Bangla Handwritten Character Segmentation Using Structural Features: A Supervised and Bootstrapping Approach

TAPAN KUMAR BHOWMIK, Institute of Artificial Intelligence and Cognitive Engineering, University of Groningen, Netherlands
SWAPAN KUMAR PARUI, Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata, India
UTPAL ROY, Department of Computer and System Sciences, Visva Bharati, Santiniketan, India
LAMBERT SCHOMAKER, Institute of Artificial Intelligence and Cognitive Engineering, University of Groningen, Netherlands

In this article, we propose a new framework for segmentation of Bangla handwritten word images into meaningful individual symbols or pseudo-characters. Existing segmentation algorithms are not usually treated as a classification problem. However, in the present study, the segmentation algorithm is looked upon as a two-class supervised classification problem. The method employs an SVM classifier to select the segmentation points on the word image on the basis of various structural features. For training of the SVM classifier, an unannotated training set is prepared first using candidate segmenting points. The training set is then clustered, and each cluster is labeled manually with minimal manual intervention. A semi-automatic bootstrapping technique is also employed to enlarge the training set from new samples. The overall architecture describes a basic step toward building an annotation system for the segmentation problem, which has not so far been investigated. The experimental results show that our segmentation method is quite efficient in segmenting not only word images but also handwritten texts. As a part of this work, a database of Bangla handwritten word images has also been developed. Considering our data collection method and a statistical analysis of our lexicon set, we claim that the relevant characteristics of an ideal lexicon set are present in our handwritten word image database.

CCS Concepts: • **Applied computing** → **Document analysis**; **Optical character recognition**; Annotation; • **Computing methodologies** → *Supervised learning;*

Additional Key Words and Phrases: Supervised classification based segmentation, handwriting segmentation, annotation, structural features, SVM classifier, bootstrapping, Bangla handwriting database

ACM Reference Format:
Tapan Kumar Bhowmik, Swapan Kumar Parui, Utpal Roy, and Lambert Schomaker. 2016. Bangla handwritten character segmentation using structural features: A supervised and bootstrapping approach. ACM Trans. Asian Low-Resour. Lang. Inf. Process. 15, 4, Article 29 (April 2016), 26 pages.
DOI: http://dx.doi.org/10.1145/2890497

Authors' addresses: T. K. Bhowmik and L. Schomaker, Institute of Artificial Intelligence and Cognitive Engineering, University of Groningen, P.O. Box 407, 9700 AK Groningen, The Netherlands; emails: tkbhowmik@gmail.com, L.Schomaker@ai.rug.nl; S. K. Parui, Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, 203 B. T. Road, Kolkata - 700108, India; email: swapan@isical.ac.in; U. Roy, Department of Computer and System Sciences, Visva Bharati, Santiniketan - 731235, India; email: roy.utpal@gmail.com.

1. INTRODUCTION

Character segmentation is one of the key modules in a system for segmentation-based handwritten word recognition that seeks to decompose a word image into sub-images of meaningful individual patterns/symbols [A. El-Yacoubi and Suen 1999], called pseudo-characters [Koerich et al. 2003] or glyphs. Such a pseudo-character may be (i) an individual character, (ii) a part of a single character, (iii) parts of two adjacent characters, or (iv) two or more individual characters. Segmentation in the case of (ii) is known as oversegmentation and that in the case of (iv) is known as undersegmentation. In a system where segmentation leads to more over- and/or under-segmentation, a higher error rate is incurred. The segmentation task is more challenging and difficult when handwriting in a script is cursive. Although developing a successful handwriting recognition system seems to be a long-term goal, currently some less ambitious tasks such as recognition of postal addresses, bank checks, and the like have been undertaken as a handwritten word recognition problem. In a problem like postal address recognition, where the lexicon size is small, the number of address words such as post office and city names is not very large, and the recognition engine recognizes the address words directly without using any character segmentation. But when the lexicon size is large, it is very difficult to develop a recognition system without using segmentation. In recent years, more sophisticated techniques have been investigated for recognition of unconstrained handwritten texts mainly in Roman script [Alessandro Vinciarelli and Bunke 2004], where segmentation and recognition processes work interactively. The underlying predefined character models play an important role in making a decision on character boundaries as well as in generating possible hypotheses of characters within such a boundary. Commonly, a sliding-window technique, specific to a script, is used to focus on meaningful parts (ideally, a character or a part of character) of a word image in a specific order. The main drawback of this kind of approach is that sometimes, for a single word image, it can generate a huge number of character hypotheses that reduce the system's performance. The performance is not only in terms of accuracy but also in terms of recognition time. An efficient windowing technique can improve the system performance by reducing the number of hypotheses, but establishing an efficient windowing technique is one of the big challenges. In fact, it is an open issue in developing an unconstrained handwriting recognition system. One of the ways to establish an efficient windowing technique is to use predetermined segmentation knowledge. Thus, there is always an added advantage to incorporating a segmentation module in an unconstrained handwriting recognition system.

In this study, segmentation of a handwritten Bangla word image into meaningful individual symbols or pseudo-characters is addressed. The task considered here is quite difficult due to (i) the complex nature of individual Bangla basic characters (even in printed form) and (ii) the extremely cursive nature of Bangla handwriting. It is pertinent here to note that the cursiveness of the handwritten form of other major Indian scripts is much less than that of Bangla script. However, for implementing and testing the proposed segmentation algorithm, we developed a handwriting database as well. A set of the names of some small, medium, and large towns in the eastern region of India was used to build the database. Here, 119 such names were considered, and 300 handwritten samples for each such word were collected, covering a reasonably large spectrum of handwriting styles. Furthermore, a statistical analysis on the lexicon indicates that it contains several desirable properties of a Bangla corpus.

The existing segmentation algorithms are not usually treated as a classification problem. However, in the present study the segmentation algorithm is looked upon as a two-class supervised classification problem. Such an algorithm needs (i) a training set and a test set consisting of both segmenting and nonsegmenting points belonging to the word images and (ii) a set of feature vectors from these points. Due to the variation in handwriting style, one needs to generate a large set of segmenting and nonsegmenting points from the training set of word images. However, to manually generate such a large set is not practical. Here, we first start with a relatively small manually generated training set and build a classifier based on this training set, and then we generate more training samples using a semi-supervised bootstrapping technique on the basis of both the classifier and the earlier training set. More training samples will lead to a more robust classifier.

The article is organized as follows. Section 2 deals with the selection of a lexicon set of Bangla words and the development of a database of word images on the basis of the lexicon set. In Section 3, the basic segmentation techniques are discussed. Section 4 discusses the related works on Bangla character segmentation. The proposed segmentation method is described in Section 5, which also proposes a semi-automatic bootstrap technique for labeling new data. The procedure for segmenting a word image into characters is described in Section 6. The segmentation results and an analysis of the results are reported in Section 7. Finally, concluding remarks are made in Section 8.

2. DATABASE

The Bangla language is most widely used by a large number of people in the eastern part of India and Bangladesh. In fact, Bangla is the second most popular language in the Indian subcontinent and the fifth most popular language in the world. The Bangla handwriting has its own complexity due to its cursive nature. The complete Bangla character set consists of 11 basic vowels (1–11 characters), 39 basic consonants (12–50 characters), 10 vowel modifiers (51–60 characters) and 2 consonant modifiers (61–62 characters), as shown in Figure 1, and around 200 compound characters. The vowel modifiers correspond to the second to eleventh vowels shown in Figure 1(a). The first vowel character does not get modified.

Recent years have seen increased research interest in recognition of Bangla handwriting [Bhowmik et al. 2009; Bhattacharya et al. 2012] because of computerization at various levels of Indian administration. Some standard image databases of isolated Bangla handwritten characters (including numerals, basic characters, and vowel modifiers) have been developed [Bhattacharya and Chaudhuri 2005]. Recently, a database of handwritten Bangla and Bangla-English mixed script has been developed [Sarkar et al. 2012] that incorporates only a few writing styles from literate people. Apart from these, no standard image databases are available for unconstrained handwriting in Bangla text. Developing a representative database containing all the characteristics of Bangla script is not only difficult but also a tedious task, and it requires expert linguistic knowledge. But, as a less ambitious task, we developed a handwriting image database with a small lexicon set that possesses most of the characteristics of a large corpus of Bangla. This set features 300 writers of various levels of age, literacy, and profession. The database was developed for the present study and its possible extension for future work.

2.1. Selection of Lexicon Set

A set of names of small, medium, and large towns in the state of West Bengal (the Bangla-speaking state in India) is considered as the lexicon set for the development of an image database of handwritten Bangla words. The lexicon set is chosen to include most of the important basic characters and vowel modifiers. Eight rare basic characters (reference numbers: 4, 6, 7, 9, 11, 21, 47, and 49 in Figure 1) and one rare vowel modifier (reference number: 56) are kept out of the database because of the nonavailability of town names containing such characters, which together constitute less than 0.3% of a Bangla corpus [Chaudhuri and Ghosh 1998]. Furthermore, words of varying length are included in the lexicon set (see Table I).

অ	আ	ই	ঈ	উ	ঊ	ঋ	এ	ঐ	ও
1	2	3	4	5	6	7	8	9	10
ঔ									
11									

(a)

ক	খ	গ	ঘ	ঙ	চ	ছ	জ	ঝ	ঞ
12	13	14	15	16	17	18	19	20	21
ট	ঠ	ড	ঢ	ণ	ত	থ	দ	ধ	ন
22	23	24	25	26	27	28	29	30	31
প	ফ	ব	ভ	ম	য	র	ল	শ	ষ
32	33	34	35	36	37	38	39	40	41
স	হ	য়	ড়	ঢ়	ৎ	ং	ঃ	ঁ	
42	43	44	45	46	47	48	49	50	

(b)

কা	কি	কী	কু	কূ	কৃ	কে	কৈ	কো	কৌ
51	52	53	54	55	56	57	58	59	60

(c)

ক্য	র্ক								
61	62								

(d)

Fig. 1. Bangla character set with reference numbers (a) vowels, (b) consonants, (c) vowel modifiers involving the first consonant character, and (d) consonant modifiers involving the first consonant character.

2.2. Statistical Analysis of the Lexicon Set

Corpus-based statistical analysis of any language has real-life uses in the areas of OCR, language processing, cryptography, linguistics, generic content recognition, speech analysis, spelling error correction, and the like. Chaudhuri and Ghosh [1998] presented a statistical study for a Bangla corpus of 4.6 million words compiled from naturally occurring printed materials of different disciplines like literature, science, commerce, and electronic mass media. Their statistical analysis reflects the real-life occurrence of various words together with different word lengths and characters (vowels, consonants, vowel modifiers, and compound characters) of the Bangla corpus. The definition of word length here is the number of characters in a word excluding the vowel modifier(s). The compound characters are taken as single characters. In their study, it was observed that word lengths from 2 to 9 in the corpus cover around 99.5% of all the words. In the present study, we considered words having word lengths from 2 to 9 only. The percentages of occurrence of various word lengths in our lexicon set and in the study of Chaudhuri and Ghosh [1998] are shown in Table I.

Table I. Occurrence of Words with Different Lengths in Our Lexicon Set and in the Bangla Corpus [Chaudhuri and Ghosh 1998]

Word length	Percentage of occurrence in Bangla corpus [Chaudhuri and Ghosh 1998]	Percentage of occurrence in our lexicon set
1	0.16	0.00
2	11.54	4.27
3	22.75	19.66
4	25.37	36.75
5	18.33	24.79
6	10.96	10.26
7	6.10	1.71
8	3.05	1.71
9	1.42	0.85
11	0.23	0
12	0.09	0
13	0.03	0
14	0.01	0
15	0.00	0
16	0.00	0
17	0.00	0
18	0	0

Table II. Comparison of the Percentage of Occurrence of Vowels, Consonants, and Compound Characters in Chaudhuri and Ghosh [1998] with that in the Present Study

Characters	Percentage of occurrence in Chaudhuri and Ghosh [1998]	Percentage of occurrence in our lexicon set
Vowels	39.7	38.0
Consonants	52.9	61.0
Compound Characters	7.3	1.0

In our lexicon set, the vowels occur in 38.01% and the consonants occur in 60.95% of the cases. The comparison of the percentage of occurrence of vowels, consonants, and compound characters in our lexicon set with that in the Bangla corpus [Chaudhuri and Ghosh 1998] is shown in Table II. From the study of Chaudhuri and Ghosh [1998], it is observed that, in Bangla, the consonants and vowels (excluding the compound characters) occur in approximately 93% of the cases, a very large figure in comparison to the compound characters, whose number is around 200. We excluded the compound characters from our study because of their small percentage of occurrence and enormous complexity. The goal of the present study is to develop a method of segmentation as a part of a whole recognition system for handwritten words that involve an overwhelming majority of Bangla characters. However, the task here, though limited in scope, is quite challenging, keeping in mind the complexity of the cursive nature of Bangla handwriting.

The difference in the percentage of occurrence of individual characters (vowels, consonants, and vowel modifiers) present in our lexicon along with those in the Bangla corpus is shown in Figure 2. The distribution in Figure 2 shows that our lexicon set is able to represent the real-life occurrence of individual characters in Bangla.

2.3. Preparation of Image Database

For our database development, we used 300 native Bangla writers of various levels of age, literacy, and profession. Each of them wrote all the words in our lexicon set of 119 words. Attempts were made to cover a reasonably large spectrum of handwriting styles

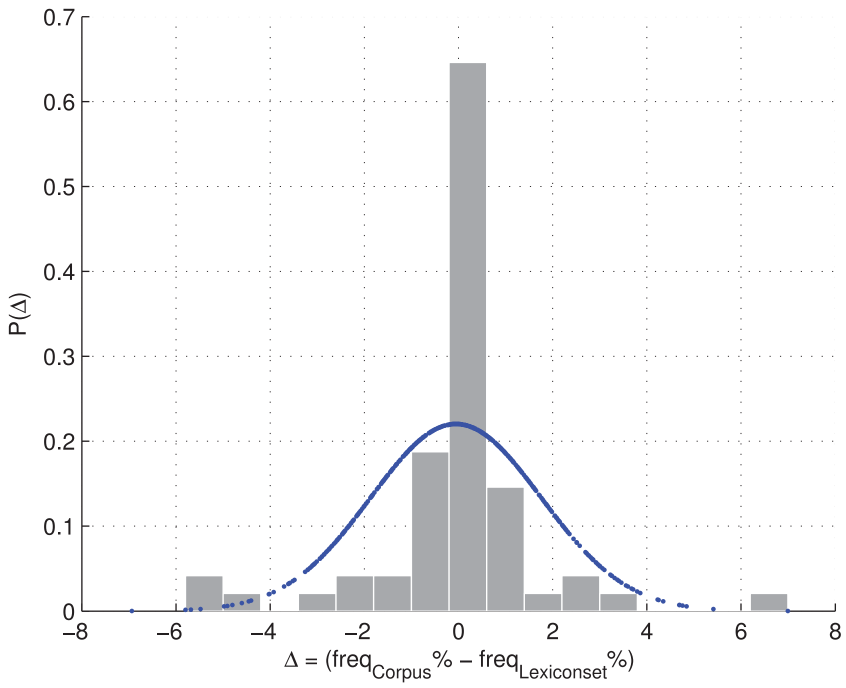


Fig. 2. Distribution of the difference between the Bangla corpus [Chaudhuri and Ghosh 1998] and our lexicon set at character level. Here, the difference in occurrences of individual characters is distributed with mean $m = -0.06$, standard deviation $\sigma = 1.89$, and skewness $\gamma = -0.58$.

in Bangla. These word images were scanned with a resolution of 300 dpi and stored as tiff images in grayscale format. In total, 35,700 word images form the database for our experiment. Considering our data collection method and statistical analysis of our lexicon set, we can claim that our database can be considered as a balanced database suitable for use in a research problem. All the words in the lexicon along with their reference numbers and two handwritten samples are shown in Figures 3 and 4.

3. BASIC SEGMENTATION TECHNIQUES

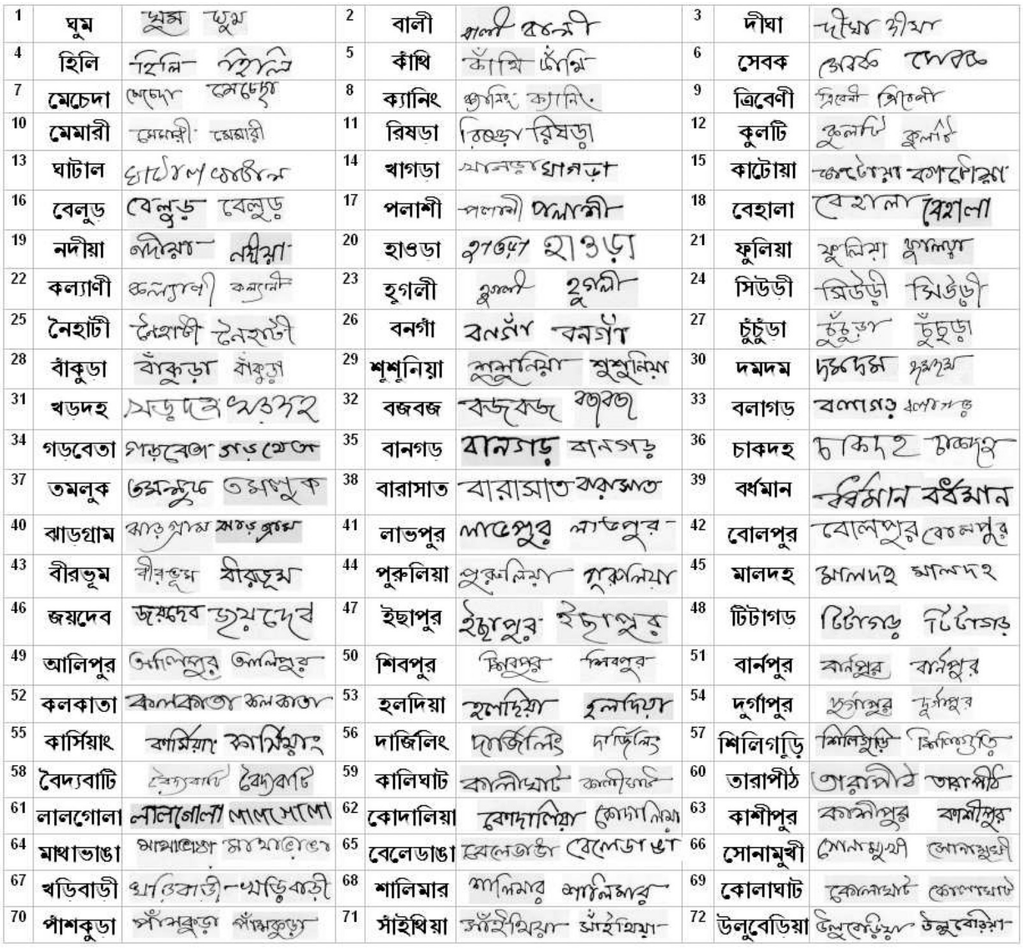
Segmentation algorithms can be classified into three main categories: region-based, contour-based, and recognition-based methods. A brief description of these categories is given here.

3.1. Region-Based Segmentation

The main idea in this method is first to identify the background regions and then to extract some features such as valleys, loops, reservoirs and the like from them. Top-down and bottom-up matching algorithms are used to construct the segmentation path by identifying such features. This type of work was reported in Simon [1992], Balestri and Masera [1988], and Pal and Datta [2003]. However, such methods tend to become unpredictable while segmenting connected characters that share a long segment. Also, the problem of extracting proper features from degraded images is common.

3.2. Contour-Based Segmentation

Several contour-based algorithms were proposed for character segmentation [Casey and van Horne 1992; Bozinovic and Srihari 1989; Kim and Govindaraju 1997; Koerich et al. 2003]. Casey and Horne's algorithm observes that when two characters



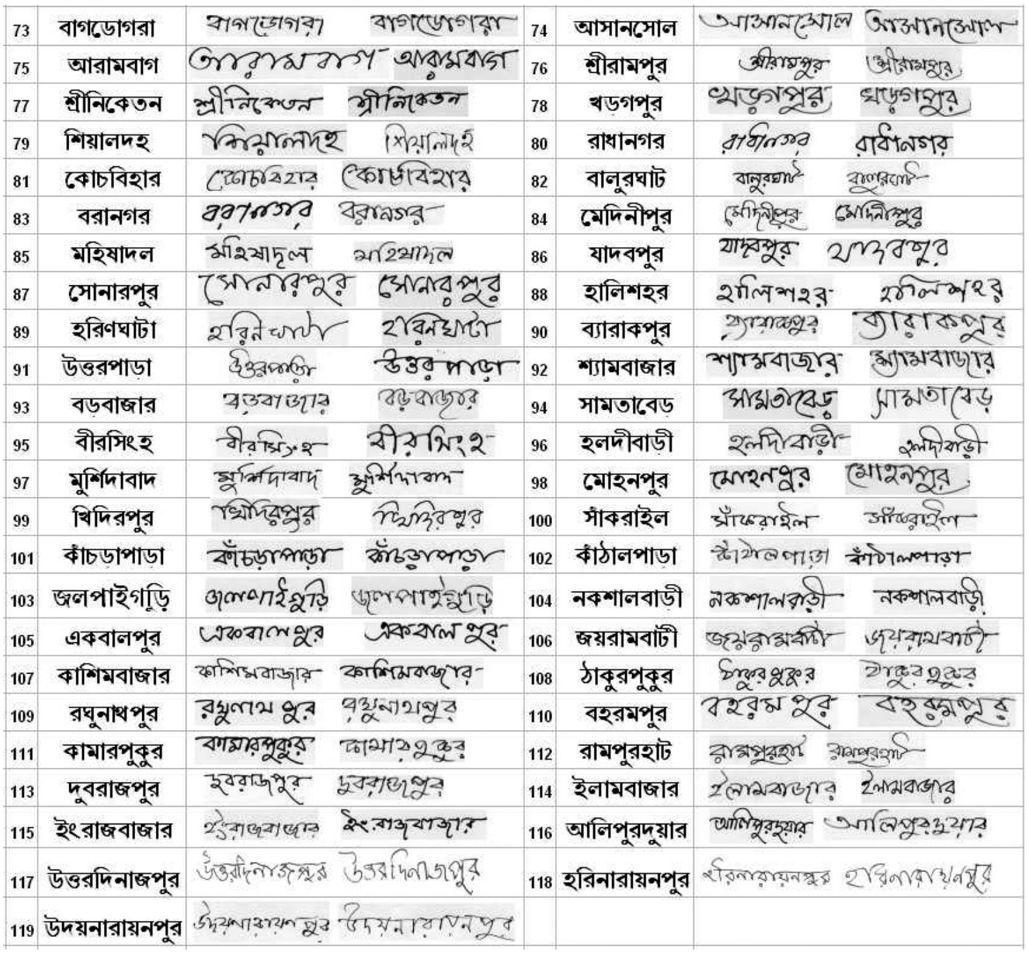
1	ঘুম	2	বালী	3	দীঘা
4	হিলি	5	কাঁথি	6	সেবক
7	মেচেদা	8	ক্যানিং	9	ত্রিবেণী
10	মেমারী	11	রিষড়া	12	কুলটি
13	ঘাটাল	14	খাগড়া	15	কাটোয়া
16	বেলুড়	17	পলাশী	18	বেহালা
19	নদীয়া	20	হাওড়া	21	ফুলিয়া
22	কল্যাণী	23	হুগলী	24	সিউড়ী
25	নৈহাটী	26	বনগাঁ	27	চুঁচুড়া
28	বাঁকুড়া	29	শুশুনিয়া	30	দমদম
31	খড়দহ	32	বজবজ	33	বলাগড়
34	গড়বেতা	35	বানগড়	36	চাকদহ
37	তমলুক	38	বারাসাত	39	বর্ধমান
40	ঝাড়গ্রাম	41	লাভপুর	42	বোলপুর
43	বীরভূম	44	পুরুলিয়া	45	মালদহ
46	জয়দেব	47	ইছাপুর	48	টিটাগড়
49	আলিপুর	50	শিবপুর	51	বার্নপুর
52	কলকাতা	53	হলদিয়া	54	দুর্গাপুর
55	কার্সিয়াং	56	দার্জিলিং	57	শিলিগুড়ি
58	বৈদ্যবাটি	59	কালিঘাট	60	তারাপীঠ
61	লালগোলা	62	কোদালিয়া	63	কাশীপুর
64	মাথাভাঙা	65	বেলেডাঙা	66	সোনামুখী
67	খড়িবাড়ী	68	শালিমার	69	কোলাঘাট
70	পাঁশকুড়া	71	সাঁইথিয়া	72	উলুবেড়িয়া

Fig. 3. Lexicon words with reference numbers and two handwritten samples (1–72).

are touching, there should be a concavity located at the point of contact. Therefore, the algorithm examines the middle section of a touching character image for concavities. Local curvature is estimated through edge direction, which is obtained by a contour-following algorithm. English handwritten characters are in general connected in the lower contour of a word image. For identifying the Pre-Segmented Points (PSP), Bozinovic and Srihari's algorithm first searches for local minima along the lower contour of the word. At every such local minimum, it then looks left and right within certain limits to locate zones in which PSPs are eligible to be placed. Each of these zones is characterized by both the sequence of single runs in vertical projection and a frequency of less than a predetermined threshold. If such a zone is found, a PSP is placed at the middle of it. A similar work is found in the literature [Schomaker et al. 2004] where a contour-based algorithm is used to fragment a connected component for automatic writer identification.

3.3. Recognition-Based Segmentation

Recognition-based segmentation algorithms [Casey and Nagy 1982; Kimura et al. 1993a, 1993b; Nohl et al. 1992; Yu et al. 2001; Bulacu et al. 2009] consist of two



73	বাগডোগরা	74	আসানসোল
75	আরামবাগ	76	শ্রীরামপুর
77	শ্রীনিকেতন	78	খড়গপুর
79	শিয়ালদহ	80	রাধানগর
81	কোচবিহার	82	বালুরঘাট
83	বরানগর	84	মেদিনীপুর
85	মহিষাদল	86	যাদবপুর
87	সোনারপুর	88	হালিশহর
89	হরিণঘাটা	90	ব্যারাকপুর
91	উত্তরপাড়া	92	শ্যামবাজার
93	বড়বাজার	94	সামতাবেড়
95	বীরসিংহ	96	হলদীবাড়ী
97	মুর্শিদাবাদ	98	মোহনপুর
99	খিদিরপুর	100	সাঁকরাইল
101	কাঁচড়াপাড়া	102	কাঁঠালপাড়া
103	জলপাইগুড়ি	104	নকশালবাড়ী
105	একবালপুর	106	জয়রামবাটী
107	কাশিমবাজার	108	ঠাকুরপুকুর
109	রঘুনাথপুর	110	বহরমপুর
111	কামারপুকুর	112	রামপুরহাট
113	দুবরাজপুর	114	ইলামবাজার
115	ইংরাজবাজার	116	আলিপুরদুয়ার
117	উত্তরদিনাজপুর	118	হরিনারায়নপুর
119	উদয়নারায়নপুর		

Fig. 4. Lexicon words with reference numbers and two handwritten samples (73–119).

steps: generation of segmentation hypotheses and choice of the best hypothesis. In this method, recognition is done iteratively according to the hypothesis generating the most satisfactory recognition score. Word-level knowledge may be utilized during the recognition process in the form of statistics, as a lexicon of possible words, or by a combination of these. Such methods are not only time-consuming but also their output depends heavily on the performance of the character recognition engine.

Apart from these three main categories, several other methods for handwritten character segmentation have been published [Yanikoglu and Sandon 1998; Casey and Lecolinet 1996; Lu and Shridhar 1996]. Maragoudakis et al. [2003] proposed a segmentation method where an initial PSP is detected through an analysis of the vertical histogram. For final segmentation, a Bayesian Belief Network is used as a classifier. Information on the position where the starting segment boundary intersects a character, the vertical histogram, the width of the current segment, and the position of the most horizontal and vertical strokes is used as the feature vector.

4. BANGLA HANDWRITTEN CHARACTER SEGMENTATION

Several research studies on character segmentation for non-Indian scripts have been reported, and satisfactory results have been obtained. However, to the best of our

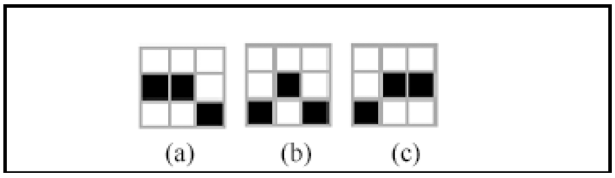


Fig. 5. Patterns observed in Bangla words.

knowledge, only a few studies on segmentation are available for Bangla handwritten scripts [Bishnu and Chaudhuri 1999; Pal and Datta 2003; Roy et al. 2005; Basu et al. 2007; Sankar et al. 2009]. These methods are based on conventional unsupervised approaches and are highly dependent on several handcrafted threshold values. As an initial work, we developed a method for segmentation of Bangla handwritten words into characters (see [Roy et al. 2005]) in an unsupervised manner. Since we will compare our proposed work with the previous study, we give a brief overview of the method and point out its advantages and disadvantages.

After a detailed study on cursive Bangla handwriting, it was observed that, in most cases, the connection between two adjacent characters occurs at the upper portion of the word sample. In other words, characters are connected along the waist line or *Matra/Headline* of a Bangla word. It may be noted that in English handwriting, the connection between characters is normally near the baseline. Furthermore, the contour fragment that connects two adjacent characters may take one of the patterns shown in Figure 5. These patterns are used to characterize the segmenting points throughout the word image. Hence, we identify all such points where the corresponding contour fragments follow any of these patterns. The set of possible segmenting points are now used to correct the skew of a word sample. For this purpose, a graphical path with the segmenting points is constructed based on several heuristics that emphasize certain characteristics of Bangla words. This graphical path is considered as an approximation to the Headline. In a de-skewed word sample, the Headline should ideally be a horizontal straight line segment. Therefore, the graphical path is aligned with the horizontal line as much as possible in order to correct the skew of the word sample. Finally, the segmenting points that belong to a defined zone around the graphical path are taken to be the candidate segmenting points. The word sample is segmented at these candidate points.

The advantage of this method is that, with only the graphical path, both skew correction and segmentation can be achieved. Unlike many other approaches, separate methods are not required for skew correction and segmentation. It has been observed that, given the proper thresholds, this procedure performs satisfactorily most of the time. However, it uses a large number of thresholds, and the main disadvantage is that the threshold values are determined in an ad-hoc manner and are very specific to the database. No formal method is used to select these threshold values. This issue is crucial since this method heavily depends on heuristics and provides no scope to introduce learning techniques for capturing uncertainty. In addition, this method does not address the issue of undersegmentation properly. Keeping these issues in mind, our next attempt was to solve the segmentation problem in a supervised way, where a classifier is learned to label the segmenting and nonsegmenting points [Bhowmik et al. 2005]. The improved classification performance of this attempt encouraged us to develop a more generic segmentation technique in a similar supervised fashion. The present work thus describes a segmentation scheme based on supervised selection of segmenting points with contour pattern matching. This procedure does not involve any heuristics and therefore overcomes the drawbacks of our previous attempt. Also, the issue of undersegmentation is satisfactorily addressed here.

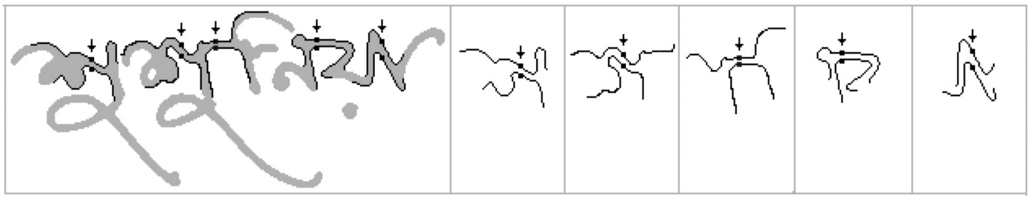


Fig. 6. Segmenting points (indicated by arrows) observed in the Bangla word image "*SHUSHUNIA.*" Segmenting points are characterized by the structural nature of their neighborhoods.

5. SUPERVISED CHARACTER SEGMENTATION BASED ON CONTOUR PATTERN MATCHING

In the present study, for a given contour point, we examine a number of preceding and following contour points (Figure 6)—called *a sequence pattern*—reflecting the structural nature of the neighborhood. Two such sequence patterns, one coming from the lower contour and the other coming from the upper contour, are said to indicate a candidate segmenting point if they satisfy certain conditions (to be provided later). A feature vector is computed from a candidate segmenting point. To find the candidate segmenting points and to extract features from these candidate segmenting points for an input word image, we first binarize the image and then apply the filtering technique to smooth the edges before contour tracing. Now, to form a training set, several candidate segmenting points are considered and labeled as "segmenting points" and "nonsegmenting points." A Support Vector Machine (SVM) is then trained on the basis of the training set and finally used to discriminate between segmenting and nonsegmenting points. A few threshold values (such as thickness of the writing pen and height of the *middle zone*) are used to select the initial set of candidate segmenting points. These threshold values are computed automatically from the word image. In the following sections, we define these threshold parameters and describe the procedure to compute them.

5.1. Preprocessing

In this section, we define some common features of Bangla handwritten words such as the thickness of the writing pen, the busiest zone (called *middle zone*), the *Matra* line, and the segmenting region, which are used as the threshold parameters for this segmentation process.

5.1.1. Thickness of Writing Pen. To find the thickness of the writing pen, a word image is scanned both row- and column-wise in order to get the runs of object pixels. The frequency distribution of the run-lengths is considered, and the most frequent run-length is called the thickness T of the writing pen.

5.1.2. Zone Detection. To determine the zones of a binary word image, a horizontal projection profile is analyzed. The projection is obtained in the form of α_i as follows:

$$\alpha_i = \begin{cases} 1, \text{ if } \left(m_i - \frac{1}{B}\sum_j m_j\right) > 0 \\ 0, \text{ otherwise} \end{cases} \tag{1}$$

where m_i is the number of object pixels in the i^{th} row of the word image and B is the number of object pixel rows of the word image (an object pixel row means a row that has at least one object pixel). Now, if H is the height or the number of rows of the word image, then both i and j range from 0 to $(H - 1)$. Obviously, if there is no non-object rows (i.e., having only zero entries), then the values of B and H are the same. The min and max indices of positive α_i give the middle zone boundaries of the word image.

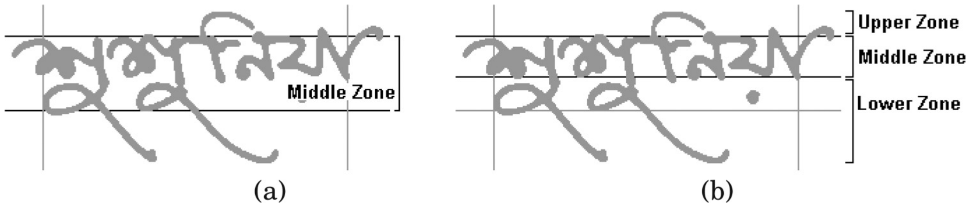


Fig. 7. For the word image "*SHUSHUNIA*", (a) middle zone detected by α_i, (b) zones detected by α'_i.

$\alpha_i > 0$ indicates that the i^{th} row is significant. The first significant row (say, row_u) from the top and the first significant row (say, row_l) from the bottom define the *middle zone*. The height of the middle zone is thus $h = (row_l - row_u + 1)$. The portion above the middle zone is called the *upper zone,* and the portion below is called the *lower zone* of the word image. In most of the cases, this measurement works satisfactorily. But for those cases when a vowel modifier itself has a long horizontal object pixel run (due to certain handwriting styles), the significant row (for which $\alpha_i > 0$) does not appear in the desired position, and, as a result, the zones are not detected properly (Figure 7(a)). To circumvent this problem, we adopt the following procedure. Let

$$\alpha'_i = \begin{cases} 1, & \text{if}\left(\frac{\alpha_i^{Run}}{\alpha^{RunMax}} - \zeta\right) > 0 \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

where α_i^{Run} = length of non-zero run at i^{th} position in $(\alpha_0, \alpha_1, \ldots \alpha_{H-1})$, α^{RunMax} = maximum length of non-zero runs in $(\alpha_0, \alpha_1, \ldots \alpha_{H-1})$, and $\zeta \geq 0$ is a bias term. For example, suppose α_i values are [0, 1, 0, 1, 1, 1, 0, 0]. Here, the length of the non-zero run at the 1^{st} position in α_i is 1, whereas at the 3^{rd} 4^{th} or 5^{th} positions in α_i is 3. So, α_i^{Run} values would be [0, 1, 0, 3, 3, 3, 0, 0]. It is clear that the middle zone gives rise to a long non-zero run in $(\alpha_0, \alpha_1, \ldots \alpha_{H-1})$. In fact, in most of the cases, the maximum length of a non-zero run corresponds to the middle zone. Thus, if the ratio $\frac{\alpha_i^{Run}}{\alpha^{RunMax}}$ is close or equal to 1, the i^{th} index falls in the middle zone. If the index i falls outside the middle zone, then the ratio is close to zero. It has been observed from our database that any value between $\zeta = 0.3$ and $\zeta = 0.5$ is quite appropriate to identify whether the indices i are in the middle zone or outside it. The minimum and maximum indices of positive α'_i give the *middle zone* boundaries of the word image. The word image "*SHUSHUNIA*" and its detected middle zones with α_i and α'_i measures are shown in Figure 7. Hence α'_i (rather than α_i) is used here for the detection of the middle zone.

5.1.3. Matra Region. The *Matra* region of a word image is defined as the region bounded by two imaginary horizontal lines corresponding to the rows $row_u - \frac{h}{2}$ and $row_u + \frac{h}{2}$. Here, row_u and h are the upper boundary of the *middle zone* and height of the *middle zone,* respectively, which were defined earlier. It is assumed that the upper boundary line of the middle zone divides the *Matra* region into two equal segments. We denote the matra region by M. The candidate segmenting points are assumed to fall within M.

5.1.4. Significant Contour Region. Consider a word image having more than one outer contour (this is possible when the word image has more than one connected component) as well as more than one inner contour (this happens when there are more than one hole). An outer contour region is the set of pixels enclosed by the contour including the contour pixels. An inner contour region is the set of pixels enclosed by the contour excluding the contour pixels. Suppose the outer contour regions are denoted by $R_1, R_2, \ldots$ and the inner contour regions by $R'_1, R'_2, \ldots$. Note that the outer contour region R_1 in Figure 8 has both object and background pixels, whereas the outer contour

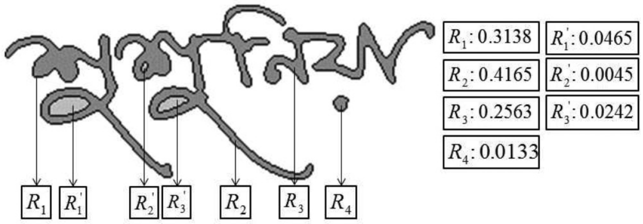


Fig. 8. Four outer contour regions R_1, R_2, R_3, and R_4 and three inner contour regions R_1', R_2', and R_3' occur in the word image "*SHUSHUNIA*." The significance values of these regions are shown on the right.

region R_3 in the same Figure has only object pixels. However, an inner contour region has only background pixels. Let $n(R_i)$ denote the number of object pixels in R_i and let $n(R_i')$ denote the number of pixels in R_i'. We now define a *significance value* for each region R (an outer or an inner contour region) as

$$sig(R) = \frac{n(R)}{max\left\{\sum_i n(R_i), n(R_1'), n(R_2'), \ldots\right\}}, (i = 1, 2, \ldots).$$

An inner contour region or an outer contour region is called a *significant region* with respect to the word image if its corresponding significance value is greater than a certain threshold. The motivation behind defining the significance value for a contour region is to decide whether the region needs to be considered for contour tracing. If the region is sufficiently small, it is unlikely to be useful in the segmentation process. From our database, it is observed that the threshold value of 0.01 works well in discarding such small regions. Figure 8 shows the contour regions and their corresponding significance values for the word image "*SHUSHUNIA.*" There are four outer contour regions denoted by R_1, R_2, R_3, and R_4 and three inner contour regions R_1', R_2', and R_3'. Now, if we round off the significance values to two decimal places, then, according to the threshold of 0.01, the regions R_4 and R_2' are not significant.

5.2. Detection of Candidate Segmenting Points by Contour Tracing

In a detailed study on Bangla handwriting, it is seen that the two consecutive characters (including vowel modifiers) and their connecting line form certain structural patterns. A few such patterns are shown in Figure 6. On the basis of these patterns, the candidate segmenting points are to be detected. In order to obtain these patterns, the outer and inner contours of the word images are traced first from the left-most object pixel to the right-most object pixel and then from the right-most object pixel to the left-most object pixel in the counterclockwise direction. If a word image has more than one connected component, then a similar tracing process is applied on each component. It is also to be noted that the left-most and right-most object pixels are chosen from the *middle zone* of the word image. The tracing process generates an 8-directional chain code (Figure 9) along with the positional information for an object pixel.

We now describe how the tracing is done. First, suppose P and Q are, respectively, the left-most and the right-most pixels of a connected component. We assume that the directional code at P is 1 (let us denote it by d); that is, if one arrives at P from the pixel on the left, which is a background pixel by definition. (This directional code of P is a dummy, and its true directional code will be determined at the end.) The task now is to find the next pixel on the contour. We search for it in the 8-neighborhood of P in the following manner. Let $d^* = d - 2$ (mod.8). (Here, in the mod.8 operation, we treat 0 as 8). We check if the pixel in the 8-neighborhood of P in the direction d^* is a background pixel. If it is, then d^* is set to $d^* + 1$ (mod.8). We repeat the process with the

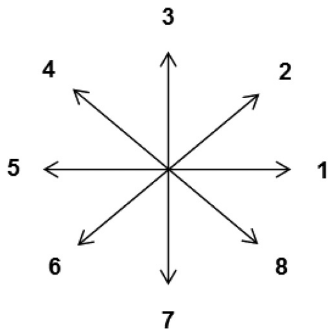


Fig. 9. Eight directional codes.

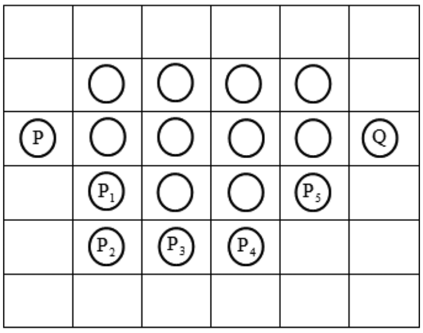


Fig. 10. The object pixels are marked with circles, and $P, P_1, \ldots, P_5, Q$ are the lower contour points that are visited during the tracing process in the counterclockwise direction. The chain code from P to Q is 8, 7, 1, 1, 2, 2.

new d^* and stop when an object pixel P_1 (in the 8-neighborhood of P) is encountered in the direction of d^* from P. Then, P_1 is the contour pixel next to P and P_1 has the directional code d^*. Then we set $d = d^*$ and find the contour pixel P_2 next to P_1 in a similar fashion by letting $d^* = d - 2$ (mod.8). This process continues until the right-most point Q is reached. Note that the tracing here is in the counterclockwise direction. For pixels $P_1, P_2, \ldots, Q$, we keep their directional codes and positions. For example, in the connected component shown in Figure 10, P and Q are the left-most and right-most points, respectively, of the connected component where $d = 1$ and $d^* = 7$. Now, the neighborhood pixel of P in direction 7 is a background pixel, and thus we increase d^* to 8. We see that the neighborhood pixel (P_1) of P in the direction 8 is an object pixel. So, we set $d = 8$, d^* becomes 6 and the contour code of P_1 becomes 8. In a similar fashion, we get the pixels P_2, P_3, P_4, P_5 and then reach Q. The chain code of the resulting contour is thus 8, 7, 1, 1, 2, 2.

Note that the contour just obtained is the lower contour of the connected component. In a similar fashion, we start tracing from Q in the counterclockwise direction to terminate at P. Note that the chain code at Q is already obtained as 2. Also, we obtain the chain code of P only at the end. The chain code of the upper contour is then $4, 5, 5, 5, 6$. Let us now consider a word image for a real-life illustration (Figure 11). The lower and upper contours of the third component in the image are given by $\widehat{C_3^{(B)} C_3^{(E)}}$ and $\widehat{C_3^{(E)} C_3^{(B)}}$ respectively. So far, we have discussed tracing the outer contour. Tracing of an inner contour will be similar.

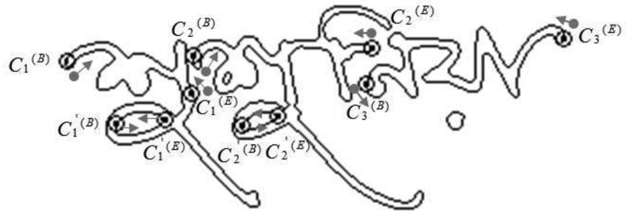


Fig. 11. Lower contours and upper contours for three connected components are shown for the word image "*SHUSHUNIA.*" There are three lower outer contours $\widehat{C_1^{(B)}C_1^{(E)}}$, $\widehat{C_2^{(B)}C_2^{(E)}}$, and $\widehat{C_3^{(B)}C_3^{(E)}}$ and three upper outer contours $\widehat{C_1^{(E)}C_1^{(B)}}$, $\widehat{C_2^{(E)}C_2^{(B)}}$, and $\widehat{C_3^{(E)}C_3^{(B)}}$ for the three components. Similarly, $\widehat{C_1'^{(B)}C_1'^{(E)}}$ and $\widehat{C_2'^{(B)}C_2'^{(E)}}$ and $\widehat{C_1'^{(E)}C_1'^{(B)}}$ and $\widehat{C_2'^{(E)}C_2'^{(B)}}$ are the two lower and two upper inner contours, respectively, that occur in the word image. The direction of tracing is shown with arrow marks.

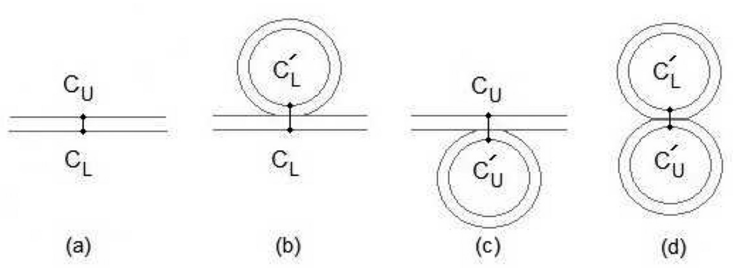


Fig. 12. Different scenarios for selecting candidate segmenting points.

Now, for each contour point, we have its directional code and position. Let us represent it as (d, x, y). Thus, the boundary of a connected component is represented as (d_1, x_1, y_1), $(d_2, x_2, y_2), \dots, (d_i, x_i, y_i), \dots,$ where $d_i \in \{1, 2, \dots 8\}$ is the directional code (Figure 9) and (x_i, y_i) are the coordinates of the i-th contour point. The index i increases as the tracing progresses. Let C and C' denote the sequence of the form $\{(d_i, x_i, y_i), i = 1, 2, \dots\}$ containing the outer and inner contour points, respectively. We now partition the sequence C into two disjoint sequences C_L and C_U (i.e., $C = C_L \cup C_U$), where C_L and C_U are the sequences containing all the lower and upper outer contour points of the word image, respectively. Similarly, C' is also divided into two disjoint sequences C'_L and C'_U. For a word image with one connected component, we obtain only a single outer contour, but the number of inner contours may be zero, one, or more. Suppose the inner contours are represented as C'_i $(i = 1, 2, \dots n)$, where n is the number of inner contours of a word image. Now, for each inner contour, we can write $C'_i = C'_{iL} \cup C'_{iU}$ as mentioned earlier. Now the goal is to see if a contour point of a word image can be identified as a candidate segmenting point. For this, a sequence S is considered that is a sequence of elements from the cartesian product of upper and lower contours. This sequence S of pairs of contour points is formed on the basis of the following criteria:

(1) Consider pairs of contour points (x_i, y_i) and (x_k, y_k) from $C_L \times C_U$ (cartesian product of lower and upper outer contours) that satisfy $(x_i, y_i) \in M$, $(x_k, y_k) \in M$, $|y_i - y_k| \leq (2 * T + 1)$ and $x_i = x_k$ (see Figure 12(a)). Here, M is the *Matra* region and T is the thickness of the writing pen.

(2) Consider pairs of contour points (x_i, y_i) and (x_k, y_k) from $C_L \times C'_L$ (cartesian product of lower outer contour and lower inner contour) that satisfy $(x_i, y_i) \in M$, $(x_k, y_k) \in M$, $|y_i - y_k| \le (2 * T + 1)$ and $x_i = x_k$ (see Figue 12(b)).
(3) Consider pairs of contour points (x_i, y_i) and (x_k, y_k) from $C_U \times C'_U$ (cartesian product of upper outer contour and upper inner contour) that satisfy $(x_i, y_i) \in M$, $(x_k, y_k) \in M$, $|y_i - y_k| \le (2 * T + 1)$ and $x_i = x_k$ (see Figure 12(c)).
(4) Consider pairs of contour points (x_i, y_i) and (x_k, y_k) from $C'_{rL} \times C'_{sU}$ $(r \ne s)$ (cartesian product of lower and upper inner contours) that satisfy $(x_i, y_i) \in M$, $(x_k, y_k) \in M$, $|y_i - y_k| \le (2 * T + 1)$ and $x_i = x_k$ (see Figure 12(d)).

The sequence S of pairs of points satisfying any of the just given four criteria is a sequence of candidate segmenting points. As far as the Bangla script is concerned, most of the points of S satisfy the first criterion. We include the second, third, and fourth criteria mainly to pick up the segmenting points for touching characters (Figure 14(b)). However, this has a drawback in that it may include oversegmenting points also into S because, in many cases, the character itself forms an inner contour region. However, in the case of touching characters where inner contours are present, the significance level of these inner contour regions generated by individual characters tends to be very low, and hence these inner contour regions are very likely to be removed.

So far, we have considered word images with a single connected component having more than one inner contour region. In case a word image has more than one connected component, this procedure is employed for each such connected component. Suppose, for n such components in a word image, the sequences of candidate segmenting points are $S_i, i = 1, 2, \ldots n$. Then, $S = \{S_1, S_2, \ldots S_n\}$. In Figure 11, there are three connected components in a word image from which three outer contours and two inner contours (generated from holes) are obtained. Each outer contour has two segments: namely, lower and upper contours which are denoted as $C_{iL} : \widehat{C_i^{(B)} C_i^{(E)}}$ and $C_{iU} : \widehat{C_i^{(E)} C_i^{(B)}}$, $(i = 1, 2, 3)$. Similarly, each inner contour has two segments as $C'_{jL} : \widehat{C'^{(B)}_j C'^{(E)}_j}$ and $C'_{jU} : \widehat{C'^{(E)}_j C'^{(B)}_j}$, $(j = 1, 2)$.

5.3. Skew Detection and Correction

In our previous algorithm (see [Roy et al. 2005]), we discussed both consistent and inconsistent skew estimation techniques. Here, we deal with only consistent skew estimation on the basis of the sequence S. Since most of the points in S are segmenting points, and they are close to the approximate *Matra* (waist line), the least square line fitted by the points of S is used to approximate the *Matra*. The angle between the approximate *Matra* and the horizontal line is defined as the skew angle (θ). If the skew angle is significant, then it is necessary to first de-skew the word image (as described earlier) and then perform again the whole process of finding the *upper*, *lower,* and *middle zones*; sequence S; and the approximate *Matra* region. Also note that the *Matra* line in a de-skewed word image is close to the horizontal line, which is denoted by $y = y_M$ for further reference.

5.4. Feature Extraction

Now, from the sequence S, we extract the feature vector for each point of S. As we mentioned earlier, a segmenting point has a neighborhood with certain structural patterns that are different from the patterns corresponding to a nonsegmenting point. In this study, these patterns are described as feature vectors from the chain code. Let us consider a pair of points $p = \{(d_i, x_i, y_i), (d_j, x_j, y_j)\}$ of S. Now, the points (d_i, x_i, y_i) and (d_j, x_j, y_j) come from different contour sequences. Suppose $(d_i, x_i, y_i) \in C_L$ and

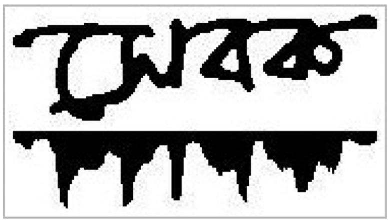


Fig. 13. The word image "*SEBAK*" and its vertical histogram.

$(d_j, x_j, y_j) \in C_U$. Now we consider $\mathcal{L}$ preceding contour points and $\mathcal{L}$ following contour points of (d_i, x_i, y_i) to form the sequence C_L. A similar set of points around (d_j, x_j, y_j) from C_U is considered. Here, we consider $\mathcal{L}$ to be equal to the height (h) of the *middle zone*. The frequencies of the directional codes of the preceding and the following contour points separately form a feature vector of the point (d_i, x_i, y_i). Since the directional codes belong to $\{1, 2, \dots 8\}$, the dimension of the feature vector is $(8 + 8) = 16$. The reason behind taking 8-directional codes of the preceding and following adjacent contour points is that these directional codes are sufficient to describe the pattern formed around the point (d_i, x_i, y_i). Similarly, we also extract the feature vector of dimension 16 for the point (d_j, x_j, y_j). The concatenation of both feature vectors constitutes the actual feature vector for p to determine if p is a segmenting point. However, three other features are added to distinguish between the segmenting and nonsegmenting points more accurately. These are described here:

(1) The positional information of a pixel is an important characteristic that can be used to identify relevant segmenting points. The segmenting region is around the *Matra*. So, pixels that are far from the *Matra* are less likely to be segmenting pixels. The positional information P of p is defined as

$$P = \frac{\mid \frac{(y_i+y_j)}{2} - y_M \mid}{h},$$

where $y = y_M$ is the approximate *Matra* line and y_i and y_j are the $Y-$ coordinates of the pair of points (x_i, y_i) and (x_j, y_j) of p. Here, h is the height of the *middle zone* and the division by h is done for normalization.

(2) Generally, a segmenting region is bounded by two long vertical strokes. Thus, the vertical profile analysis shows local minima at the segmenting regions (see Figure 13). From this observation, another component H_V is added to the feature vector:

$$H_V = \frac{\#\,(\textit{object pixels in the column containing } (x_i, y_i))}{h}.$$

(3) Finally, when considering vertical runs of object pixels, it is observed that the number of such runs at a segmenting point is in general smaller than that at a non-segmenting point. The number of vertical runs for p is denoted by

$$H_{Run} = \#\,(\textit{vertical runs of the object pixels in the column containing } (x_i, y_i)).$$

Thus, for a single segmenting point p, a feature vector of length 35 $(16 + 16 + 3)$ is extracted as: $(\frac{f_{li}^{(1)}}{\mathcal{L}}, \frac{f_{li}^{(2)}}{\mathcal{L}}, \dots \frac{f_{li}^{(8)}}{\mathcal{L}}, \frac{f_{ri}^{(1)}}{\mathcal{L}}, \frac{f_{ri}^{(2)}}{\mathcal{L}}, \dots \frac{f_{ri}^{(8)}}{\mathcal{L}}, \frac{f_{lj}^{(1)}}{\mathcal{L}}, \frac{f_{lj}^{(2)}}{\mathcal{L}}, \dots \frac{f_{lj}^{(8)}}{\mathcal{L}}, \frac{f_{rj}^{(1)}}{\mathcal{L}}, \frac{f_{rj}^{(2)}}{\mathcal{L}}, \dots \frac{f_{rj}^{(8)}}{\mathcal{L}}, P, H_V, H_{Run})$, where $f_{li}^{(k)}$ and $f_{ri}^{(k)}$ are the frequencies of the directional code k $(k = 1, 2, \dots 8)$ in the sequences of $\mathcal{L}$ preceding and $\mathcal{L}$ following directional codes in C_L respectively. Similarly, $f_{lj}^{(k)}$ and $f_{rj}^{(k)}$ are the frequencies of the directional code k $(k = 1, 2, \dots 8)$ in the

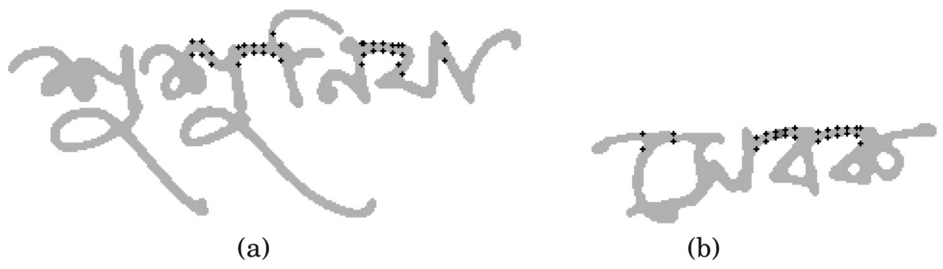


Fig. 14. Candidate segmenting points of the word images (a) "*SHUSHUNIA*" and (b) "*SEBAK.*"

sequences of $\mathcal{L}$ preceding and $\mathcal{L}$ following codes in C_U, respectively. It is to be noted that the feature vectors of the point (d_i, x_i, y_i) and its neighboring point in C_L are nearly the same. To remove this redundancy as much as possible, we take the points from C_L while maintaining a certain gap. In other words, instead of taking all the consecutive points, we skip some points before taking the next point. In a similar way, we extract the feature vectors for all the other selected points of S. The feature points extracted from the word image while maintaining a 5-pixel gap are shown in Figure 14.

On the basis of the features just described, the present segmentation task is considered as a 2-class supervised classification problem. One class represents the segmenting points and the other the nonsegmenting points. This characteristic of the problem prompted us to use the SVM, which is an ideal tool for two-class classification problems [Vapnik 1998].

5.5. Learning SVM Classifiers

SVM is a kind of machine learning algorithm based on recent advances in statistical learning theory. The SVM constructs a hyperplane in a high-dimensional feature space as the decision surface between two classes. All the segmenting feature vectors are considered as positive patterns and belong to the same class. Similarly, all the nonsegmenting feature vectors are considered as negative patterns belonging to the other class. The feature vectors corresponding to segmenting points are termed segmenting feature vectors, whereas the feature vectors corresponding to nonsegmenting points are termed nonsegmenting feature vectors. To train the SVM, we use Linear and RBF kernels with different values of γ.

5.5.1. Labeling Feature Vectors via Clustering. The task now is to prepare a set of segmenting and nonsegmenting feature vectors for training the classifier in a supervised manner. For this purpose, we select 100 samples at random from each word class and then extract 261,815 feature vectors corresponding to 261,815 candidate segmenting points from these 11,900 (100×119) handwritten word images. Manually annotation of these 261,815 points as segmenting and nonsegmenting points is a strenuous and time-consuming task. As a practical solution, we use the following semi-automatic annotation technique. First, we use the K-means clustering method to cluster the set (say, C) of 261,815 feature vectors into 100 clusters. We form as many as 100 clusters as we can, so that we can reasonably assume that each cluster is more or less homogeneous in the feature space. Then, a small number (say, $4q$) of points selected at random from each of the 100 clusters is manually annotated as segmenting or nonsegmenting points. If the number of such annotated segmenting points is more than $3q$ for a cluster, then all the points in that cluster are annotated as segmenting points. If the number of these segmenting points is less than q for a cluster, then all the points in that cluster are annotated as nonsegmenting points. In other words, we annotate an entire cluster when the confidence level is high. Otherwise, all the points in the

Table III. Performance of SVM Classifier

	Accuracy (%)		(μ, σ)	
SVM Kernel	Training	Test	Training	Test
RBF with $\gamma = 0.10$	88.72	87.14	(88.90, 1.31)	(86.71, 1.67)
RBF with $\gamma = 0.20$	88.91	87.49		
RBF with $\gamma = 0.40$	90.85	88.30		
Linear	87.15	83.92		

cluster are ignored during the annotation process. In this experiment, it is observed that any value of $q \geq 20$ works well in deciding whether the cluster is the segmenting or nonsegmenting cluster. Here, we considered the value of q as 25. In this way, we get 104,204 segmenting feature vectors and 47,755 nonsegmenting feature vectors (let C_{annot} denote the set consisting of these annotated feature vectors).

5.5.2. Classifying Feature Vectors with SVM. Now we would like to test the effectiveness of the semi-automatic annotation technique and also see how efficient the features defined in the preceding subsection are in discriminating between segmenting and nonsegmenting points. To do that, each of these two sets of feature vectors is again divided into two sets for training and test. A_{train} containing 70,000 samples randomly selected from 104,204 segmenting feature vectors and A_{test} containing the rest are respectively the training and test sets for the segmenting points. Similarly, B_{train} containing 32,000 samples randomly selected from 47,755 nonsegmenting feature vectors and B_{test} containing the rest are respectively the training and test sets for the nonsegmenting points. An SVM-based classifier is then built using the training set. For constructing the SVM, we use Linear and RBF kernels with values $\gamma = 0.10, 0.20,$ and 0.40. The RBF kernel with values $\gamma = 0.40$ gives the minimum test error for the present problem. The recognition rates for the test set with different kernels are shown in Table III.

5.6. Add-in Bootstrapping

To understand the misclassification behavior of the SVM-based classifier just used, we manually checked several of the samples that the classifier misclassified. We concluded that the enormous variability in Bangla handwriting was not accommodated in the training set on whichthe SVM classifier was built. We then decided to see if additional training samples can lead to better classifier training. To get more training samples, a semi-automatic bootstrapping approach is proposed to label additional samples. The algorithm is described as follows.

Let X_{train} be the training set of feature vectors defined earlier ($X_{train} = A_{train} \cup B_{train}$). Let us assume that $X = \{x_1, x_2, \ldots, x_N\}$ is a set of unannotated feature vectors extracted from the candidate segmenting points in S. In other words, X is a subset of $C - C_{annot}$.

(1) Select a sample x from X.
(2) Find r (r is an odd positive integer) nearest neighbor samples $x_{k_1}, x_{k_2}, \ldots, x_{k_r}$ from X_{train} using the Euclidian distance.
(3) Find the maximum occurring label from the nearest-neighbor samples.
(4) Assign the maximum occurring label to x.
(5) Repeat 1, 2, 3, and 4 for all samples in X.

Before incorporating new annotated feature vectors into the training set, we further classify each sample in X with the existing SVM classifier. If the sample gets the label "segmenting point" from both the nearest-neighbor algorithm (described above) and the SVM classifier, then it is included in A_{train}. Similarly, if the sample gets the label "nonsegmenting point" from both the nearest-neighbor algorithm and the SVM classifier, then it is included in B_{train} (so far, there is no manual intervention here).

Table IV. Performance of SVM Classifier after Bootstrapping

	Accuracy (%)		(μ, σ)	
SVM Kernel	Training	Test	Training	Test
RBF with $\gamma = 0.10$	90.92	90.40	(91.27, 0.99)	(90.53, 0.91)
RBF with $\gamma = 0.20$	91.94	91.17		
RBF with $\gamma = 0.40$	92.39	91.46		
Linear	89.82	89.10		

Otherwise, the sample is manually examined, annotated accordingly, and included in the expanded training set with a corresponding label. Here, the value of r is taken as 9.

As a case study, N was taken as 50,000. We did not take N as $\mid C - C_{annot} \mid$ since that would have resulted in a huge manual intervention that we could not afford. The expanded training sets A_{train} and B_{train} now have 105,981 and 46,019 samples, respectively. The SVM described earlier is trained again on the basis of the new training set of samples. However, the improvement in accuracy achieved by this is found to be marginal.

At this stage, we need to enlarge the training set further, but we cannot afford any manual intervention. In fact, our intention is to build a system that can start from a few training samples and automatically increase the training samples that can lead to building a more powerful classifier. Thus, we resort to the following way to increase the number of training samples. We consider here the rest of our database, which has 23,800 (200×119) word samples. From these samples, we compute the candidate segmenting points and the corresponding feature vectors. We then employ the earlier bootstrapping algorithm (excepting the manual intervention part) on these feature vectors. Here, if the nearest-neighbor algorithm and the SVM classifier do not agree on a sample, it is ignored and not included as a training sample. The SVM is finally trained based on this enlarged training set. The final recognition results on the previous test set (mentioned in Section 5.5.1) are shown in Table IV.

6. SEGMENTATION

The goal now is to segment arbitrary Bangla word images into characters or smaller character parts. An SVM classifier is designed to distinguish between the segmenting and nonsegmenting points on the basis of the feature vector extracted from the possible segmenting points (sequence S). Each possible segmenting point here is a pair of points along with their corresponding 8-directional codes. For any input image, we extract the sequence (S) of possible segmenting points by contour tracing. For each point of S, we then extract the feature vector and feed it to the SVM classier to identify whether the point is segmenting or not. Suppose $p^{(s)} = \{(d_i, x_i, y_i), (d_j, x_j, y_j)\} \in S$ is such a point classified as a segmenting point. Only positional information (x_i, y_i) and (x_j, y_j) of $p^{(s)}$ is used for segmentation. To segment the image, we draw a vertical line segment of a non-object pixel value from (x_i, y_i) to (x_j, y_j) (see Figure 15). In most cases, this is sufficient to segment the image. But in a few cases, as when a vowel modifier touches the character at the bottom or when two characters touch each other, the vertical line segment joining the two points is not sufficient to segment the image properly. In such cases, we extend the vertical line up to the top or the bottom of the image in such a way that one single connected component is segmented into two components (see Figure 15(b)). Our proposed scheme may produce multiple segmenting points in a single segment that requires only one segmenting point. These multiple points (excepting any one of them) may seem to be redundant, but these other points have some indirect role to play on the final selection of one point. Their presence not only improves the confidence level of the segment but also removes unnecessary connecting

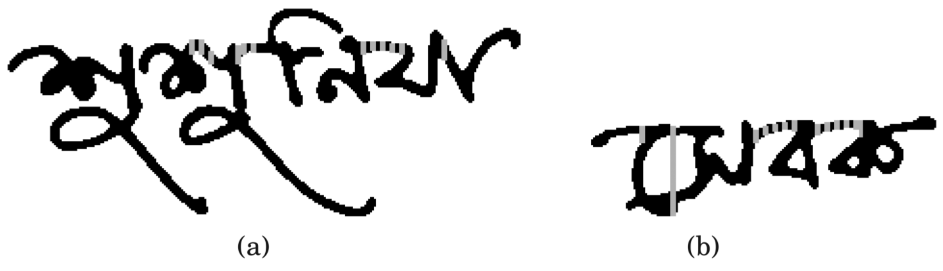


Fig. 15. Segmented word images (a) “*SHUSHUNIA*” and (b) “*SEBAK*.”

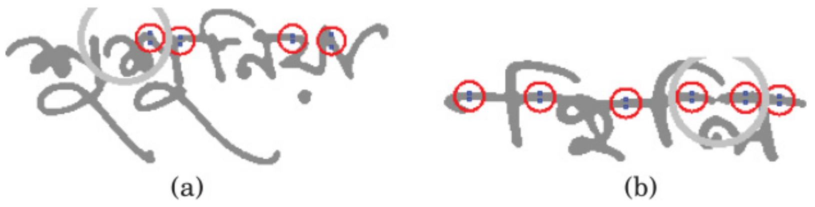


Fig. 16. The small red circles show the representative segmenting points. The large gray circle in (a) “*SHUSHUNIA*” shows an oversegmenting point, whereas the large gray circle in (b) “*SEBAK*” shows an appropriate point and a redundant point.

lines between characters. The two segmented word images produced by our algorithm are shown in Figure 15. Our next goal is to determine a representative segmenting point from among the multiple segmenting points present in a single segment.

6.1. Determination of Representative Segmenting Point of a Single Segment

To determine a representative segmenting point, we apply a clustering technique to choose a single one from a chain of multiple segmenting points. For this, a distance is defined between two consecutive segmenting points in the following way. Note that a segmenting point is composed of two points (one on the upper contour and the other on the lower contour). Now, for two consecutive segmenting points, we consider the distance between two corresponding upper contour points and the distance between two corresponding lower contour points. The maximum of these two distances defines the distance between two consecutive segmenting points. The distance between two contour points is taken here as the number of contour pixels lying between them. Now we join two consecutive segmenting points if the distance between them is less than a threshold (say, T). This will result in segments (or chains) of connected segmenting points (in fact, they are single linkage clusters). For each such connected segment (or chain), we find its middle segmenting point to represent the whole segment.

The choice of the threshold T is important here. If it is on the higher side, the multiple segmenting points lying on two adjacent segments may end up in a single connected segment (via the single linkage clustering just mentioned), which will result in one single segmenting point. However, the two adjacent segments need two segmenting points. To avoid such occurrences as much as possible, we keep the value of T somewhat low. In that case, an error of the opposite nature may occur. That is, one single segment may give rise to more than one connected segment (via single linkage clustering). In that case, we will get more than one segmenting point for a single segment. For example, in Figure 16(b), the gray circled segment requires only one segmenting point, although our algorithm produced two segmenting points (indicated by red circles). However, in such cases, any one of these segmenting points can do the segmentation task satisfactorily. Now, for user-based evaluation (provided in the next section), we mark any

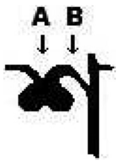


Fig. 17. Segmenting points are found at positions A and B for Bangla character SHA (Figure 1(40)). SHA can generate pseudocharacters in four possible ways. (i) ($\neg A$, $\neg B$): both A and B are unsegmented; (ii) ($\neg A$, B): B is segmented but not A; (iii) (A, $\neg B$): A is segmented but not B; and (iv) (A, B): both A and B are segmented.

one among these representative segmenting points as the potential segmenting point of a segment and the rest as redundant segmenting points. Now, a potential segmenting point can sometimes be an oversegmenting point. Otherwise, it is an appropriate segmenting point that correctly performs the segmentation task.

7. RESULTS AND DISCUSSIONS

The proposed method was applied to our Bangla word image database consisting of 35, 700 handwritten word samples. The database contains 119 names of towns/cities in West Bengal, each of which has 300 different writing samples. From the segmentation results obtained by the SVM classifier, we observe that the segmented parts of the word images do not always match a character or a modifier. In fact, there are both over- and undersegmentation. The segmented sub-images of the word samples are called pseudocharacters. The objective of our segmentation algorithm is to extract the pseudocharacter set so that it fits closely to the character/modifier set of the lexicon. We know that the number of pseudocharacters in undersegmentation is much higher than that in oversegmentation. While developing the algorithm, we put much emphasis on decreasing the chance of undersegmentation. A typical example of the possibilities of oversegmentation with our algorithm for the Bangla character "SHA" (40 in Figure 1) is shown in Figure 17. Here, two positions (A and B) are possible where the character can be segmented. If both A and B are unsegmented; that is, ($\neg A$, $\neg B$), the segmentation process generates pseudocharacter number 31 in Figure 19. But in the cases when ($\neg A$, B), (A, $\neg B$), and (A, B) occur, the generated pseudocharacter sets are {72, 58}, {64, 23}, and {64, 63, 58} in Figure 19, respectively.

In Table IV, we describe the results of segmentation at the feature level. The experimental results show that the proposed features are quite efficient in distinguishing between segmenting and nonsegmenting points. However, we need to evaluate the efficiency of the proposed segmentation scheme for segmenting a word into characters. To evaluate the result, we need a ground-truth data. But generating ground-truth data for handwritten character segmentation is a difficult task. In fact, it is impossible for scripts like Bangla, Devnagari, and others where two consecutive characters are connected via an unconstrained horizontal line/curve. Obviously, it is a tedious job, and, even for a single segmenting point, different people can mark the segmenting point at different places. In fact, any one point along the connection line between two consecutive characters could be a segmenting point. However, some studies report the segmentation results by visually inspecting the points (which may be possible for a few pages or words). Considering the inherent difficulties in such cases, here we evaluate our segmentation results in two different ways: (i) pseudocharacter-based evaluation and (ii) user-based evaluation. For pseudocharacter-based evaluation, we consider all the segmenting points ($p^{(s)}$) in order to generate character fragments, whereas for user-based evaluation, we consider only representative segmenting points.

Table V. The Segmentation Results Shown for 250 Word Images

Users	# Words	# Segmenting points found		# Potential segmenting points		# Redundant (R)	# Missing (M)	Accuracy ($\frac{A}{(A+O+M)} \times 100$)
		Before group	After group	# Appropriate (A)	# Over (O)			
1	50	2, 680	384	292	67	25	10	79.13
2	50	1, 890	335	258	62	15	5	79.39
3	50	1, 993	325	248	65	12	8	77.26
4	50	2, 340	340	265	52	23	10	81.04
5	50	2, 204	373	281	71	21	6	78.49
Overall accuracy								79.06

7.1. Pseudocharacter-based Evaluation

In this evaluation method, we calculate how close the generated fragments (pseudocharacter set) are to the original lexicon set. In our database, the numbers of basic characters, vowel modifiers, and compound characters are 42, 7, and 3 respectively. By applying our algorithm on our database, we get a set of pseudocharacters that is shown in Figure 19. This set of pseudocharacters is generated by grouping similar shapes manually. It is also noted that this set of pseudocharacters represents around 98% of the whole set of pseudocharacters generated by our segmentation scheme. The remaining (around 2%) pseudocharacters are considered noise pseudocharacters since they occur only once in the whole database and do not fit into any of the pseudocharacters described in Figure 19. From this figure, it is clear that our result has substantial agreement with the character/modifier set. Also, note that a resultant pseudocharacter is of the following types: (i) an individual character, (ii) a part of a single character, (iii) an individual modifier, (iv) an individual character with a modifier, or (v) a part of a modifier. Note that, based on a small lexicon set, the proposed segmentation algorithm has produced 75 pseudocharacters. A larger lexicon set is likely to contain more variations in character shapes due to newer character/modifier combinations, and the present algorithm may give rise to a larger set of pseudocharacters.

7.2. User-based Evaluation

For user-based evaluation, we selected five Bangla-speaking volunteers who are in the habit of writing in Bangla. Each was provided 50 sample word images randomly selected from our database. In our previous evaluation, all the segmenting points are used in order to generate pseudocharacters. But here the multiple segmenting points are clustered to generate a single segmenting point for a particular segment (see Section 6.1). The aim is to evaluate how the system performs in segmenting the word image into ideal characters. For selecting the candidate segmenting points, we maintain a 5-pixel gap between the previous point and the current point; that is, we ignore the pixels lying between the previous and current points. Here, for clustering, the associated threshold is chosen as two times the gap (i.e., a 10-pixel gap). All such consecutive candidate segmenting points that have a distance less than or equal to 10 form a cluster. This is in fact single-linkage clustering, and each such cluster produces a representative segmenting point. We asked a volunteer whether a representative segmenting point marked by the system is an appropriate, a redundant, or an oversegmenting point. We also asked him to count the number of missing segmenting points in the word image. Note that if two or more segmenting points occur in a single segment, one is considered as either an appropriate segmenting point or an oversegmenting point, and the rest are redundant segmenting points. Examples of over- and redundant segmenting points are shown in Figure 16(a) and Figure 16(b), respectively.

The detailed evaluation results for all users are shown in Table V. From the user-based evaluation in Table V, we found that 5.46% segmenting points are redundant,

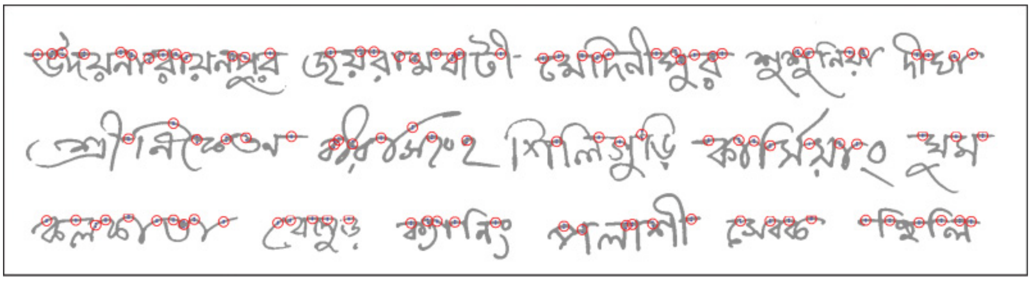


Fig. 18. A few segmentation results on word images from our database.

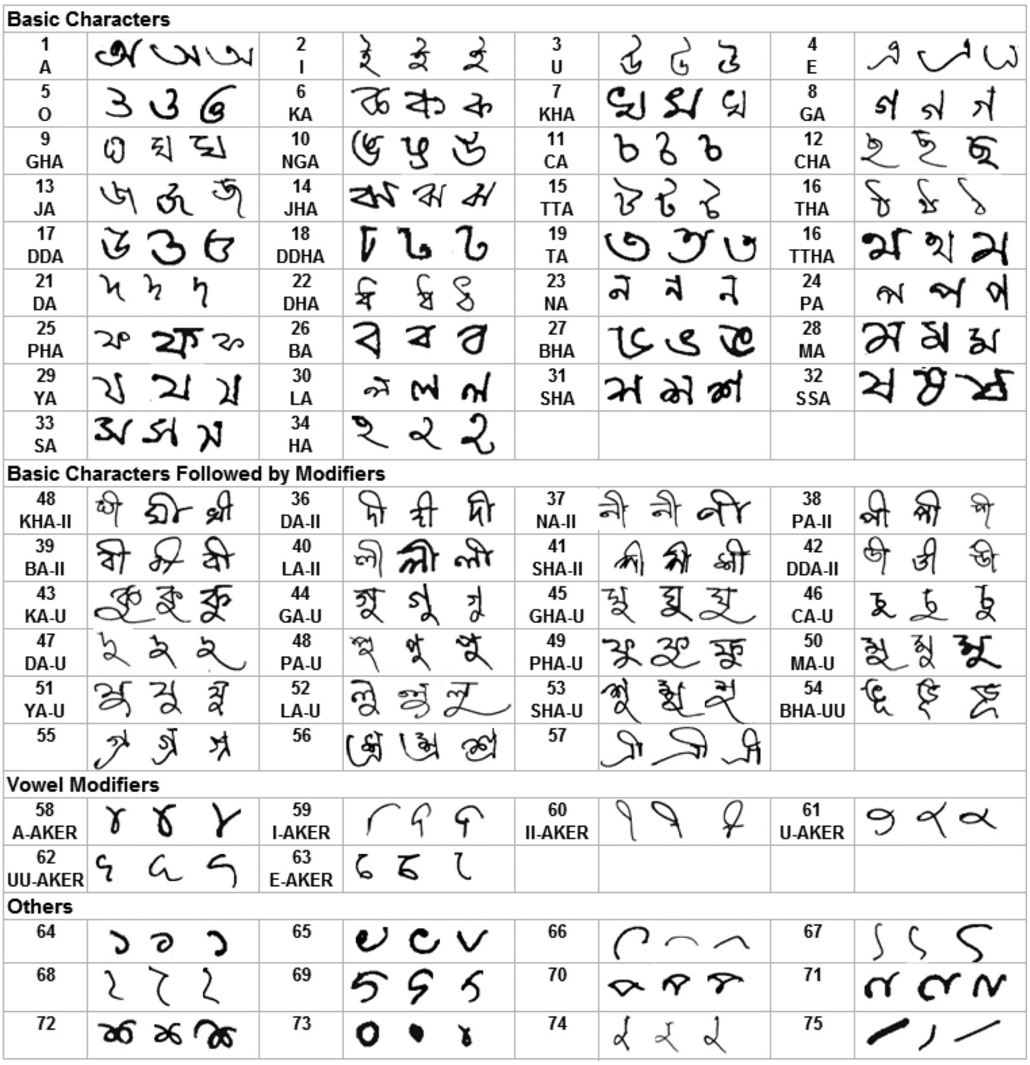
18.04% are oversegmenting points, and only 2.17% are undersegmenting points (i.e., the points where the image should be segmented but the algorithm failed to segment). Since the number of redundant points is less here, we can say that our clustering scheme is quite efficient. The undersegmentation results are impressive because our primary objective is to reduce undersegmentation as much as possible. Since the redundant points do not contribute to oversegmentation, we have not considered them for the computation of segmentation accuracies. Finally, if we consider the complexity of handwriting in Bangla, the overall segmentation accuracy of 79.06% is quite satisfactory in segmenting a word image into ideal characters. A few interesting segmentation results on word images from our database are shown in Figure 18 for the sake of illustration.

To compare our results with previous work [Roy et al. 2005], a similar user-based evaluation was performed. For each user, we selected the same 50 word images that were assigned in the earlier evaluation. We asked them to count the number of appropriate and oversegmenting points marked by the algorithm described in Roy et al. [2005]. In addition, we asked them to count the places where the segmenting points were not found by the algorithm. The statistics are based on feedback from the 5 users, and it is found that the number of appropriate segmenting points is 1, 171, the number of oversegmenting points is 255, and the number of instances of under-segmention is 143. Therefore, in this case, the percentage of undersegmentation, over-segmentation, and overall segmentation accuracies are 9.12%, 16.29%, and 74.59%, respectively. If we compare these results with our proposed method, it can be seen that our method reduces the undersegmentation percentage by around 7%. In fact, it meets one of our main objectives: that is, to reduce undersegmentation occurrences. Although the oversegmentation occurrences are slightly higher in our method, it enhances the overall segmentation accuracy by around 4.5%.

So far, we have described the segmentation results on our own database. Now we deploy our trained classifier on Bangla handwritten text. The performance of the segmentation result is shown in Figure 20, where the possible segmenting points are marked manually in Figure 20(a) and the segmenting points annotated by the system are marked in Figure 20(b). Here, each segmenting point consists of a pair of points, usually one coming from the lower contour and the other from the upper contour. Note also that the method does not use any line or word segmentation technique, but simply uses the connected components.

8. CONCLUSION

In this article, we described a framework for annotation and segmentation of Bangla handwritten word images into pseudocharacters. The novelty of this framework is that it is an unconventional supervised approach, and also it can annotate the segmenting



Basic Characters							
1 A		2 I		3 U		4 E	
5 O		6 KA		7 KHA		8 GA	
9 GHA		10 NGA		11 CA		12 CHA	
13 JA		14 JHA		15 TTA		16 THA	
17 DDA		18 DDHA		19 TA		16 TTHA	
21 DA		22 DHA		23 NA		24 PA	
25 PHA		26 BA		27 BHA		28 MA	
29 YA		30 LA		31 SHA		32 SSA	
33 SA		34 HA					
Basic Characters Followed by Modifiers							
48 KHA-II		36 DA-II		37 NA-II		38 PA-II	
39 BA-II		40 LA-II		41 SHA-II		42 DDA-II	
43 KA-U		44 GA-U		45 GHA-U		46 CA-U	
47 DA-U		48 PA-U		49 PHA-U		50 MA-U	
51 YA-U		52 LA-U		53 SHA-U		54 BHA-UU	
55		56		57			
Vowel Modifiers							
58 A-AKER		59 I-AKER		60 II-AKER		61 U-AKER	
62 UU-AKER		63 E-AKER					
Others							
64		65		66		67	
68		69		70		71	
72		73		74		75	

Fig. 19. Pseudocharacter set with reference numbers (pseudocodes).

points with minimal manual intervention, a topic that has not so far been investigated for this type of problem. Moreover, this is a very generalized framework. It will work for other *Matra-based* Indian scripts such as Devnagari. It will also work for Roman script with some minimal changes. For training an SVM classifier, an unannotated training set is generated first from the candidate segmenting points; this is then clustered and each cluster is labeled with minimal manual intervention. A semi-automatic bootstrapping technique is then employed to enlarge the training set. The objective is to build a system that can start with a few training samples and generate more training samples automatically when a new set of data is encountered. As a part of this work, a database of Bangla handwritten word images has been developed that incorporates the relevant characteristics of an ideal Bangla lexicon set. Although our primary focus of this work is to develop a segmentation scheme for Bangla handwritten words, the segmenting

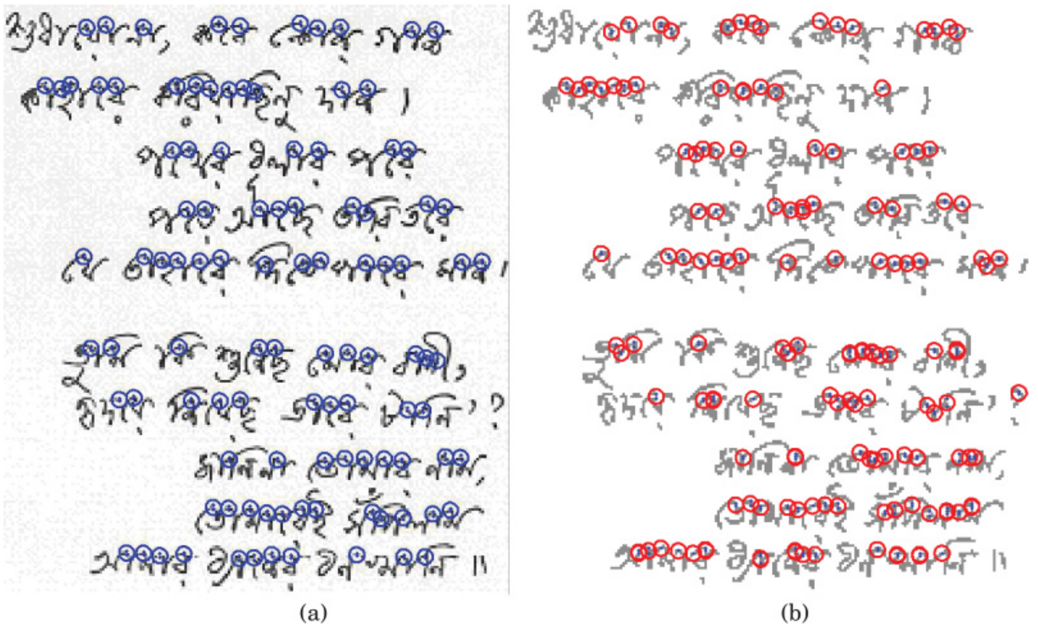


Fig. 20. (a) Manually annotated segmenting points in blue. (b) System-annotated segmenting points in red.

points can be used to develop a better windowing technique for a segmentation-free HMM-based recognition system, which we intend to investigate in the future.

REFERENCES

R. Sabourin A. El-Yacoubi, M. Gilloux, and C. Y. Suen. 1999. An HMM-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21, 8 (1999), 752–760.

Samy Bengio Alessandro Vinciarelli and Horst Bunke. 2004. Offline recognition of unconstrained handwritten texts using hmms and statistical language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26, 6 (2004), 709–720.

M. Balestri and L. Masera. 1988. A system for isolating characters in cursive script. *Signal Processing IV: Theories and Applications* (1988), 845–846.

S. Basu, R. Sankar, N. Das, M. Kundu, M. Nasipuri, and D. K. Basu. 2007. A fuzzy technique for segmentation of handwritten bangla word images. In *Proceedings of the 7th International Conference on Computing: Theory and Application (ICCTA)*. IEEE, 367–371.

U. Bhattacharya and B. B. Chaudhuri. 2005. Database for research on recognition of handwritten characters of indian scipts. In *Proceedings of the 8th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 322–326.

Ujjwal Bhattacharya, Malayappan Shridhar, Swapan K. Parui, P. K. Sen, and B. B. Chaudhuri. 2012. Offline recognition of handwritten Bangla characters: An efficient two-stage approach. *Pattern Analysis Applications* 15, 4 (2012), 445–458.

Tapan Kumar Bhowmik, Pradip Ghanty, Anandarup Roy, and Swapan Kumar Parui. 2009. SVM-based hierarchical architectures for handwritten Bangla character recognition. *International Journal on Document Analysis and Recognition (IJDAR)* 12, 2 (2009), 97–108.

T. K. Bhowmik, A. Roy, and U. Roy. 2005. Character segmentation for handwritten Bangla word recognition using artificial neural networks. In *Proceedings of International Workshop on Neural Networks and Learning in Document Analysis and Recognition (NNLDAR)*. 28–32.

Arijit Bishnu and B. B. Chaudhuri. 1999. Segmentation of Bangla handwritten text into characters by recursive contour following. In *Proceedings of the 5th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 402–405.

R. M. Bozinovic and S. N. Srihari. 1989. Off-line cursive script word recognition. *IEEE Transactions on Pattern Analysis Machine Intelligence* 11 (1989), 68–83.

Marius Bulacu, Axel Brink, Tijn van der Zant, and Lambert Schomaker. 2009. Recognition of handwritten numerical fields in a large single-writer historical collection. In *Proceedings of the 10th International Conference of Document Analysis and Recognition*. IEEE, 808–812.

R. G. Casey and E. Lecolinet. 1996. A survey of method and strategies in character segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18, 7 (1996), 690–706.

R. G. Casey and G. Nagy. 1982. Recursive segmentation and classification of composite character patterns. In *Proceedings of the 6th International Conference on Pattern Recognition*. 1023–1026.

R. G. Casey and J. van Horne. 1992. Segmenting of touching characters in postal addresses. In *U.S. Postal Service 5th Advanced Technical Conference*, Vol. 3. Washington, DC, 743–745.

B. B. Chaudhuri and S. Ghosh. 1998. A statistical study of Bangla corpus. In *Proceedings of the International Conference on Computational Linguistics, Speech and Document Processing (ICCLSDP)*. CVPR Unit, Indian Statistical Institute, Kolkata, C32–C37.

G. Kim and V. Govindaraju. 1997. A lexicon driven approach to handwritten word recognition for real-time applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19, 4 (1997), 366–379.

F. Kimura, M. Shridhar, and Z. Chen. 1993a. Improvements of a lexicon directed algorithm for recognition of unconstrained handwritten words. In *Proceedings of the 2nd International Conference on Document Analysis and Recognition*. IEEE, 18–22.

F. Kimura, M. Shridhar, and N. Narasimhamurthi. 1993b. Lexicon directed segmentation-recognition procedure for unconstrained handwritten words. In *Proceedings of the 3rd International Workshop on Frontiers in Handwriting Recognition*. 122–131.

Alessandro L. Koerich, Robert Sabourin, and Ching Y. Suen. 2003. Lexicon-driven HMM decoding for large vocabulary handwriting recognition with multiple character models. *International Journal of Document Analysis and Recognition (IJDAR)* 6 (2003), 126–144.

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

M. Maragoudakis, E. Kavallieratou, N. Fakotakis, and G. Kokkinakis. 2003. An effective stochastic estimation of handwritten character segmentation bounds. In *Competitive Environment, Renewable Energy, Distributed Generation.*

C. R. Nohl, C. J. C. Burges, and J. I. Ben. 1992. Character-based handwritten address word recognition with lexicon. In *Proceedings of the U.S. Postal Service 5th Advanced Technical Conference*, Vol. 3. Washington, DC, 167–182.

U. Pal and Sagarika Datta. 2003. Segmentation of Bangla unconstrained handwritten text. In *Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 1128–1132.

A. Roy, T. K. Bhowmik, S. K. Parui, and U. Roy. 2005. A novel approach to skew detection and character segmentation for handwritten Bangla words. In *Proceedings of International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 203–210.

R. Sankar, N. Das, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu. 2009. A two-stage approach for segmentation of handwritten Bangla word images. In *Proceedings of the 11th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 403–408.

R. Sarkar, N. Das, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu. 2012. CMATERdb1: A database of unconstrained handwritten Bangla and Bangla-English mixed script document image. *International Journal of Document Analysis and Recognition (IJDAR)* 15 (2012), 71–83.

Lambert Schomaker, Marius Bulacu, and Katrin Franke. 2004. Automatic writer identification using fragmented connected-component contours. In *Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition (IWFHR)*. IEEE, 185–190.

J. C. Simon. 1992. Off-line cursive word recognition. In *Proceedings of the IEEE 80(7)*. IEEE, 1150–1161.

V. Vapnik. 1998. *Statistical Learning Theory*. Wiley, New York.

Berrin Yanikoglu and Peter A. Sandon. 1998. Segmentation of off-line cursive handwriting using linear programming. *Pattern Recognition* 31, 12 (1998), 1825–1833.

M. L. Yu, P. C. K. Kwok, C. H. Leung, and K. W. Tse. 2001. Segmentation and recognition of Chinese bank check amounts. *International Journal of Document Analysis and Recognition (IJDAR)* 3 (2001), 207–217.

Received February 2015; revised December 2015; accepted January 2016