7. Development of OHWR System for Bangla

S. K. Parui (PI), Dr. U. Bhattacharya & Team
ISI, Kolkata

Summary

Our major concern is the development of a generic recognizer for unconstrained Bangla handwriting. Unconstrained handwriting style in Bangla is mixed cursive in nature. Although this is the most natural style of handwriting, it is the most difficult in respect of its automatic recognition. The degree of the recognition difficulty is further increased for Bangla due to its large alphabet size. Moreover, the sample data (more than 1,00,000 samples) collected during Phase I, were written without any kind of supervision and thus identifying a smaller subset of class A data (word samples in which each segmented character can be identified manually without the context of the word) from this large database was found to be impossible.

In spite of the above unfavourable situation, we are working hard, towards the development of a generic recognizer for this handwriting database. The latest status of our progress in this endeavour is summarized below.

1. A semi-automatic annotation tool is developed.
2. Till date we annotated 37,824 cursive word samples at Unicode level using the above tool.
3. The above annotation exercise produced 2,24,586 sub-stroke samples.
4. We identified 7833 training word samples from our sample database and segmented them using our automatic segmentation algorithm. This produced a training database of 52,139 sub-strokes.
5. Using the above annotation tool we identified these 52,139 sub-strokes into their individual character classes.
6. We manually identified 114 different sub-stroke classes corresponding to the above training set of sub-strokes. These sub-strokes belong to our present symbol set of 110 aksharas.
7. We decided on a novel feature set for these 114 sub-stroke classes.
8. We implemented a novel classification scheme for the sub-strokes.
9. The symbol and word recognition modules are now under development.
10. We developed a data collection tool for Andriod-based tabs.
11. We developed a novel lightweight handwriting recognition system for handheld devices.
12. We published 5 papers in Proceedings of reputed Conferences on our OHWR activities.
**Introduction**

Unlike other Indian scripts unconstrained handwriting in Bangla is cursive in nature similar to English. However, connectedness between adjacent characters in a cursive written Bangla word occurs in its upper portion (as shown in Fig. 1) in contrast to English where it usually occurs in the lower portion. Another dissimilarity with English is that Bangla like several other Indic scripts, consists of a very large character set many of which are extremely complex in shape.

![Figure 1: A sample of unconstrained Bangla handwritten word consisting of 9 characters numbered i, ii, ..., ix and written using 9 strokes numbered 1, 2, ..., 9; stroke number 9 represents a headline (“matra”); dotted vertical line segments show the character boundaries in connected situations. One color represents one stroke.](image)

To the best of our knowledge, there does not exist enough studies of online Bangla handwriting. A few studies of online unconstrained Bangla handwriting recognition include [1], [2] and [3]. In [1], a segmentation based analytic scheme for recognition of unconstrained Bangla handwritten words were studied, in [2], a study of writer independent online Bangla word recognition task based on hidden Markov models (HMM) was presented and in [3], a combination of multi-layer perceptron (MLP) and support vector machine (SVM) classifiers was studied for limited vocabulary Bangla cursive handwriting recognition. Additionally, in the literature, there exists a few reports on studies of online isolated Bangla character/numeral recognition [4], [5], [6], [7], [8] and [9]. In the earliest work on online Bangla handwriting recognition [4], Bangla isolated basic characters were considered. An HMM based scheme was studied in [5] for recognition of online handwritten isolated Bangla numerals. Bhattacharya et al. [6] studied a direction code based feature vector for recognition of online handwritten Bangla basic characters. The first benchmark recognition results on online handwritten isolated characters of a few Indian scripts including Bangla were reported in [9].

**Data collection and annotation**

In Phase II, we collected paragraph level data using a Newspaper report on last general election consisting of 676 words and covering around 95% of the alphabetic characters appearing in a news corpus of Bangla. This paragraph level handwritten data has been collected from 31 writers. This new database consists of 73 paragraphs, 3122 sentences, 33774 words and 1,71,774 characters at Unicode level.

Unconstrained handwritten word database collected in Phase I is now being annotated at character level. Till date we annotated 37824 word samples consisting of 1,25,629 characters at Unicode level. The whole of new paragraph level data had been annotated at word level and stored as XML files.

**Semi-automatic annotation scheme for Bangla online mixed cursive handwriting samples**

In Phase II our group at the Indian Statistical Institute, Kolkata has developed a GUI-based semi-automatic scheme [10] for annotation of cursive handwritten Bangla
Unlike other Indian scripts unconstrained handwriting in Bangla is cursive in nature similar to English. However, connectedness between adjacent characters in a cursive written Bangla word occurs in its upper portion (as shown in Fig. 1) in contrast to English where it usually occurs in the lower portion. Another dissimilarity with English is that Bangla like several other Indic scripts, consists of a very large character set many of which are extremely complex in shape.

To the best of our knowledge, there does not exist enough studies of online Bangla handwriting. A few studies of online unconstrained Bangla handwriting recognition include [1], [2] and [3]. In [1], a segmentation based analytic scheme for recognition of unconstrained Bangla handwritten words were studied, in [2], a study of writer independent online Bangla word recognition task based on hidden Markov models (HMM) was presented and in [3], a combination of multiplayer perceptron (MLP) and support vector machine (SVM) classifiers was studied for limited vocabulary Bangla cursive handwriting recognition. Additionally, in the literature, there exists a few reports on studies of online isolated Bangla character/numeral recognition [4], [5], [6], [7], [8] and [9]. In the earliest work on online Bangla handwriting recognition [4], Bangla isolated basic characters were considered. An HMM based scheme was studied in [5] for recognition of online handwritten isolated Bangla numerals. Bhattacharya et al. [6] studied a direction code based feature vector for recognition of online handwritten Bangla basic characters. The first benchmark recognition results on online handwritten isolated characters of a few Indian scripts including Bangla were reported in [9].

In Phase II, we collected paragraph level data using a Newspaper report on last general election consisting of 676 words and covering around 95% of the alphabetic characters appearing in a news corpus of Bangla. This paragraph level handwritten data has been collected from 31 writers. This new database consists of 73 paragraphs, 3122 sentences, 33774 words and 1,71,774 characters at Unicode level.

Unconstrained handwritten word database collected in Phase I is now being annotated at character level. Till date we annotated 37824 word samples consisting of 1,25,629 characters at Unicode level. The whole of new paragraph level data had been annotated at word level and stored as XML files.

In Phase II our group at the Indian Statistical Institute, Kolkata has developed a GUI-based semi-automatic scheme [10] for annotation of cursive handwritten Bangla words at character boundary levels and a scheme for XML representation of such annotated word samples. The present system implemented for annotation of unconstrained handwriting of Bangla may easily be customized for other scripts. Currently this system is in use for character level annotation of approximately 1,00,000 samples of Bangla online mixed cursive handwritten words. A block diagram of our annotation scheme is shown in Fig. 2.

![Figure 2. Block diagram of the annotation tool](image)
Our annotation tool has two separate units, one of which takes care of segmentation of a document page into words and the other is used for annotation of individual words at the sub-stroke level.

1. **Line and word segmentation unit.** It consists of a GUI-based unit for segmenting an online handwritten document page into lines and each line into words. Line segmentation module compares two consecutive (temporal order) strokes w.r.t. the distributions of both x- and y-values of the respective sets of sample points to decide if the next stroke belongs to a new line. Off-line information such as horizontal projection profile is considered to settle possible confusions. Although this line segmentation approach is less prone to errors, the GUI-based module provides options for manual correction of any error occurring during this stage.

2. Automatic segmentation of words is usually more difficult than line segmentation. Often the rectangular box enclosing a word overlaps with a similar box enclosing an adjacent word. We have dealt with this problem efficiently by considering each pair of consecutive strokes in a line and obtaining the histogram of the absolute difference (Dstroke) in x-values of the rightmost sample point of the first stroke and the leftmost sample point of the next stroke. An output of our segmentation approach is shown in Fig. 3. This output is manually checked and then a “save” button is clicked. Each

![Figure 3. Screenshot of the segmentation output – horizontal and vertical line segments are used to show line and word segmentations respectively](image-url)
Our annotation tool has two separate units—one of which takes care of segmentation of a document page into words and the other is used for annotation of individual words at the sub-stroke level.

1. **Line and word segmentation unit.**
   It consists of a GUI-based unit for segmenting an online handwritten document page into lines and each line into words. Line segmentation compares two consecutive (temporal order) strokes w.r.t. the distributions of both x- and y-values of the respective sets of sample points to decide if the next stroke belongs to a new line. Off-line information such as horizontal projection profile is considered to settle possible confusions. Although this line segmentation approach is less prone to errors, the GUI-based module provides options for manual correction of any error occurring during this stage.

2. **Automatic segmentation of words is usually more difficult than line segmentation.** Often the rectangular box enclosing a word overlaps with a similar box enclosing an adjacent word. We have dealt with this problem efficiently by considering each pair of consecutive strokes in a line and obtaining the histogram of the absolute difference (Dstroke) in x-values of the rightmost sample point of the first stroke and the leftmost sample point of the next stroke. An output of our segmentation approach is shown in Fig. 3. This output is manually checked and then a “save” button is clicked. Each segmented word is stored as a separate ASCII file named such as “Rita Chatterjee_2_8_Word_665_siddhAnta.txt”. This file stores the ink of the word “siddhAnta” (phonetic transliteration) written by the writer named “Rita Chatterjee” in the 8th position of the 2nd line of the document page and the serial number of the word in our corpus is 665. This file stores the coordinates of the sample points in addition to the Unicode of the word.

2. **Character level annotation unit.**
   This is the GUI-based second unit of our semi-automatic annotation tool. Since a character in a word may consist of non-integral number of strokes and no off-the-shelf recognition engine is available, strokes across multiple characters are manually segmented into sub-strokes by clicking the mouse. The tool displays a cardinal number against each stroke representing their temporal order. The annotator uses the mouse to mark the desired segmentation points (character boundary) on the display of the word. It is not necessary to place the mouse pointer exactly on the trajectory—the toolkit searches for a point on the trajectory nearest to the mouse click and introduces a soft PEN_UP creating two sub-strokes. The “Refresh” button (provided in the second row of the top panel) can be used to undo placement of any unintentional mark. Once all such segmentation points are marked, the “Accept” button is pressed to finalize segmentation of the word into sub-strokes. These are shown in Fig. 4.

![Figure 4. A screenshot of the character level annotation tool.](image-url)
Once the tool accepts manually introduced segmentation points, it places new cardinal numbers alongside individual sub-strokes (if a stroke is not segmented by placing a soft PEN_UP over it, the stroke is composed of a single sub-stroke). Now, the annotator manually places these cardinal numbers in the respective edit boxes provided in the second row of the top panel. Once all the entries are incorporated, the “Save” button provided in the first row of the top panel is pressed to store the ink data along with annotation information in an ASCII file. This final stage is shown in Fig. 5.

3. XML representation unit. Existing studies of representation of annotated handwriting samples of Indic scripts did not consider the peculiarity of unconstrained Bangla handwriting in which a character may be formed by a non-integral number of strokes. We used a novel strategy of XML representation of annotated Bangla handwriting samples. The peculiar nature of character-stroke relationship existing in unconstrained Bangla handwriting is tackled by introducing tags called “stroke_substroke_relation” and “char_composition”. A part of our XML file is shown in Fig. 6 and our XML schema is shown in Fig. 7.
Once the tool accepts manually introduced segmentation points, it places new cardinal numbers alongside individual sub-strokes (if a stroke is not segmented by placing a soft PEN_UP over it, the stroke is composed of a single sub-stroke). Now, the annotator manually places these cardinal numbers in the respective edit boxes provided in the second row of the top panel. Once all the entries are incorporated, the “Save” button provided in the first row of the top panel is pressed to store the ink data along with annotation information in an ASCII file. This final stage is shown in Fig. 5.

3. XML representation unit.

Existing studies of representation of annotated handwriting samples of Indic scripts did not consider the peculiarity of unconstrained Bangla handwriting in which a character may be formed by a non-integral number of strokes. We used a novel strategy of XML representation of annotated Bangla handwriting samples. The peculiar nature of character-stroke relationship existing in unconstrained Bangla handwriting is tackled by introducing tags called “stroke_substroke_relation” and “char_composition”. A part of our XML file is shown in Fig. 6 and our XML schema is shown in Fig. 7.

Figure 6. A part of XML file

Figure 7. Chart showing the XML schema
Identification of stroke classes from handwritten isolated Bangla basic characters

An online handwritten character sample is composed of one or more strokes. We have identified seventy-five stroke classes on the basis of the varying handwriting styles present in our database of Bangla basic characters. Each character sample is a sequence of strokes emanating from one or more of these stroke classes. Additionally, we developed a new database of handwritten Bangla strokes from our existing database of Bangla basic characters [11]. This is the first such database at least for Bangla script, if not for any of the Indic scripts.

Our database of Bangla online handwritten basic characters contains 38,567 samples and it is divided into a training set of 29,951 samples and a test set of 8,616 samples. For the entire database, we have identified strokes of 75 different shapes (shown in Fig. 8). We have developed a database of samples of these 75 stroke classes from the above database of handwritten samples of 50 character classes in the following way. For each handwritten character sample, the strokes (pen down to pen up) are first identified and each of these strokes is then assigned to the corresponding stroke bin.

We presented an entry-level lightweight personalized Bangla handwriting recognition system suitable for touchscreen based Android devices in [12]. Recently, we presented the second and an improved version of the initial system in [13]. This system is well suited for minimal user-lag on devices having only limited computing power.

Bangla handwriting recognizer for Android-based handheld devices in sharp contrast to standard laptops or desktop computers. Moreover, the approach is user-adaptive in the sense that it can adapt through user corrections to wrong predictions. With an increasing number of interactive corrections by the user, the recognition accuracy improves significantly. An input stroke is first re-sampled generating a fixed small number of sample points such that at most two critical points are preserved. The feature vector consists of only the x- and y-coordinates of the above points. The mean feature vector and the inverted feature covariance matrix for each stroke class are stored as Serialized Objects on the SD card of the device. These are used for recognition of input strokes based on Mahalanobis distance.

A Look-Up Table (LUT) of stroke combinations as keys and corresponding character class as values is used for the final Unicode character output. In case of an incorrect character output, user corrections are used to automatically update the LUT adapting to the user’s particular handwriting style.

Figure 8. Ideal shapes of Bangla basic characters and the corresponding stroke classes identified from their online handwritten samples.
Our database of Bangla online handwritten basic characters contains 38,567 samples and it is divided into a training set of 29,951 samples and a test set of 8,616 samples. For the entire database, we have identified strokes of 75 different shapes (shown in Fig. 8). We have developed a database of samples of these 75 stroke classes from the above database of handwritten samples of 50 character classes in the following way. For each handwritten character sample, the strokes (pen down to pen up) are first identified and each of these strokes is then assigned to the corresponding stroke bin.

**Bangla handwriting recognizer for Android-based handheld devices**

We presented an entry-level lightweight personalized Bangla handwriting recognition system suitable for touchscreen based Android devices in [12]. Recently, we presented the second and an improved version of the initial system in [13]. This system is well suited for minimal user-lag on devices having only limited computing power in sharp contrast to standard laptops or desktop computers. Moreover, the approach is user-adaptive in the sense that it can adapt through user corrections to wrong predictions. With an increasing number of interactive corrections by the user, the recognition accuracy improves significantly. An input stroke is first re-sampled generating a fixed small number of sample points such that at most two critical points are preserved. The feature vector consists of only the x- and y-coordinates of the above points. The mean feature vector and the inverted feature covariance matrix for each stroke class are stored as Serialized Objects on the SD card of the device. These are used for recognition of input strokes based on Mahalanobis distance. A Look-Up Table (LUT) of stroke combinations as keys and corresponding character class as values is used for the final Unicode character output. In case of an incorrect character output, user corrections are used to automatically update the LUT adapting to the user’s particular handwriting style.

Figure 8. Ideal shapes of Bangla basic characters and the corresponding stroke classes identified from their online handwritten samples.

Figure 9. Block diagram of our lightweight personalized Bangla handwriting recognizer (a) stroke classification module, (b) user adaptive character recognition module
This system consists of two major modules – one for stroke classification and the other for character recognition. The latter module includes a component for user adaptation. Individual strokes of a character are first recognized in an independent way. A look-up-table is used to determine the character based on the results of stroke classification. If the character output is wrong, the user may choose the correct output from a list of a few top choices. The block diagrams of these two modules are shown in Fig. 9(a) and Fig. 9(b) respectively.

Figure 9(c). The system recognizes handwriting character-by-character and the results are placed in the editable Text Area at the bottom of the Tab’s screen using Unicode values. Top 10 matches of the last written character along with respective match scores are shown at the right sidebar.
Our system has a GUI based front-end consisting of the following:
(1) A View where the character is to be written,
(2) A “Canvas” area below the transparent View area,
(3) A Text Area, and
(4) Prediction Scores and choices

The character to be written is drawn in the Overlay View as shown in Fig. 9(c) and is reflected persistently in the Canvas area below. This ensures that a word is visible in its entirety instead of only the letter that is currently being drawn onscreen.

**Feature computation**

Features are formed directly from the coordinates of a stroke, i.e., \((x_1, y_1, x_2, y_2, ..., x_n, y_n)\). Since we resampled each stroke into 16 points as described above, the value of \(n\) in our implementation is 16. Thus, here we consider a 32-dimensional feature vector.

**Stroke classification**

The well-known Mahalanobis distance is used in this case as a trade-off between sophistication of classification and the limitations of the computational power of the device. The test device was a Samsung Galaxy Tab with a dual core 1 GHz processor. Using DTW matching on Euclidean distance between points on the stroke required 9-11 secs. The present approach takes less than 1 sec. Naturally, the less time taken per stroke recognition the more is the usability of the handwriting recognition system.

Formally, the Mahalanobis distance of a multivariate vector \(x = (x_1, x_2, x_3, ..., x_n)^T\) from a group of values with mean \(\mu = (\mu_1, \mu_2, \mu_3, ..., \mu_n)^T\) and covariance matrix defined as:

\[D_M(x) = \sqrt{(x - \mu)^T S^{-1} (x - \mu)}.\]

In order to use the Mahalanobis distance to classify an \(N\)-dimensional input stroke as belonging to one of \(k\) stroke classes \(\{s_1, s_2, ..., s_k\}\), we first estimate the \(N \times N\) covariance matrix \(S\) and the \(N \times 1\) mean vector \(\mu\) of each class, based on the collected UNIPEN samples known to belong to each class. Then, given a test sample from user’s gesture on touchscreen, we compute the Mahalanobis distance to each class, and classify the test point as belonging to that class for which the Mahalanobis distance is the minimum. Formally,

\[s_{\text{input}} = \{s_i | \min D_M(s_{\text{input}}, s_i) \text{ for } i = 1, 2, ..., k\}.\]

In the implementation for Android devices, the matrix \(S^{-1}\) and the mean vector for each class are pre-calculated on a workstation and stored as Java serialized objects on the device’s SD card. Thus, only 3 matrix multiplications per stroke class are needed to be performed on the device to classify each stroke, in addition to the time needed for fast access of serialized objects from secondary memory. The squared Mahalanobis distance measure is used to avoid the overhead of a floating point square-root operation.

**User-adaptive Character recognition**

A look-up table (LUT) is formed with \(<\text{key}, \text{value}>\) pairs. An empirically determined time quantum of user inactivity, is taken to signify the end of writing a character.

The initial LUT is created beforehand on the basis of some commonly found \(<\text{key}, \text{value}>\) pairs. This is later adapted to
a user’s particular usage of strokes by dynamically updating the LUT. If the user corrects the output character by replacing it with a new character, then that stroke sequence <key,value> pair in the LUT is modified with the new character class ID. To handle size explosion of LUT over user corrections, a frequency count of the usage of each <key,value> pair is maintained. The least used (key, value) pairs are deleted after the LUT size crosses a threshold. This threshold may be fixed empirically depending upon the configuration of the device.

Here, it is assumed that generally a single person will be using a particular device. Thus, the corrected LUT <key,value> pairs would be valid for that user’s writing style. Moreover, it has been observed that writers usually have unique ways of the order of writing the constituent strokes of a character. Hence, the order of strokes is not meaningful (and the stroke IDs are sorted on numerical value before computing the hash), as two different users may write the same character in different stroke sequences.

Identification of sub-stroke classes forming basic characters, modifiers and compound characters from unconstrained handwritten Bangla words

In the present study, the Bangla symbol set consists of all basic characters, all modifiers and several compound characters as shown in Fig. 10. When the basic input unit is a handwritten word sample, it needs to be segmented into smaller units in order to finally recognize the word sample. We use the algorithm proposed in [1] for such segmentation and each smaller unit of the sample is called a sub-stroke which is either a basic character/modifier/compound character or a part of such a symbol. For example, the input word sample shown in Fig. 11(a) has three strokes (indicated by red, green and blue colours). The segmentation algorithm detects six segmenting points (namely, S1, S2, …, S6) and segments the word sample in these six positions (Fig. 11(b)). The resultant sub-strokes are indicated by different colours. For example, the first stroke in Fig. 11(a) gets segmented into two sub-stroke samples.

Figure 10. Bangla symbol set considered in the present study.

Basic Vowels
অ আ ই ইউ গু এ ঐ ও ঔ

Basic Consonants
ক খ গ ঘ ঙ চ ছ ঞ ছঞ জ ঝ ঞ
ঞঝঞ ছঞ

Modifiers
া ি ও দূ ধূ তৃ

Compound Characters
কব্র ছব্র দব্র ঘব্র ঙব্র

dbhr
cbhr

Figure 11. (a) An input word sample having three strokes. (b) The word sample in (a) after segmentation into sub-strokes where each S represents a segmenting point and each E represents pen up/down position.

Figure 12. Headlines and baselines (indicated by red and blue colours respectively) of four word samples.
Identification of sub-stroke classes forming basic characters, modifiers and compound characters from unconstrained handwritten Bangla words

In the present study, the Bangla symbol set consists of all basic characters, all modifiers and several compound characters as shown in Fig. 10. When the basic input unit is a handwritten word sample, it needs to be segmented into smaller units in order to finally recognize the word sample. We use the algorithm proposed in [1] for such segmentation and each smaller unit of the sample is called a sub-stroke which is either a basic character/modifier/compound character or a part of such a symbol. For example, the input word sample shown in Fig. 11 (a) has three strokes (indicated by red, green and blue colours). The segmentation algorithm detects six segmenting points (namely, $S_1$, $S_2$, ..., $S_6$) and segments the word sample in these six positions (Fig. 11(b)). The resultant sub-strokes are indicated by different colours. For example, the first stroke in Fig. 11(a) gets segmented into two sub-stroke samples.

Figure 10. Bangla symbol set considered in the present study.

Figure 11. (a) An input word sample having three strokes. (b) The word sample in (a) after segmentation into sub-strokes where each S represents a segmenting point and each E represents pen up/down position.

Figure 12. Headlines and baselines (indicated by red and blue colours respectively) of four word samples.
Using language specific knowledge, we identified 114 sub-stroke classes in the Bangla script. In Fig. 13, these sub-stroke classes are shown. The printed forms of basic characters/modifiers/compound characters are shown in bold. On the right of each of these symbols, are shown the sub-stroke classes that may constitute the handwritten form of the corresponding symbol. On the basis of the sub-stroke samples that are present in a word sample, we find both the headline and the baseline of the word sample an artificial neural network model. It is observed that finding the baseline is more difficult than finding the headline. In Fig. 12, four word samples are shown in which the headlines and the baselines are indicated by red and blue colours respectively. In Fig. 12(a), the sample has no upper or lower zone; in Fig. 12(b), the sample has upper zone, but no lower zone while in Figs. 12(c) and 12(d), the samples have both upper and lower zones. Before we extract the features for recognition, we normalize the size and the position of a word sample in the following way. We set row numbers at the headline and at the baseline at 150 and 250 respectively, by scaling and shifting the (x, y) coordinates.

In order to extract the feature vector from a sub-stroke, we reduce the number of points lying on a sub-stroke so that the basic shape of the sub-stroke is preserved. An extremum point on the sub-stroke is defined as follows. Let \( p_i = (x_i, y_i) \). Let

- \( p_i \) is an extremum point if any of the following four conditions holds. Let
  - \( x_{i-1} < x_i < x_{i+1} \)
  - \( y_{i-1} < y_i < y_{i+1} \)
  - \( x_{i-1} > x_i > x_{i+1} \)
  - \( y_{i-1} > y_i > y_{i+1} \)

In all the above four cases, at least one inequality should hold. Let \( q_i \) denote the sequence of the extremum points in a sub-stroke sample. A sub-stroke sample is shown in Fig. 14(a) on the basis of the points \( p_i \) and the points \( q_i \) are shown in Fig. 14(b). It is to be noted that although the points \( q_i \) are much less in number than the points \( p_i \), the essential shape information of the sub-stroke sample is largely preserved by the points \( q_i \).

Let \( L_i \) be the length of the directional line segment starting at \( q_i \) and ending at \( q_{i+1} \) and \( \theta_i \) be the angle made by the movement from \( q_i \) to \( q_{i+1} \). Note that \( \theta_i \) belongs to \([0, 2\pi)\). The angles \( \theta_i \) are quantized into the eight codes in \( D \) (shown in Fig. 15). For example, the sequence of such codes for the sub-stroke sample of Fig. 14(b) is 186542114678.

For a sub-stroke sample, the features that are extracted from it are:

1. **Length**
2. **Width**
3. **Height**
4. **Minimum y-value in the sample**
5. **Maximum y-value**
6. **y-value of the first point in the sample**
7. **y-value of the last point**
8. **Average y-value**
9. **Length of the part of the sample in direction 1**
10. **Length of the part of the sample in direction 2**

Figure 13. Sub-stroke classes that a handwritten character (shown in bold) in a cursively written word may be composed of, are shown.
Using language specific knowledge, we identified 114 sub-stroke classes in the Bangla script. In Fig. 13, these sub-stroke classes are shown. The printed forms of basic characters/modifiers/compound characters are shown in bold. On the right of each of these symbols, are shown the sub-stroke classes that may constitute the handwritten form of the corresponding symbol. On the basis of the sub-stroke samples that are present in a word sample, we find both the headline and the baseline of the word sample an artificial neural network model. It is observed that finding the baseline is more difficult than finding the headline. In Fig. 12, four word samples are shown in which the headlines and the baselines are indicated by red and blue colours respectively. In Fig. 12(a), the sample has no upper or lower zone; in Fig. 12(b), the sample has upper zone, but no lower zone while in Figs. 12(c) and 12(d), the samples have both upper and lower zones. Before we extract the features for recognition, we normalize the size and the position of a word sample in the following way. We set row numbers at the headline and at the baseline at 150 and 250 respectively, by scaling and shifting the \((x, y)\) coordinates.

**Feature Extraction**

In order to extract the feature vector from a sub-stroke, we reduce the number of points lying on a sub-stroke so that the basic shape of the sub-stroke is preserved. An extremum point on the sub-stroke is defined as follows. Let \(p_{i-1}, p_i \) and \(p_{i-1} \) be three consecutive points on a sub-stroke. \(p_i \) is said to be an extremum point if any of the following four conditions holds. Let \(p_i = (x, y)\).

1. \(x_i \leq x_{i+1} \) and \(x_i \leq x_{i+1} \),
2. \(y_i \leq y_{i+1} \) and \(y_i \leq y_{i+1} \),
3. \(x_i \geq x_{i+1} \) and \(x_i \geq x_{i+1} \),
4. \(y_i \geq y_{i+1} \) and \(y_i \geq y_{i+1} \).

In all the above four cases, at least one inequality should hold. Let \(q_i \) \((i=1, 2, ..., n)\) denote the sequence of the extremum points in a sub-stroke sample. A sub-stroke sample is shown in Fig. 14(a) on the basis of the points \(p_i \) and the points \(q_i \) are shown in Fig. 14(b). It is to be noted that although the points \(q_i \) are much less in number than the points \(p_i \), the essential shape information of the sub-stroke sample is largely preserved by the points \(q_i \). Let \(L_i \) be the length of the directional line segment starting at \(q_i \) and ending at \(q_{i+1} \) and \(\theta_i \) be the angle made by the movement from \(q_i \) to \(q_{i+1} \). Note that \(\theta_i \) belongs to \([0, 2\pi)\). The angles \(\theta_i \) are quantized into the eight codes in \(D\) (shown in Fig. 15). For example, the sequence of such codes for the sub-stroke sample of Fig. 14(b) is 186542114678.

For a sub-stroke sample, the features that are extracted from it are

1. Length,
2. Width,
3. Height,
4. Minimum y-value in the sample,
5. Maximum y-value,
6. y-value of the first point in the sample,
7. y-value of the last point,
8. Average y-value,
9. Length of the part of the sample in direction 1,
10. Length of the part of the sample in direction 2,
(16) Length of the part of the sample in direction 8.

Let these 16 features be denoted by $U_1, U_2, \ldots$ and $U_{16}$ respectively.

Figure 14. (a) Equidistant points $p_i$ in a sub-stroke sample.
(b) Extremum points $q_i$ preserving the essential shape the sub-stroke sample in (a).

Figure 15. The set of directional codes $D = \{1, 2, \ldots, 8\}$
Classification

First phase sub-stroke classifier

A first phase sub-stroke classifier will now be developed on the basis of these features. For each sub-stroke class, we consider all the training samples. The 16 features $U_1, U_2, \ldots, U_{16}$ are extracted from all these samples and their mean values and standard deviations are computed. Let $M_i = (\mu_{i1}, \mu_{i2}, \ldots, \mu_{i16})$ denote the mean vector of the $i$-th sub-stroke class ($i = 1, 2, \ldots, 114$). Let $\sigma_{ij}$ denote the standard deviation of the $j$-th feature of the $i$-th class. For a test sample having feature vector $F = (u_1, u_2, \ldots, u_{16})$ the following distance is computed for each class $i$.

$$\text{Dist}_i = \frac{\sum_{j=1}^{16} (u_j - \mu_{ij})^2}{\sigma_{ij}^2}$$

For a given test sub-stroke classes are sorted in the increasing order of the above distance. The set of top 30 classes is selected and sent to the second phase sub-stroke classifier.

Second phase sub-stroke classifier

Here, we consider a Markov model for the second stage classification of a sub-stroke sample. The set of states here is $D = \{1, 2, \ldots, 8\}$ as shown in Fig. 15. A Markov model is specified by the initial state probability distribution and the state transition probability distribution. These two distributions need to be estimated during the training stage. This is done on the basis of a set of handwritten sub-stroke samples for each sub-stroke class. This is obtained from our sub-stroke database produced from the training set of 7,883 handwritten word samples.

For a sub-stroke class, we compute from each sample a sequence of states, say, $d_1, d_2, \ldots, d_k$. For example, the sequence of states for the sub-stroke sample of Fig. 14(b) is ‘186542114678’. For each sub-stroke class, the initial state probabilities $\pi_i$ ($1 \leq i \leq 8$) are computed as

$$\text{Number of sub-stroke samples for which } d_i = i$$

$$\text{Number of sub-stroke samples in the class}$$

The state transition probabilities $a_{ij}$ ($1 \leq i, j \leq 8$) are computed as

$$\text{Number of occurrences of } d_i = i \text{ and } d_{i+1} = j$$

$$\text{Number of occurrences of } d_i = i$$

For each of the sub-stroke class, we construct a Markov model on the basis of $\pi_i$ and $a_{ij}$ defined above.

Now, the probability of a sub-stroke sample with the state sequence ‘421146’ for the Markov model defined by $\pi_i$ and $a_{ij}$ above is $\mu_4 a_{42} a_{21} a_{11} a_{14} a_{46}$. Thus, for any arbitrary sub-stroke sample, we can find its state sequence and the probability of this state sequence under a Markov model.

Thus, for a test sample the first phase classifier produces top 30 sub-stroke classes. In the second phase, we compute the probabilities that the test sample is emanated from each of these top 30 sub-stroke classes. The test sample is classified into the sub-stroke class corresponding to the maximum probability computed during the second stage.

Symbol level and word level classification will be done next which are now in the designing stage.

References


