when water is poured from top (bottom) of the component. A bottom reservoir of a component is visualized as a top reservoir when water will be poured from top after rotating the component by 180°.

Water reservoir area: By area of a reservoir we mean the area of the cavity region where water will be stored. The number of points (pixels) inside a reservoir is computed and this number is considered here as the area of the reservoir.

Water flow level: The level from which water overflows from a reservoir is called as water flow level of the reservoir (see Fig. 7).

Base-line: A line, passing through the deepest point of a reservoir and parallel to the water flow level is the reservoir base-line.

Height of a reservoir: By height of a reservoir we mean the depth of water in the reservoir. In other words it is the perpendicular distance between base-line and water flow level line.

Reservoir border points: The inner contour points obtained from the reservoir portion of a component are the reservoir border points.

Base area points: By *base-area* points of a reservoir we mean the border points of the reservoir having height less than R_L from the reservoir base-line. Here, R_L is the stroke width of the component. For a component, R_L is calculated as follows. A component is scanned both row-wise and column-wise. Suppose n different runs of lengths $r_1, r_2, \ldots . r_n$ with frequencies $f_1, f_2 \ldots .. f_n$, respectively, are obtained from the component by the scanning. Then R_L is considered as the run-length r_i having maximum frequency. In other words, R_L will be r_i if $f_i = \max(f_j)$, $j = 1, 2, \ldots .. n$.

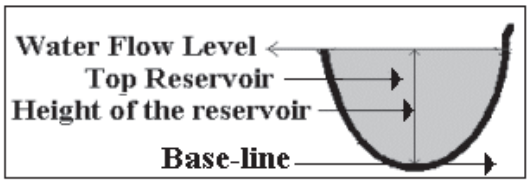


Fig. 7 A top water reservoir and its different features are shown. Water reservoir portion is marked with gray shade.

4. Slant Correction and Character Pre-Segmentation

4.1 Slant Correction

Slant correction is an important preprocessing step for handwriting recognition [2], [23]–[25]. In this paper the gray scale image of an input city-name is binarized by Otsu's algorithm [26]. The slant of the binary image is estimated and corrected using the method proposed by Kimura et al. [2]. The estimation of slant is computed based on the chain code information of the contour of the city-name image (different chain codes 0, 1, 2, and 3 are shown in Fig. 8 (a)). Slant angle θ is estimated based on the Eq. (1).

$$\theta = \tan^{-1}\left(\frac{n_1 + n_2 + n_3}{n_1 - n_3}\right) \quad (1)$$

where n_i is the total number of chain code elements at an angle of $i \times 45°$ ($i = 0, 1, 2, 3$, are the different chain codes). Here $n_1 + n_2 + n_3$ is the total value in vertical as well as two slanted directions and $n_1 - n_3$ is the change in horizontal direction as shown in Fig. 8 (b). Since the number of horizontal chain code element n_0 does not interfere in slant, we ignored n_0 in slant estimation. An example of a Bangla city-name and its slant corrected version is shown in Fig. 9.

4.2 Character Pre-Segmentation

To find optimal character segmentation by the segmentation-recognition scheme using dynamic programming, we pre-segmented the city-names into primitives. As mentioned earlier, when two or more characters sit side by side to form a word in Bangla text, head-lines of the characters touch at the upper part and generate big regions (as shown in

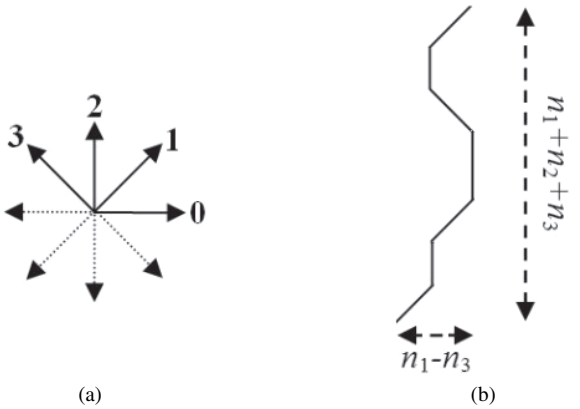


Fig. 8 (a) Different chain codes used (b) a segment and the representation of $n_1 + n_2 + n_3$ and $n_1 - n_3$ on this segment are shown.

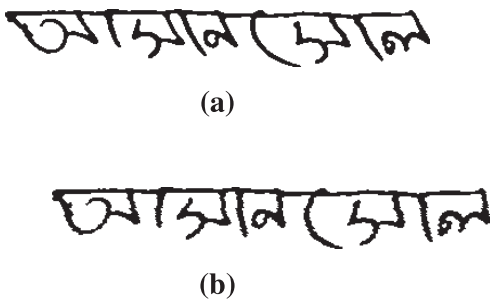


Fig. 9 (a) Input city-name (b) slant corrected version of the input city-name.

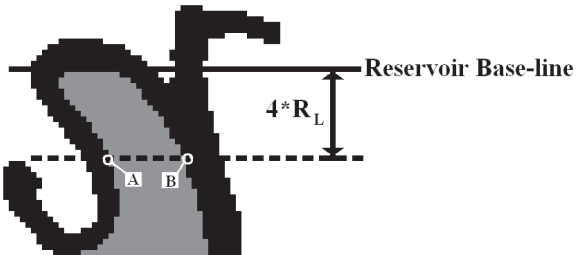


Fig. 10 Candidate border point detection. Here two candidate border points are shown by small circles and marked as A and B.

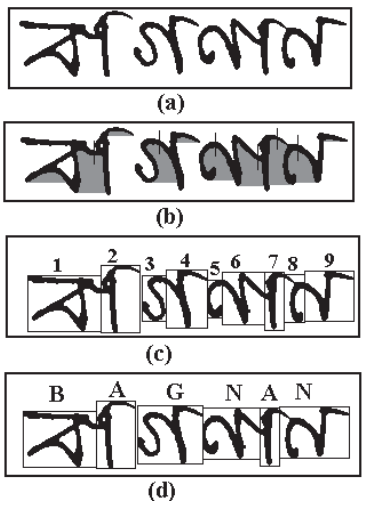


Fig. 11 Example of character segmentation from a city-name. (a) An input city-name (b) bottom reservoirs (marked by gray shade) obtained from the characters of the city-name. Pre-segmentation columns are marked by small vertical lines (c) Individual components are marked by disjoint boxes and they are numbered (d) optimum character segmentation. Here segmented individual characteres and their respective codes are shown.

Fig. 1). Based on such touching nature of Bangla characters in a word, we use a novel character pre-segmentation technique based on water reservoir concept.

For pre-segmentation we find bottom reservoirs from an input city-name. To detect segmentation point of a reservoir of a component, we first find two *candidate border points* of the reservoir. Computation of candidate border points is done as follows. From the reservoir base-line we consider the portion of the reservoir upto height $4 * R_L$ and we note two border points of the reservoir at this height. These two border points are considered as candidate border points. For illustration see Fig. 10 where two candidate border points are shown and marked as A and B. Let the coordinates of these two candidate border points A and B be (x_l, y_l) and (x_r, y_r), respectively. Here y is the vertical axis and considered as row and x is the horizontal axis and considered as column. We scan the component vertically for each of the columns between x_l and x_r and find the number of crossing in each of the columns. The column from which we get minimum number of crossing is considered for pre-segmentation. If we get two or more such columns with minimum number of crossing then the column having minimum stoke width is considered for segmentation. Please note that when two components touch, the stroke width of the touching portion is lower than the neighboring part. Based on this property, we decided to use the column having minimum stroke width for segmentation.

If the height of a reservoir is less than $4 * R_L$ then we consider the portion of the entire reservoir for candidate border points detection and in this case the candidate border points lie on the water flow level.

The pre-segmentation columns obtained from the city-name shown in Fig. 11 (a) are marked by small vertical lines in Fig. 11 (b). Please note that all the reservoirs are not considered for pre-segmentation. We compute the median of the heights of different reservoirs obtained in a city-name and those reservoirs having height greater than $0.8 *$ median are considered for segmentation. Because of the structural shape of some characters in Bangla, some small reservoirs may be obtained where segmentation should not be done and we use this threshold to ignore the segmentations of such small reservoirs. In character pre-segmentation our aim was to segment a city-name into individual characters as much possible avoiding much over segmentation.

After detection of pre-segmentation columns from an input city-name image, the image is split vertically at each pre-segmentation column and is separated into horizontally

non-overlapping zones. A connected component analysis is applied to the split image to detect the bounding boxes enclosing each connected component. Connected components in the split city-name image and their bounding boxes are shown in Fig. 11 (c). These boxes are numbered (from left to right) and these numbers are shown in Fig. 11 (c). For numbering, boxes are sorted from left to right according to the location of their centroids. If two or more boxes have the same x coordinates (as mentioned earlier x is the horizontal axis here) of their centroids, they are sorted from top to bottom. Numbers at the top of the boxes in Fig. 11 (c) show the order of the sorted boxes. These connected components are regarded as primitive segments and each of which corresponds to a full character or a part of a character.

There exists only a few research works towards Bangla word/city-name recognition [27], [28]. Some methods first segment Bangla words into characters then individual segmented characters are recognized. Main drawback of such method is proper character segmentation in Bangla because of the presence of modified and compound characters. To overcome such limitation on segmentation, we propose this method where characters are initially pre-segmented using water reservoir concept and then pre-segmented city-names are recognized using dynamic programming. To get idea of the effectiveness of our pre-segmentation method we performed some experiments. We applied pre-segmentation techniques discussed in [2] and noted that our proposed water reservoir based pre-segmentation showed better recognition results (comparison results are given in the last paragraph of Sect. 7.8).

5. Feature Extraction

Histograms of direction chain code of the contour points of the components are used as features for recognition [2]. For recognition we use 64 dimensional features and the feature extraction procedure is described below.

At first, the bounding box is divided into 7×7 blocks (as shown in Fig. 12 (c)). In each of these blocks the direction chain code for each contour point is noted and frequency of direction codes is computed. Here we use chain code of four directions only as shown in Fig. 8 (a). [direction 0 (horizontal), 1 (45 degree slanted), 2 (vertical) and 3 (135 degree slanted)]. Thus, in each block, we get an array of four integer values representing the frequencies of chain code in these four directions. These frequencies are used as feature. Histogram of the values of these four direction codes in each block of a Bangla character is shown in Fig. 12 (d). Thus, for 7×7 blocks we get $7 \times 7 \times 4 = 196$ features. To reduce the feature dimension, after the histogram calculation in 7×7 blocks, the blocks are down sampled into 4×4 blocks, with a Gaussian filter [29] of size 5×5. The standard deviation of the Gaussian is 1.81, which is selected so that the original sampling points contribute to final features as equally as possible. The positions of sampling points are located every two horizontal/vertical blocks starting from the block at left upper corner. As a result we have 64 ($4 \times 4 \times 4$) features for recognition. Histogram of these values of all the four directions obtained after down sampling is shown in Fig. 12 (e). The feature vector is normalized and a power transformation with 0.5th power is applied for each component [2], [23].

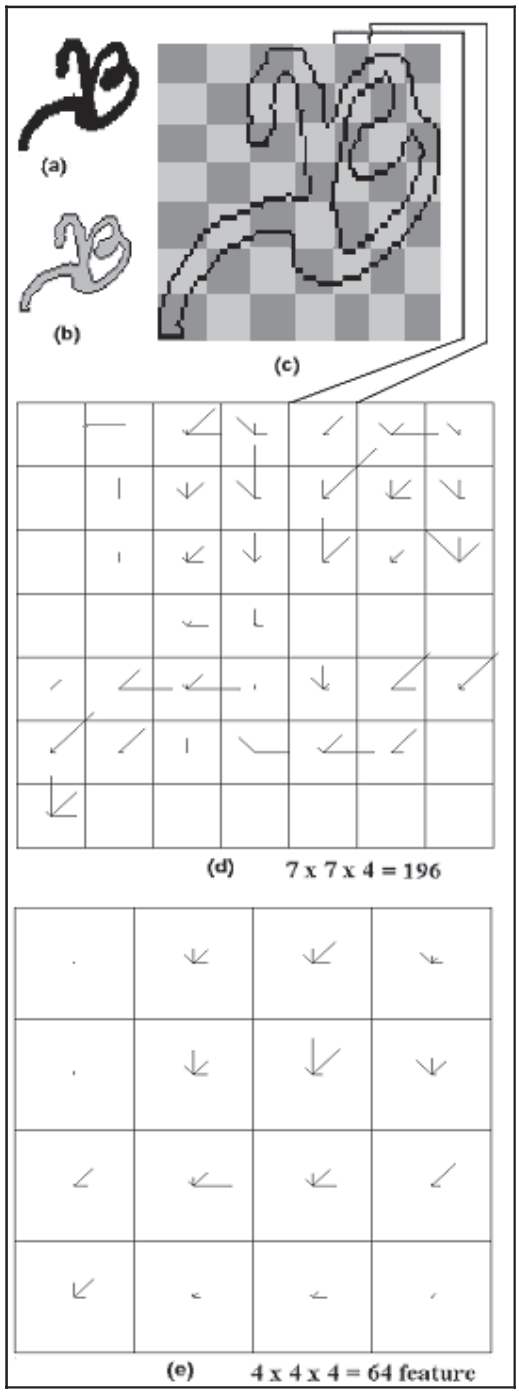


Fig. 12 Example of feature extraction (a) A Bangla character. (b) Contour of the character. (c) 7×7 segmented blocks shown in the zoomed version of the image shown in (b). (d) Block-wise chain code histogram of the contour points. (e) Histogrtam of chain code direction of the contour points after down sampling into 4×4 blocks from 7×7 blocks.

One critical point in segmentation-recognition techniques using dynamic programming is the speed of

feature extraction, because the correct segmentation points have to be determined in optimization process with respect to the total likelihood of the resultant characters. The use of the cumulative chain code histogram enables one to realize high-speed feature extraction. Border following and chain coding are performed only once to an input city-name image, and the chain code feature vector of a rectangular region including one or more boxes is extracted by a small number of arithmetic operations for high-speed feature extraction [2].

6. Segmentation-Recognition Using Dynamic Programming

The core of a dynamic programming algorithm is the module that takes a city-name image, a string to incorporate contextual information, and a list of the primitives from the city-name image and returns a value that indicates the confidence that the city-name image represents the string. Given a lexicon city-name, the primitive segments of the city-name image are merged and matched against the characters in the lexicon city-name so that the average character likelihood is maximized using dynamic programming.

6.1 Markov Chain Representation

The number of the boxes (primitives) of an input city-name image is usually 1.2 to 2 times as many as the number of characters in the city-name image. In order to merge these primitive components into characters and find the optimum character segmentation, dynamic programming (DP) is applied using the total likelihood of characters as the objective function [2], [23]. The ASCII lexicon of possible city-names is utilized in the process of dynamic programming to incorporate contextual information. The likelihood of each character is calculated using the MQDF. To apply the DP, the boxes are sorted from left to right according to the location of their centroids as mentioned earlier. If two or more boxes have the same x coordinates of their centroids, they are sorted from top to bottom. Numbers at the top of the boxes in Fig. 11 (c) show the order of the sorted boxes. It is worth observing that the box segmentation and the box sorting process reduce the segmentation problem to a simple Markov process, in most cases. For example, the box 1 corresponds to character "B" of the Bangla city-name *BAGNAN*, box 2 corresponds to character "A", boxes 3 and 4 correspond to character "G", boxes 5 and 6 correspond to character "N", box 7 corresponds to character "A", and boxes 8 and 9 correspond to character "N". See Fig. 11 (d) where characters' codes of this city-name image are given. These assignments of boxes to character are represented, for example, by
B A G - N - A N -
where "-" is used to denote the continuation. The assignment is also represented, for example, by

	B	A	G	N	A	N
i -->	1	2	3	4	5	6
$j(i)$ -->	1	2	4	6	7	9

where i denotes the letter number, $j(i)$ denotes the number of the last box corresponding to the i-th letter. Note that the number of the first box corresponding to the i-th letter is $j(i-1)+1$. Given $[j(i), i = 1, 2, .., n]$ the total likelihood of characters is represented by

$$L = \sum_{i=1}^{n} l(i, j(i-1)+1, j(i)) \tag{2}$$

where $l(i, j(i-1)+1, j(i))$ is the likelihood for i-th letter.

The optimal assignment (the optimal segmentation) that maximizes the total likelihood is found in terms of the dynamic programming as follows:
The optimal assignment $j(n)^*$ for n-th letter is the one such that

$$L^* = L(n, j(n)^*) = Max\, L(n, j(n)) \tag{3}$$

where $L(k, j(k))$ is the maximum likelihood of partial solutions given $j(k)$ for the k-th letter, which is defined and calculated recursively by

$$\begin{aligned} L(k, j(k)) &= \underset{j(1),j(2).....j(k-1)}{Max} \left\{ \sum_{i=1}^{k} l(i, j(i-1)+1, j(i)) \right\} \\ &= \underset{j(1),j(2)...j(k-1)}{Max} \Bigg[l(k, j(k-1)+1, j(k)) \\ &\quad + \sum_{i=1}^{k-1} l(i, j(i-1)+1, j(i)) \Bigg] \\ &= \underset{j(k-1)}{Max} \Bigg[l(k, j(k-1)+1, j(k)) \\ &\quad + \underset{j(1),j(2)\cdots\cdots j(k-2)}{Max} \left\{ \sum_{i=1}^{k-1} l(i, j(i-1)+1, j(i)) \right\} \Bigg] \\ &= \underset{j(k-1)}{Max} \Big[l(k, j(k-1)+1, j(k)) + L(k-1, j(k-1)) \Big] \end{aligned} \tag{4}$$

and $L(0, j(0)) = 0$ for $j(0) = 1, 2 \ldots .., m$ (5)

Starting from (5), all $L(k, j(k))$'s are calculated for $k = 1, 2, \cdots\cdots, n$ using (4) to find $j(n)^*$ using (3). The rest of $j(k)^*$'s ($k = n-1, n-2, \ldots, 1$) are found by back tracking a pointer array representing the optimal $j(k-1)^*$'s which maximizes $L(k, j(k))$ in (4).

6.2 MQDF for Character Likelihood

Character likelihood is calculated by the following MQDF [10].

$$\begin{aligned} g(X) = &\left\{ \left|X - \hat{M}\right|^2 - \sum_{i=1}^{k} \frac{\lambda_i}{\lambda_i + h^2} \left[{\phi_i}^T (X - \hat{M}) \right]^2 \right\} \Big/ h^2 \\ &+ \ln \left[h^{2(n-k)} \prod_{i=1}^{k} (\lambda_i + h^2) \right] \end{aligned} \tag{6}$$

where X denotes the input feature vector, $\hat{M}$ denotes the sample mean vector for each character class, and λ_i and ϕ_i denote the eigenvalues and eigenvectors of the sample covariance matrix. k is the number of eigenvalues considered here and n is the feature size. We set same values of h^2 and k as was used in [2], i.e. k is set to 20 and h^2 to $3/7 * \sigma^2$, where σ^2 is the mean of eigenvalues λ_i's over i and character classes.

Given a feature vector, $g(X)$ is calculated for a character class specified by a city-name lexicon. It may be noted that $g(X)$ is negative (minus) log likelihood and it is equal to negative of $l(i, j(i-1)+1, j(i))$.

6.3 Length Estimation of Input City-Name

Estimation of an input city-name length is helpful to reduce the number of possible lexicon city-name and speed up the city-name recognition process. It also reduces the possibility of confusion between city-names with different length. The upper bound (U_b) and lower bound (L_b) of the length of an input city-name are estimated and the lexicon city-names not having the length within the interval $[U_b, L_b]$ are skipped and not used in the segmentation recognition process. Therefore, under-estimate (over-estimate) of the upper bound (lower bound) excludes the possibility of correct city-name recognition, while tighter bounds are preferable to achieve higher speed and accuracy.

The U_b and L_b of an input city-name are estimated as follows depending on the result of the pre-segmentation. The city-name image is split vertically at each pre-segmentation column (as discussed in Sect. 4.2) to have horizontally non-overlapping zones. Smaller zones (those having width smaller than $0.5 \times R_L$, R_L is discussed in Sect. 3) are disregarded and considered as a part of their preceding zones. The number of zones obtained is the value of the upper bound U_b. The lower bound L_b is determined as

$$L_b = \text{trunc}(U_b/2)$$

where trunc(x) denotes the integer part of x.

Because of the script characteristics of Bangla sometimes we may get some double touching characters in an input city-name image. To take care of such city-names we considered the lower and upper bound of length as $L_b - 2$ and $U_b + 2$, respectively, for our experiment.

6.4 Use of Positional Information for Recognition

In Bangla script, most of the basic characters sit only in the middle zone of a text line, and most of the modifiers sit either in the upper or lower zone of a text line (different zones of a text line is discussed in Fig. 6). As a result, topmost and bottommost points of a Bangla character/modifier lie in any of the four reference lines: upper line, head-line, base line and lower line (see Fig. 6 where these lines are shown). We use the positional information of the topmost and bottommost points of a character/modifier in our system to reduce the mis-recognition rate. Since topmost and bottommost points of a character/modifier lie in any of the four reference lines, the y-coordinates of topmost and bottommost points of different characters/modifiers of a city-name generally produce four different clusters. The separability of these clusters is better when characters of the city-name are correctly segmented and recognized and it is worse when mis-segmentation or mis-recognition occurs. The *separability measure* (P_S) is defined by

$$P_S = S_B/(S_B + S_W)$$

where S_B and S_W are between-class and within-class variance of the y-coordinates of the clusters, respectively. This measure takes the value between 0 and 1, and the maximum value is achieved when within-class variance is zero i.e. position of all characters exactly match to the reference lines. The separability measure P_S is calculated after the segmentation-recognition using dynamic programming for each lexicon city-name. The clustering of the y-coordinates are performed using the result of the character segmentation and recognition. At first, based on the recognition result, characters with/without ascender/descender are checked and then the clusters of y-coordinates are detected based on whether each character has ascender/descender or not. Calculation of this separability measure requires only the y-coordinates of the uppermost and lowermost positions of the characters of a city-name and it does not require the different reference lines, the detection of which is not easy for a handwritten city-name because of writing styles of different individuals. Because of this advantage we use this separability measure in our scheme.

6.5 Confidence Value Computation

Depending on the optimal likelihood and separability measure we compute the confidence value (C_V) of an input city-name and the confidence value is defined by

$$C_V = \frac{L^*}{n} + \alpha \times P_S$$

Here n is the number of characters in the input city-name, L^* is the optimal total likelihood obtained from Eq. (3) of Sect. 6.1, P_S is the separability measure obtained from Sect. 6.4, and α is a constant used as coefficient of separability measure P_S. The value of α is considered as 30 from the experiment using the test samples of all 8625 city-names. To decide the value of α we computed word recognition accuracy using different values of α between 0 and 50 with an increment 10. We noted that maximum recognition rate was obtained when $\alpha = 30$ and hence we considered α as 30.

For an input city-name we compute the C_V for the lexicon words having length within the upper and lower bound of the length of the input city-name. So if there are N such lexicon words, we get N confidence values for an input city-name. The lexicon city-name from which we get highest confidence value for an input city-name, we recognize the

input city-name as that lexicon city-name. To obtain a lexicon string, each character of Bangla has been given a code (ID) and that coding scheme is discussed in [22]. Using this character codes lexicon string is generated.

We used a rejection scheme in our experiment and the rejection was done based on the following criteria (i) optimal likelihood value of the best recognized city-name is lower than T_1, and (ii) difference of the optimal likelihood values of the best and the second-best recognized city-names is less than T_2. If for an input sample, any one of the above criteria is true then the sample was rejected. Values of the parameters T_1 and T_2 used in our experiment are given later.

7. Experimental Results and Discussion

For the experiment of our city-name recognition scheme we collected 8625 handwritten samples of 84 Indian city-names written in Bangla script. We collected postal data from zonal post office of Kolkata city and city-names having higher frequency of occurrence in these collected data were considered here for the experiment. Number of samples of each city-name was at least 100. These data were collected from the images of handwritten address block of Indian postal documents as well as from some individuals of different professionals like students, researchers, businessmen etc. The images were scanned at 300 DPI and stored in tagged image file (TIF) format.

We have used 5-fold cross validation scheme for recognition result computation. Here database was divided into 5 subsets and testing was done on each subset using other four subsets for training. The recognition rates for all the test subsets were averaged to calculate recognition accuracy. Note that for training we manually segmented the city-names into characters.

For recognition result computation we used different measures and they are defined as follows:

$$\textit{Recognition rate} = \frac{N_C * 100}{N_T}$$

$$\textit{Error rate} = \frac{N_E * 100}{N_T}$$

$$\textit{Reject rate} = \frac{N_R * 100}{N_T}$$

$$\textit{Reliability} = \frac{N_C * 100}{N_E + N_C}$$

Where N_C is the number of correctly classified city-names, N_E is the number of misclassified city-names, N_R is the number of rejected city-names and N_T is the total number of city-names tested by the classifier. Here $N_T = N_C + N_E + N_R$.

7.1 Global Recognition Results

From the experiment we noted that the city-name recognition accuracy of the proposed scheme was 94.08%, when no rejection was considered. Also, from the experiment we noted that 96.74% (97.55%) accuracy was obtained when first two (three) top choices of the recognition results were considered. Details results with different choices are shown in Table 1. From the Table 1 it can be noted that recognition accuracy increases 2.66% when we considered two top choices instead of one top choice. From the Table it can also be noted that recognition accuracy increases only 0.26% when we considered five top choices instead of four top choices. We analyzed the results and noted that increment of 2.66% accuracy, when we considered two top choices instead of one top choice, was mainly due to the presence of some similar shape characters that make two city-names similar in shape.

Table 1 City-name recognition results based on different top choices.

Number of choices from top	Accuracy
Only 1 choice	94.08%
first 2 choices	96.74%
first 3 choices	97.55%
first 4 choices	98.12%
first 5 choices	98.38%

7.2 Rejection versus Error Rate Computation

From the experiment we also computed city-name recognition results with different rejection rates. We noted that 98.01% reliability with 1.82% error was obtained when 8.43% rejection was considered in the proposed system. 99.90% reliability with 0.07% error was obtained from the system when 31.66% data were rejected. City-name recognition reliability with different rejection rates is given in Table 2. To get the idea of the value of the parameters T_1 and T_2 used in our experiment, we have provided their values in the Table. The values of T_1 and T_2 that minimize the rejection rate for a given reliability are chosen here.

7.3 Recognition Accuracy versus Lexicon Sizes

To get idea about the results on different lexicon size, we also computed accuracies of our scheme based on different lexicon sizes. To get larger lexicon size, we added another 116 city-names in our lexicon. We consider lexicon of sizes 10, 20, 50, 100, 150 and 200 city-names. City-names for lexicon size 10 were chosen by considering 20th, 40th, 60th ... city-names from the full city-name lexicon of size 200. Similarly, the city-names for lexicon size 20 were chosen by considering 10th, 20th, 30th, 40th .... city-names from the full city-name lexicon of size 200. From our experiment we got 98.27%, 97.53%, 95.41%, 93.75%, 93.09% and 92.38% accuracy when we considered lexicon of sizes 10, 20, 50, 100, 150 and 200 city-names, respectively. Also we have computed accuracy of five top choices on different lexicon size and their details results are given in Table 3.

Table 2 Error and reliability results of the proposed system with respect to different rejection rates.

Values of T_1 and T_2		Reliability	Error rate	Rejection rate
T_1	T_2			
10.0	3.6	98.01%	1.82%	8.43%
25.0	5.6	99.00%	0.87%	13.12%
30.0	7.8	99.50%	0.41%	19.14%
30.0	11.8	99.90%	0.07%	31.66%
55.0	18.6	100.00%	0.00%	65.00%

Table 3 Accuracy on different lexicon size.

Accuracy on diff. top choices	Lexicon size					
	10	20	50	100	150	200
1	98.27%	97.53%	95.41%	93.75%	93.09%	92.38%
2	99.22%	98.74%	97.65%	96.46%	95.93%	95.46%
3	99.50%	99.20%	98.28%	97.43%	96.92%	96.57%
4	99.57%	99.39%	98.68%	97.98%	97.52%	97.14%
5	99.69%	99.42%	98.88%	98.25%	97.89%	97.57%

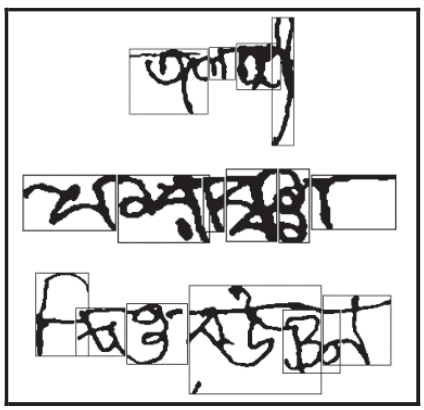
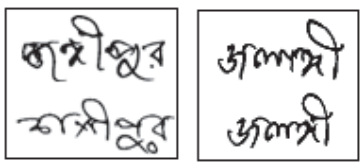


Fig. 13 Examples of some misrecognized city-names due to under segmentation.



(a) (b)

Fig. 14 Examples of some misrecognized city-names due to similar shapes structures.

7.4 Error Analysis

From the experiment we noted that some errors occurred when the set of character pre-segmentation points obtained from our pre-segmentation module did not contain actual character segmentation points. Some examples of such erroneous city-names with their segmentation are shown in Fig. 13. Another reason of mis-recognition was the shape similarity of some of the city-names. Examples of some similar-shaped city-names are shown in Fig. 14. In Fig. 14 (a), there are two different city-names and the first three characters of these two city-names are different. Although first three characters of these two city-names are different, these two city-names show similar shape behavior because of the writing style of the individuals. Because of such shape similarity some errors occurred in our proposed scheme. Similarly, in Fig. 14 (b) two city-name images are shown but they are very similar in shape although their grapheme structure is different.

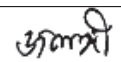
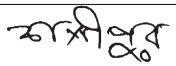
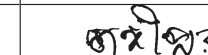
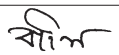
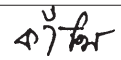
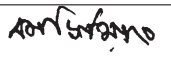
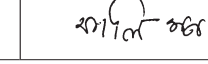
From the experiment we noticed that one of the main reasons for not getting better recognition rates was the complex structure of some of the city-names due to presence of compound characters in Bangla script. In English there is no compound character and number of characters is only 52 (26 upper-case and 26 lower-case). In Bangla there are more than 280 character shapes. Thus, recognition task of Bangla handwritten city-name is more difficult than that of English.

From the experiment we also noticed that much higher result was obtained when city-names written only by basic characters were considered. We obtained 96.44% accuracy (with 0.00% rejection) by our proposed scheme when it was tested on the city-names written in basic characters only (note that we considered 84 city-names and out of these 84 city-names 39 city-names were written only by basic characters). Also we noticed that 98.16%, 98.81%, 99.11%, 99.35% accuracy obtained from the city-names written in basic characters when we considered top 2, 3, 4

Table 4 Error and reliability results of the proposed system with respect to different rejection rates on the city-names written in basic characters only.

Reliability	Error rate	Rejection rate
98.04%	1.90%	3.38%
99.04%	0.89%	7.17%
99.54%	0.41%	9.78%
99.92%	0.06%	21.22%
100.00%	0.00%	24.66%

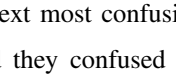
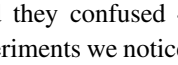
Table 5 Main confusion pair of Bangla city-names.

Confusing city-name pairs		**% of confusion**	
		When computed from the samples of two classes	When computed from samples of all the city-name classes (overall-class)
		11.20	0.26
		4.90	0.11
		4.11	0.10
		3.92	0.09

and 5 choices, respectively.

From the experiment of the city-names written in basic characters we also computed city-name recognition results with different rejection rates. We noted that 98.04% reliability was obtained when 3.38% rejection was considered in the proposed system. We obtained 100% reliability when rejection rate was 24.66%. Details results are shown in Table 4.

7.5 Confusing City-Name Pair Computation

We also noticed the main confusing pairs of Bangla city-names and their error rates are shown in Table 5. In Table 5 we provide confusion rate when it is calculated from the samples of only two confusing city-name classes (two-class) as well as when it is calculated from the samples of all the city-name classes (overall-class). The city-names  and  confused maximum between them and their confusion rates were 11.20% (two-class) and 0.26% (overall-class). The next most confusing pair was  and  , and they confused 4.90% cases between them. From the experiments we noticed that mainly similar shaped city-names confused by the system at higher rate.

7.6 Results on Poor/Noisy Image

Since we use statistical classifier and our feature detection technique is not very sensitive to noise, our scheme shows correct results even if the samples are poor in quality. To get an idea of such samples, some images where we got correct results from our system are shown in Fig. 15.

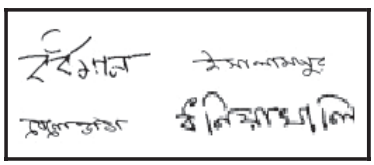


Fig. 15 Examples of some poor samples of the city-names.

7.7 Recognition Accuracy on Different Word Length

To get the idea about the recognition rates on different values of upper bound (U_b) and lower bound (L_b) of the estimated length of an input city-name, we computed recognition accuracy for different values of N and the results are shown in Table 6. Here N is the estimated length of the input city-name and this length is nothing but the number of zone (length estimation is discussed in Sect. 6.3). To have the idea about recognition time of the system for different values of upper bound (U_b) and lower bound (L_b), we also provided here the computation time required in our system. Different computation times are shown within the bracket of Table 6. We used AMD Athlon MP2800+ with 2.133 GHz computer for our experiment.

Table 6 Recognition results obtained for different values of U_b and L_b. Computation time per city-name (in mili-seconds) required for the respective U_b and L_b values is mentioned in the bracket.

	Recognition results and the respective computation time (in mili-second)			
$U_b \rightarrow$ / $L_b \downarrow$	**N-1**	**N**	**N+1**	**N+2**
N/2	89.50% (22.26)	91.55% (22.44)	91.79% (22.64)	91.81% (22.92)
N/2 -1	91.41% (22.81)	93.46% (23.27)	93.70% (23.35)	93.73% (23.55)
N/2-2	91.76% (23.21)	93.81% (23.43)	94.05% (23.66)	94.08% (24.12)
N/2-3	91.78% (23.38)	93.83% (23.65)	94.08% (23.62)	94.10% (23.78)

7.8 Comparison of Results

Although there are few research works on isolated Bangla character recognition, to the best of our knowledge there are not many studies on Bangla city-name recognition. Mashiyat et al. [27] proposed a conventional method where Bangla words are, at first, segmented into characters and then segmented characters are recognized. They reported 90% character recognition accuracy from their proposed method. Previously we published an NSHP-HMM (Non-Symmetric Half Plane-Hidden Markov Model) based method [19] where we obtained 86.44% accuracy on 76 city-names. From this current method we obtained 94.08% accuracy from 84 city-names and this results is 7.64% better than the results of the NSHP-HMM method. Note that all 76-city-names used for NSHP-HMM method are included with these 84 city-names. Recently, Bhowmik et al. [28] proposed an HMM based method for Bangla city-name recognition and they obtained 79.12% accuracy from 119 city-names. To get the idea about city-name recognition results on English city-name (on USPS data) Kimura et al. [23] obtained 95.46% accuracy when lexicon size is 100 where as for Bangla city-name we obtained 93.75% accuracy when lexicon size is 100 (see Table 3). Recently, in Arabic city-name recognition competition, 87.22% highest accuracy is reported and this result is obtained from Arabic city-name data of lexicon size 937 [5]. Thus, the results we obtained for Bangla city-name recognition is comparable with others.

To show the goodness of the proposed water reservoir based pre-segmentation method, we perform some experiments. In [2] a contour based pre-segmentation method is discussed. Here pre-segmentation points are detected at the valley points of the upper contour of a city-name. Applying the same pre-segmentation method proposed in [2] in our system we obtained 82.56% recognition accuracy on our Bangla city-name data. Considering pre-segmentation points as the valley points of the lower contour of a city-name in our system we obtained 91.07% recognition accuracy on the same Bangla city-name data. Whereas using our water reservoir based pre-segmentation algorithm we obtained 94.08% accuracy on the same Bangla dataset.

8. Conclusion

In this paper a lexicon-driven segmentation-recognition scheme on Bangla handwritten city-name recognition is proposed for Indian postal automation. India is a multi-lingual and multi-script country with more than 19 official languages, and 11 scripts are used for these languages. Although there are some research works on isolated handwritten character recognition, not much work is done towards Indian city-name recognition. From the experiment of our system we obtained encouraging results on the city-name of Bangla, which has many complex shaped characters compared to English. We hope this work will be helpful to the researcher for the future research of other Indian languages. As India is a multi-lingual and multi-script country, a city-name may be written in two or more languages. So it is necessary to develop a multi-script city-name recognition scheme for such country and in future we plan to extend this work for this purpose.

References

[1] A.L. Koerich, Y. Leydier, R. Sabourin, and C.Y. Suen, "A hybrid large vocabulary handwritten word recognition system using neural networks with hidden Markov models," Proc. 8th International Workshop on Frontiers in Handwriting Recognition, pp.99–104, 2002.

[2] F. Kimura, S. Tsuruoka, Y. Miyake, and M. Shridhar, "A lexicon directed algorithm for recognition of unconstrained handwritten words," IEICE Trans. Inf. & Syst., vol.E77-D, no.7, pp.785–793, July 1994.

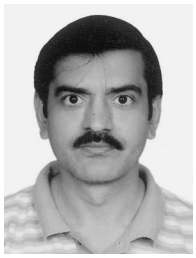
[3] C.L. Liu, M. Koga, and H, Fujisawa, "Lexicon driven segmentation and recognition of handwritten character strings for Japanese address reading," IEEE Trans. Pattern Anal. Mach. Intell., vol.24, no.11, pp.1425–1437, 2002.

[4] A. Amin, "Off-line Arabic character recognition: The state of the art," Pattern Recognit., vol.31, pp.517–530, 1998.

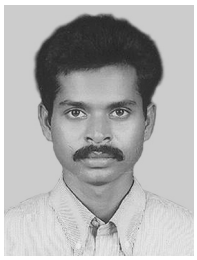
[5] V. Märgner and H.E. Abed, "Arabic handwriting recognition competition," Proc. 9th International Conf. on Document Analysis and Recognition, pp.1274–1278, 2007.

[6] R. Plamondon and S.N. Srihari, "On-line and off-line handwritten recognition: A comprehensive survey," IEEE Trans. Pattern Anal. Mach. Intell., vol.22, no.1, pp.62–84, 2000.

[7] S. Madhvanath and V. Govindaraju, "The role of holistic paradigms in handwritten word recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol.23, no.2, pp.149–164, 2001.
[8] J. Hull, T. Ho, J.T. Favata, V. Govindaraju, and S. Srihari, "Combination of segmentation-based and holistic handwritten word recognition algorithms," Proc. 2nd International Workshop on Frontiers in Handwriting Recognition, pp.229–240, 1992.
[9] M.Y. Chen, A. Kundu, and J. Zhou, "Off-line handwritten word recognition using a hidden Markov model type stochastic network," IEEE Trans. Pattern Anal. Mach. Intell., vol.16, no.5, pp.481–496, 1994.
[10] F. Kimura, K. Takashina, S. Tsuruoka, and Y. Miyake, "Modified quadratic discriminant functions and the application to Chinese character recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol.9, no.1, pp.149–153, 1987.
[11] H. Liu and X. Ding, "Handwritten character recognition using gradient features and quadratic classifier and multiple discrimination scheme," Proc. 8th International Conf. on Document Analysis and Recognition, pp.19–23, 2005.
[12] J.T. Favata, "Offline general handwritten word recognition using an approximate BEAM matching algorithm," IEEE Trans. Pattern Anal. Mach. Intell., vol.23, no.9, pp.1009–1021, 2001.
[13] U. Mahadevan and S.N. Srihari, "Parsing and recognition of city, state, and ZIP codes in handwritten addresses," Proc. 5th International Conf. on Document Analysis and Recognition, pp.325–328, 1999.
[14] X. Wang and T. Tsutsumida, "A new method of character line extraction from mixed-unformatted document image for Japanese mail address recognition," Proc. 5th International Conf. on Document Analysis and Recognition, pp.769–772, 1999.
[15] G. Kim and V. Govindaraju, "Handwritten phrase recognition as applied to street name images," Pattern Recognit., vol.31, pp.41–51, 1998.
[16] S.N. Srihari and E.J. Keubert, "Integration of hand-written address interpretation technology into the United States postal service remote computer reader system," Proc. 4th International Conf. on Document Analysis and Recognition, pp.892–896, 1997.
[17] Y. Wen, Y. Lu, and P. Shi, "Handwritten Bangla numeral recognition system and its application to postal automation," Pattern Recognit., vol.40, pp.99–107, 2007.
[18] U. Pal and B.B. Chaudhuri, "Indian script character recognition: A survey," Pattern Recognit., vol.37, pp.1887–1899, 2004.
[19] K. Roy, S. Vajda, U. Pal, B.B. Chaudhuri, and A. Belaid, "A system for Indian postal automation," Proc. 8th International Conf. on Document Analysis and Recognition, pp.1060–1064, 2005.
[20] U. Pal, K. Roy, and F. Kimura, "A lexicon driven method for unconstrained Bangla handwritten word recognition," Proc. 10th International Workshop on Frontiers in Handwriting Recognition, pp.601–606, 2006.
[21] U. Pal, A. Belaid, and C. Choisy, "Touching numeral segmentation using water reservoir concept," Pattern Recognition Lettters, vol.24, pp.261–272, 2003.
[22] B.B. Chaudhuri and U. Pal, "Relational studies between phoneme and grapheme statistics in current Bangla," Journal of Acoustical Society of India, vol.23, pp.67–77, 1995.
[23] F. Kimura, M. Shridhar, and Z. Chen, "Improvement of a lexicon directed algorithm for recognition of unconstrained handwritten words," Proc. 2nd International Conf. on Document Analysis and Recognition, pp.18–22, 1993.
[24] R. Bertolami, S. Uchida, M. Zimmermann, and H. Bunke, "Non-uniform slant correction for handwritten text line recognition," Proc. 9th International Conf. on Document Analysis and Recognition, pp.18–22, 2007.
[25] E. Taira, S. Uchida, and H. Sakoe, "Non-uniform slant correction for handwritten word recognition," IEICE Trans. Inf. & Syst., vol.E87-D, no.5, pp.1247–1253, May 2004.
[26] N. Otsu, "A threshold selection method from gray-level histograms," IEEE Trans. Syst. Man Cybern., vol.9, no.1, pp.62–66, 1979.
[27] A.S. Mashiyat, A.S. Mehadi, and K.H. Talukder, "Bangla off-line handwritten character recognition using superimposed matrices," Proc. 7th International Conf. on Computer and Information Technology, pp.610–614, 2004.
[28] T.K. Bhowmik, S.K. Parui, and U. Roy, "Discriminative HMM training with GA for handwritten word recognition," Proc. International Conf. on Pattern Recognition, 2008.
[29] K. Sawa, T. Wakabayashi, S. Tsuruoka, F. Kimura, and Y. Miyake, "Accuracy improvement by gradient feature and variance absorbing covariance matrix in handwritten Chinese character recognition," IEICE Trans. Inf. & Syst. (Japanese Edition), vol.J84-D-II, no.11, pp.2387–2397, Nov. 2001.



Umapada Pal received his Ph.D. from Indian Statistical Institute and his Ph.D. work was on the development of Printed Bangla OCR system. He did his Post Doctoral research on the segmentation of touching English numerals at INRIA (Institut National de Recherche en Informatique et en Automatique), France. During July 1997–January 1998 he visited GSF-Forschungszentrum fur Umwelt und Gesundheit GmbH, Germany to work as a guest scientist in a project on image analysis. From January 1997, he is a Faculty member of the Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata. His primary research is Digital Document Processing. He has published 115 research papers in various international journals, conference proceedings and edited volumes. In 1995, he received student best paper award from Chennai chapter of Computer Society of India. He received a merit certificate from Indian Science Congress Association in 1996. Because of his significant impact in the Document Analysis research domain of Indian language, TC-10 and TC-11 committees of IAPR (International Association for Pattern Recognition) presented 'ICDAR Outstanding Young Researcher Award' to Dr. Pal in 2003. In 2005–2006 Dr. Pal has received JSPS fellowship from Japan government. Dr. Pal has been serving as a program committee member of many conferences including International Conference on Document Analysis and Recognition (ICDAR), International Workshop on Document Image Analysis for Libraries (DIAL), International Workshop on Frontiers of Handwritten Recognition (IWFHR), International Conference on Pattern recognition (ICPR) etc. Also, he is the Asian PC-Chair for 10th ICDAR to be held at Barcelona, Spain in 2009. He has served as the guest editor of special issue of VIVEK journal on Document image analysis of Indian scripts. Also currently he is co-editing a Special issue of the journal of Electronic Letters on Computer Vision and Image Analysis. He is a life member of IUPRAI (Indian unit of IAPR) and senior life member of Computer Society of India.



Kaushik Roy obtained his Bachelor degree from NIT Silchar, Masters and Ph.D. from Jadavpur University in Computer Science and Engineering in 1998, 2002 and 2008, respectively. He is currently working as a lecturer in the Dept. of Computer Science and Engineering, West Bengal University of Technology. He has worked as Scientific Officer in Centre for development of Advanced Computing (CDAC) Kolkata and as Asst. Professor in Narula Institute of Technology, Kolkata. Earlier he had worked as a Project Linked Personnel in Computer Vision and Pattern Recognition Unit of Indian Statistical Institute Kolkata on the development of a complete system for Postal Automation. His areas of interests are in Image Processing, Pattern Recognition, Handwriting Recognition, Document Processing etc. He has published over thirty papers in International Journals and Conference Proceedings. He was a champion in the National Competition for Young IT Professional Awards 2004 (YITPA-04), conferred by the Computer Society of India. He is a member of the Indian Unit for Pattern Recognition and Artificial Intelligence (affiliated to the International Association of Pattern Recognition).

Fumitaka Kimura received his Master and Ph.D. degrees from Nagoya University in 1975 and 1981, respectively. From 1978 to 1982 he was a faculty member in the Electrical Engineering Department of Nagoya University. Since 1982, he has been with Mie University, where he is at present a professor at the department of Information Engineering. During 1989–1991, he was a visiting associate professor at the University of Michigan-Dearborn U.S.A. His research interests are in the areas of pattern recognition and its application to character recognition, image processing and document image recognition. He obtained the best paper award of DAS'96 (International Association for Pattern Recognition Workshop on Document Analysis Systems), and the best awards of the Character Recognition Technical Contests (1992, 1993, 1994) which were held as competition on the accuracy of machine recognition of handwritten numerals by the IPTP. Prof. Kimura has been serving as a program committee member of many conferences/workshops including International Conference on Document Analysis and Recognition (ICDAR), International Workshop on Frontiers of Handwritten Recognition (IWFHR) etc. He is the Asian Program Committee Chair for International Conference on Frontiers of Handwritten Recognition (ICFHR) to be at Canada in 2008. He is a member of IPS Japan, JSMBE.