A Complete Bengali OCR: A Novel Hybrid Approach to Handwritten Bengali Character Recognition

Abstract

A complete Bengali OCR is presented incorporating a novel hybrid approach to handwritten Bengali character recognition. The idea is to combine structural analysis and template matching techniques in order to recognise the handwritten Bengali characters. Handwritten Bengali characters are inherently cursive and their is an absence of well-defined strokes. In this approach, the character set has been separated into different distinct sub-classes based one some distinguishable structural features. Details of several approaches to detect these structural features are presented. Structural and syntactic features have been generated from the training samples to generate distinct character signatures. A match dictionary has been devised based on these character signatures that helps in collecting multiple prototypes from the training samples. A revised form of continuity analysis is applied to match the test characters to the characters in the match dictionary. This complete OCR has been thoroughly implemented and tested and very promising results have been achieved in this direction.

Keywords: Handwritten Bengali character recognition, hierarchical structure, syntactic recognition.

1 Introduction

Automatic processing and analysis of document images is rapidly becoming one of the most important fields in pattern recognition and machine vision applications. In recent years there has been a trend to the formalisation of a methodology for recognising the structures of various types of documents in the framework of document understanding, since the whole process of document understanding is too complex to be covered by a single specialised approach. Other fields that are closely related to this relatively new applied field are the development of standard databases, compression and decompression techniques, cross validation, image filtering and noise removal, fast information retrieval systems, document segmentation and, above all, recognition of alpha-numeric characters. All these fields are closely interrelated. Numerous research work has been done on the Roman character set and very efficient character recognition systems are now commercially available. Much effort has also been made to recognise Chinese characters because of the fact that scientist visualised the task of Chinese character recognition as the ultimate goal in character recognition. Unfortunately, very few efforts has been made so far to recognise the characters commonly found in the Indian sub-continent. This paper presents an approach to the formation of a complete character recognition system to recognise hand-printed Bengali characters.

2 Research on Indian Character Recognition

There are numerous scripts in use in different areas within the Indian sub-continent. Not many attempts have been carried out on the recognition of these character sets. Some works are reported on Devanagari and Tamil character recognition. Some attempts have also been made on Brahmi, a script widely used all over India during the third century B.C., Telegu and Bengali characters. Sethi and Chatterjee[SC76] have presented a Devanagari numeral recognition scheme. They have used the presence and absence of four basic primitives for recognising the characters with the help of a decision tree. These four basic primitives were horizontal line segment, ver-

tical line segment, right slant and left slant. They also made use of interconnections of these basic primitives. Later Sethi[Set77] attempted recognition of constraint hand-printed Devanagari script using a similar method. Sinha[Set87] has carried out a few notable work in Devanagari script recognition. He presented a syntactic pattern analysis system with an embedded picture language for Devanagari script recognition. The system stores structural descriptors for each symbol of the script in terms of primitives and their relationships. The recognition involves s search for the unknown character primitives based on the stored description and context. Sinha suggested knowledge-based contextual post-processing systems for Devanagari test recognition. Chinnuswamy and Krishnamurthy[CK80] carried out an investigation into the recognition of hand-printed Tamil characters. Later Chandrasekaran *et al.*[CCS84] carried out detailed investigation on the recognition of hand-printed Tamil characters.

Recognition of Bengali characters have also been investigated. An early attempt in this direction was the work of Ray and Chatterjee[RC84]. They presented a recognition system based on a nearest neighbour classifier employing features extracted by using a string connectivity criterion. Sattar[Sat90] in his thesis used the decision-theoretic approach for recognising Bengali printed alpha-numerical characters. The scheme uses the template matching technique for recognition. The input pattern with unknown classification is compared with a set of templates or prototypes, one for each class, defined previously during the training phase. Mia[Mia92] used a syntactic approach to extract features expressed in terms of morphs and then compiling a string of morphs that was used as a prototype code. He applied the method on a single font printed Bengali character set. The numerical codes were constructed from the relationship among the strokes representing the structure of the characters. There were several groups of characters which produced the same code due to structural similarity among them. These characters were later distinguished by comparing the stroke ratio of a specific stroke for different members of the same group. Sattar and Rahman[SR89] in their paper discussed different problems of the recognition of printed Bengali characters by applying the template matching method. They discussed the

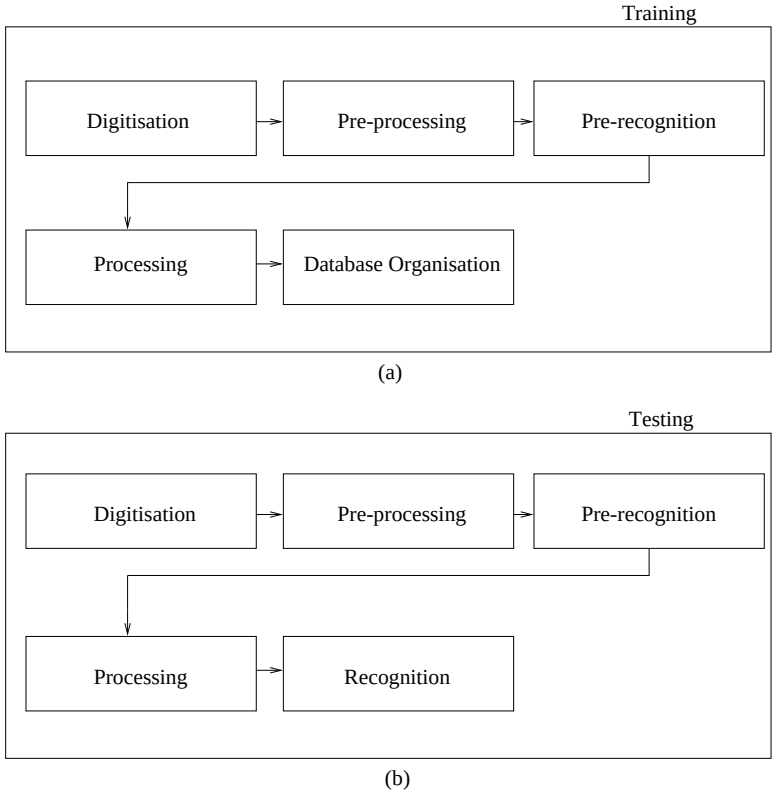

Figure 1: Organisation of the character recognition system.

optimum number of training set for designing the templates and the mismatch threshold to be used to avoid the problem of misclassification.

Curvature features have also been used by Dutta and Chaudhuri[DC93] in recognising hand-written and/or printed multi-font alphanumeric Bengali characters. They have represented the characters in terms of the primitives and structural constraints between the primitives imposed by the junctions present in the characters. A two-stage feed forward neural network was used as the basic recognition scheme. Pal and Chaudhuri[PC95], on the other hand, have presented a complete OCR system for documents of single printed Bengali font, where they have reported recognition of more than three hundred character shapes by a combination of template and feature-matching approach. Very recently Chaudhuri and Pal[CP97] have proposed an OCR system to read two printed Indian language scripts, Bengali and Devanagari.

3 Organisation of the Character Recognition System

The organisation of the proposed handwritten character recognition system is presented here. In this implementation, multiple pre-processing and processing stages have been employed. The complete system is going to be described in two parts, the training section and the testing section. Figure 1 shows how these organisations have been implemented. Once the training samples have been digitised by the scanner during the training period, they go through a set of pre-processing stages. The pre-processings include the standard smoothing, hole-filling and neighbouring filtering. The system then detects some fundamental characteristics found in the Bengali characters and the characters undergo an stage of pre-classification. Details of this pre-classification is presented in later sections of this paper. After pre-recognition, an index for group membership is determined and the characters are separated in different sub-groups. Then some structural information is extracted from the characters belonging to each of the sub-groups and a structural signature corresponding to each of the character is generated. These signatures act as the model of the characters within a sub-group. Based on these extracted information the reference prototype database is organised and the system is now ready to be tested. Figure 1(a) presents the schematic of the training phase.

During the testing phase, each of the test samples are digitised exactly in the same way as is done during the training phase. The same stages of pre-processing, pre-recognition and processing again take place. Some basic structural features are then detected and again a group membership index is calculated. After the structural signature is extracted from the test sample, it is then compared with the prototype signatures stored in the reference database corresponding to the correct sub-group and is assigned to a class whose prototype is closest to the signature of the test sample. Figure 1(b) presents the schematic of the test phase.

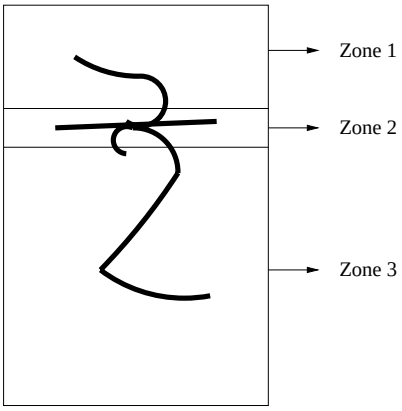

Figure 2: Distinct regions in a Bengali character symbol.

4 Distinct Characteristics in the Symbol Set

The symbols of the Bengali character set have some distinctive characteristics. Detecting these characteristics can help the formation of sub-groups within the whole character set, which can enhance the recognition rate considerably by acting as a pre-classification stage[RF97b]. In order to identify these characteristics, it is helpful to analyse the structural information of Bengali character set. It is possible to detect three distinct regions in the symbols of the Bengali Character set. Figure 2 shows an example of how these zones are identified. Zone 2 is the part of the character which might have a horizontal line segment, which is known as a *matra*. Identifying the *matra* can help simplify the recognition task. Zone 1 is the part that is above the Zone 2 just described and Zone 3 is the part below the Zone 2. Some characters in the alphabet do not have a horizontal line segment or *matra*. But even in those cases, the zonal detection is possible. Based on the characteristics of these three regions, a pre-classification stage has been devised.

5 Pre-classification: Group Formation Based on Structural Similarity

The characters of the Bengali alphabet are divided into certain groups based on some structural similarities. It is noticed that the characters can be easily separated into very clear and distinct

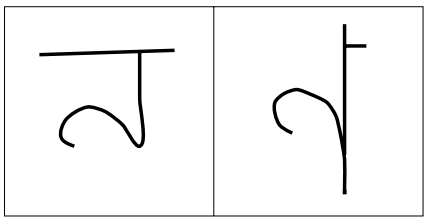

Figure 3: Differentiating between two Bengali character symbols based on *matra*.

sub-groups which allows a hierarchical matching to be performed when searching for the correct classification. A detailed discussion on how these high level features are detected and how they can help group formation can be found in [RKS98b]. The group formation is based on the absence or presence of these basic similarities or based on their combination. The following basic characteristics were identified:

- Presence or absence of a *matra*: In some of the characters of the Bengali alphabet, there is a horizontal line segment in the upper part of the symbol (Zone 2). This is known as the *matra*. This characteristic is present in some symbols, where it is completely absent in others. So detection of these horizontal lines can be a very important tool in reducing the number of possible classes to which a particular symbol might belong. Figure 3 shows how this feature can be used to differentiate between two symbols.
- Presence or absence of an *upper part* found vertically above the identified *matra*: In some symbols, there is a portion of the character extending vertically above the horizontal line segment (Zone 1). If it is possible to detect the presence of a *matra*, then it is straightforward to detect this portion. This portion is also present in other symbols having no *matra*, whereas it is completely absent in others. So detecting these parts can also act as a very efficient tool in pre-classifying the characters. Figure 4 shows how this feature can be used to differentiate between two symbols.
- Presence or absence of multiple *disjoint sections* within the character: Some symbols in the Bengali character set has more than one continuous section within the symbol. In these cases, it should be possible to detect multiple (normally 2) clusters in the character

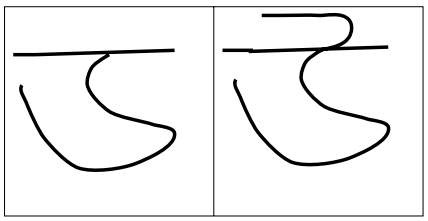

Figure 4: Differentiating between two Bengali character symbols based on *upper part*.

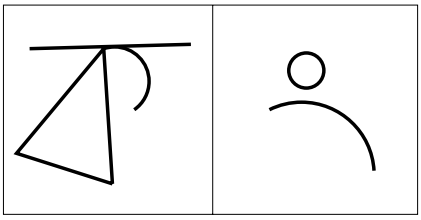

Figure 5: Differentiating between two Bengali character symbols based on *disjoint sections*.

symbol. In case of the other characters, there is only one continuous cluster present within the symbol. This characteristic can act as a major tool in creating sub-groups in an effort to reduce the possible target classes when trying to classify an unknown symbol. Figure 5 shows how this feature can be used to differentiate between two symbols.

Code Name of Criteria	Assigned Value	Interpretation
C	0	There is one disjoint section
C	1	There are two disjoint sections
M	0	No matra is present
M	1	A matra is present
U	0	There is no upper part above matra
U	1	There is an upper part above matra

Table 1: Interpretation of basic characteristics

Depending on these three variables, eight different groups representing the combinations of these variables can be formulated. Of these eight groups, only five groups are found valid in the case of Bengali characters. The other three groups represent combination of these basic characteristics which are not found in the Bengali character set. The first, second and third characteristics are named M, U and C respectively. All of these variables can be assumed to have Boolean characteristics, essentially meaning that any of these three variables can assume the value of either 0 or 1. Table 1 describes how these variables can express the characteristics of any

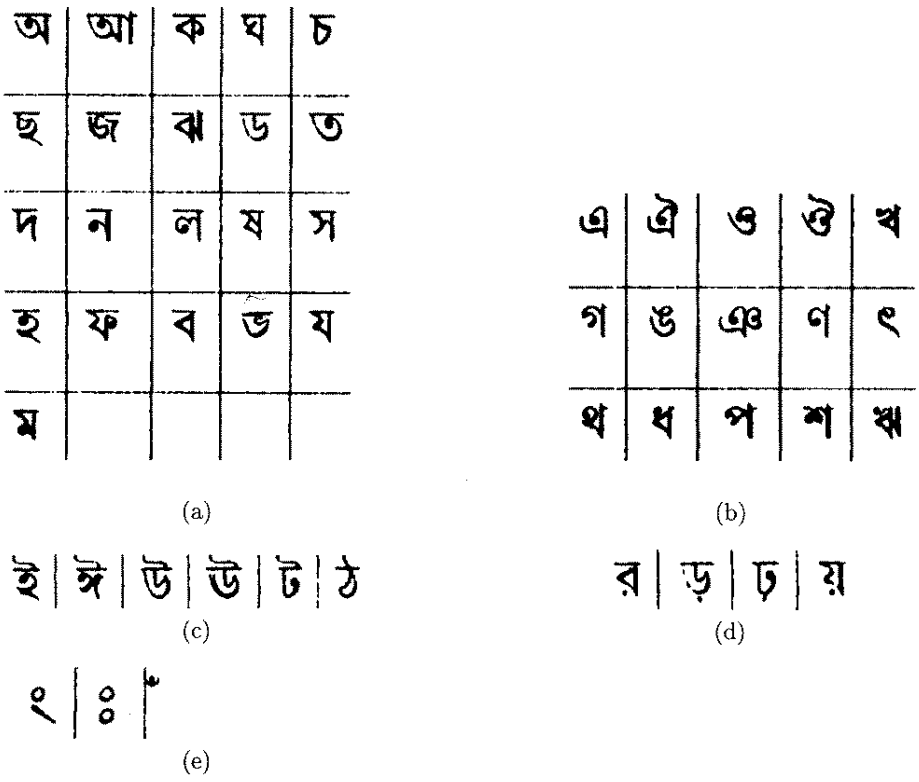

Figure 6: Different Characters belonging to the CMU code of (a) 010, (b) 000, (c) 011, (d) 110, and (e) 100.

CMU Code	Interpretation	Comment
000	There is one disjoint section, matra absent and upper part absent	Valid
001	There is one disjoint section, matra absent and upper part present	Invalid
010	There is one disjoint section, matra is present and upper part absent	Valid
011	There is one disjoint section, matra is present and upper part present	Valid
100	There are two disjoint sections, matra absent and upper part absent	Valid
101	There are two disjoint sections, no matra and upper part above matra present	Invalid
110	There are two disjoint sections, matra present and no upper part above matra	Valid
111	There are two disjoint sections, matra present and upper part above matra is present	Invalid

Table 2: Group formation based on structural criteria

character with the help of these assigned values. If the Boolean values of M, U and C are used for group formation, then a formal coding based on presence of absence of these characteristics can be conveniently called the CMU coding, whereas the value of the CMU can be indicative to the combination of these basic characteristics. For example, if a character has no matra, no upper part and one disjoint sections, then it can be called a character having the CMU code of 000. Table 2 presents the group formation based on the values presented in Table 1. Figures 6 presents the different character symbols which belong to different sub-groups formed based on the CMU codes.

6 Algorithm Development

The successful detection of these primary high-level features can have a profound influence on the ultimate performance enhancement of an OCR. The following sections present the formal algorithms to detect the three basic characteristic features, *matra*, *upper part* and the *disjoint parts*. Discussions on how these algorithms cam be practically implemented are also incorporated within each section.

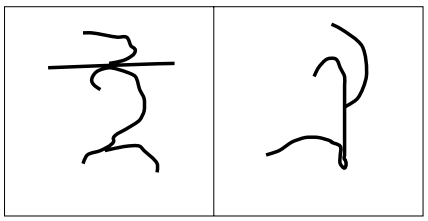

Figure 7: Definitions of an *upper part*: what qualifies and what not.

Matra Detection

The presence of a *matra* is manifested by a horizontal line on the upper part of the character symbol. It is stipulated that the presence of a horizontal or nearly horizontal line with a continuous or almost continuous pixel proximity would be an ideal candidate to be identified as a *matra*. But this is not the only consideration. Depending on the writing style, the position of the matra within the symbol with respect to the base line may vary a lot. It is assumed that to be a candidate for a *matra*, it must be found in the upper portion of the symbol. More specifically, while developing the *matra* detection algorithm, it has been assumed that it should be found within one third of the total height from top most row of pixels containing a valid symbol presence. In the actual implementation, the total number of pixels were calculated and the rows having a valid "ON" pixel were detected. Dividing the total number of pixels present within the image by the total number of rows containing those pixels, the statistical average of the number of pixels per line was calculated. It has been further assumed that the *matra* should contain at least twice the number of valid pixels with respect to the statistical average number of pixels calculated on the whole image. The formal algorithm has been presented in Algorithm 1 and can be found in the Appendix.

Upper Part Identification

The upper part is a portion of the character symbol which is above the detected *matra*. There are symbols in Bengali alphabet which have an equivalent upper part without the present of a *matra*. These symbols are not considered here. For example, the first symbol in Figure 7 is considered to have a valid *upper part*, while the second symbol in the same figure is not. In the definition

of an *upper part*, the presence of a *matra* has been assumed to be mandatory. Algorithm 2 (see Appendix) takes the help of Algorithm 1 to detect the required part in the character symbol. Obviously the relative height of the *matra* will be smaller in cases where there is an *upper part* with respect to those symbols where there is no upper part, as the *matra* is automatically placed a bit lower to accommodate the *upper part*. Bearing this in mind, it was assumed that if the line where a *matra* was identified was below one fourth height of the complete character, then there was no *upper part*. So in order to identify a proper *upper part*, it was required that due to the presence of the *upper part*, the matra was shifted down by more than one fourth of the total height of the character.

Disjoint Part Identification

Another very important characteristic feature present in the Bengali character symbols is the number of disjoint parts. By the phrase *disjoint parts*, it is assumed that these are completely separate segments present within the character symbol. Since the Bengali character symbols are highly cursive, there is a continuity present in those characters which are made up of only one such segment. On the other hand, the characters which are made of completely disjoint parts are normally written maintaining that characteristic. Detection of this feature can help a lot to identify a character as belonging to a particular sub-group of characters and it becomes easier to distinguish it since the number of possible target classes is greatly reduced. In Bengali character symbols the maximum number of completely disjoint parts can be only two. Although it is a matter of detecting one part or two, rather than detecting how many disjoint parts there are in total, the algorithm devised to calculate this can also easily determine exactly how many disjoint sections are there.

Algorithm 3 (see Appendix) presents a method to determine the presence of disjoint parts within a character symbol. The algorithm searches for continuity in the pixel field of the image and determines the number of distinct clusters that are present within that field. These disjoint

parts are characterised by the fact that from any randomly chosen pixel within that part, a path always exists that connects another randomly chosen pixel within the same part. The algorithm also assumes a sequential search from the top of the image.

Algorithm 3 is in essence a connected component labelling algorithm. This is one of the oldest problems faced in image processing and a number of solutions exist for it[DST92, AU87]. This particular algorithm implemented as part of the current character recognition system can very well be replaced by other similar algorithms.

A Note on the Algorithms

In the formal description of the algorithms, absolute values have been used in some cases. These absolute values are related to the image resolution of the test images. Further more, it has been assumed that a sequential processing from top of the image file is carried out. Sometimes three consecutive rows are loaded in memory and the algorithms are applied. This is just to ensure that the algorithms run very fast and the memory use is very low.

7 Performance of the Algorithms

A successful detection of the three main high-level characteristic features can be very helpful in forming suitable criteria for separating the characters in smaller sub-groups, as described in detail in [RKS98a]. The successes of the different algorithms are presented here in light of the complexity of the data and how the algorithms cope with extremely difficult situations. In order to test the effectiveness of the algorithms, a training and testing database has also been compiled.

A Bengali Handwritten Database

A Bengali handwritten database has been compiled. The characters were written separately and in this sense it is a compilation of hand-printed characters. The original dimension of the sample characters were restricted to 7" by 4". The scanned resolution was 400X200 pixels. A GT-4000

Approach	Successful Detection %	False Detection %	Failed to Decide %
Before rotation	77.94	19.20	2.86
After rotation	89.73	9.79	0.48

Table 3: The performance of the *matra* detection algorithm

Seiko Epson Colour Image Flat-bed Scanner was used for scanning. In the image files, an 1 is an absence of characters and a 0 is a presence, so the pixels were in reverse-video mode. But no restrictions were imposed on the writers, so the collected samples represents unconstrained hand-printed Bengali characters. Unfortunately, the size of the database was very small, a total of 521 characters written by 20 different persons were collected from the 49 classes.

Matra Detection algorithm

Table 3 presents the performance of the *matra* detection algorithm. It is clear from the table that it is highly successful in detecting the *matra*. The main sources of error in detecting the *matra* are the facts that some writers did not write the *matra* in its full length, rather produced a shorter one and some writers did not produce a purely horizontal line as is required for the successful detection of the *matra*. The *matra* detection algorithm takes care of the first problem of shorter strokes by introducing the criterion of statistical distribution on a *per line* basis, which proved to be very successful. But obviously there were instances in the samples where the *matra* was not a horizontal line at all, but deviated from the the horizontal line by about $\pm 10^0$. In such cases, a two stage method was employed. In the first stage, the image is rotated by a particular angle and then in the second stage the same *matra* detection algorithm is applied. (Various image rotation techniques can be found in [Bur97].) By searching the angular field of $\pm 10^0$, the possible existence of a *matra* is detected. If no presence can be detected within that angular field, it is assumed that no *matra* is present. Table 3 shows that if the *matra* detection algorithm is applied as part of a two stage method, then the success of detecting the *matra* increases significantly. The final column in Table 3 shows the percentage of the cases where the algorithm failed to arrive at a decision. It is interesting to note that these cases were reduced drastically when a rotational pre-detection stage was incorporated.

Approach	Successful Detection %	False Detection %	Failed to Decide %
Before rotation	72.84	25.41	1.75
After rotation	79.93	19.12	0.95

Table 4: The performance of the *upper part* detection algorithm

Successful Detection %	False Detection %	Failed to Decide %
96.27	2.84	0.89

Table 5: The performance of the *disjoint part* detection algorithm

Upper Part Detection algorithm

Table 4 presents the performance of the *upper part* detection algorithm. It is clear from the table that it is highly successful in detecting the *upper parts*. In handwritten characters, the success of finding the *upper parts* actually depends on the success of finding the *matra*. So the success is related to how well the *matra* detecting algorithm can perform. As is obvious from Section 7, the success of the *matra* detection algorithm depends on finding the correct rotated position of the character image. Table 4 demonstrates how the success of finding *upper parts* is increased with better *matra* finding approach.

Disjoint Part Detection algorithm

Detecting *disjoint parts* is by far the most difficult task, specially in the case of handwritten characters where the cursiveness of the characters makes the problem more complicated. A novel algorithm for finding *disjoint parts* in an image file has been presented in this paper. The algorithm is very successful in detecting *disjoint parts*, as is evident from Table 5. Actually, it only failed in those cases where the samples characters were not written properly. There were errors due to two deficiencies in the writing style:

- When a disjoint character was written in a way so that there was no real physical separation between the two supposedly separate parts because of the cursiveness.
- When a normally continuous character was written in a way so that there was an artificial separation between two or more parts of the same character, primarily introduced because

of abnormal pen-up/pen-down approaches.

8 Classification Method

As there is a large number of symbols present in the Bengali alphabet, the cost associated with implementing a recognition method is enormous. Since the aim of present work is to conduct an investigation into the difficult task of developing a practical Bengali character recognition system, rather that implementing an immediate working version, a simplistic classification method was employed. The following sections present the issues concerning the low-level feature extraction and classification in terms of a complete Bengali character recognition system.

Row and Column Code Generation

It is noted in the case of handwritten Bengali characters that the shape of the characters are highly cursive. There are, in general, no identifiable strokes or sequence of strokes and it is very difficult to adopt a feature extraction process that is able to represent the characteristics of these characters within a smaller feature size. The presence of a very large number of closely related symbols complicates the problem even further. Therefore a very simplistic approach has been adopted here to capture the essence of the different characters. The characters were compressed using a simple continuity criterion. The basic idea was to calculate, for each row and column, how many times an imaginary line will be intersected by the presence of the character. Whether the handwritten character is shifted, transformed in a linear direction, elongated, distorted or size transformed, the relative frequency of occurrence of the presence of stroke-parts should be more or less same. If the presence or absence of parts of a character is stored row-wise and column-wise, a fairly accurate structural description of the character can be generated. A direct consequence of this compression is that the size of the features thus calculated can be less than or equal to $m+n$ instead of the size mXn, which is the normal dimension of the character. It also has the property of invariance with respect to size transformation, stress transformation and to the rotational variance within a specific range. The row-wise and column-wise compression

algorithm is presented in Algorithm 4 (see Appendix).

Refined Row and Column Code Generation

The row-column code that is generated by Algorithm 4 is very concise and descriptive. Unfortunately there is a lot of redundant information present in these codes. It is easily possible to refine the code generation procedure which allows the generation of much smaller code of length r, where $r \ll m+n$. It is observed that the same frequency of occurrence is repeated many times in a row or column code generated from the raw character images. The number of recurrence, although dependent on the resolution of the sample character, do not provide any more information than that already provided by the first occurrence. That is why the recurrence of a particular frequency is redundant as far as the structural information is concerned. A refined row and column code is therefore generated which searches through the row and column code expressions and retains only one occurrence of a particular pattern. Algorithm 5 (see Appendix) ensures that in the refined row and column code, the redundant information is suppressed in the horizontal and vertical direction respectively.

Cleaned Refined Row and Column Code Generation

The refined row and column codes can be further compressed based on some observations on the actual strings generated on real samples. The following observations were made on these generated refined row and column codes:

- It is observed that sometimes the row and column codes start with a 0, indicating that there is a region of no presence before the character is actually mapped. These 0's are redundant as they carry no information. Obviously, the need to carry out this adjustment would not arise if the samples are size-normalised in the raw form in a way so that a window is isolated that only contains the actual character. Since this information is automatically generated by the row and column codes, and the generated code is invariant to size anyway and the size-normalisation was unnecessary.

Number of Classes	Pre-recognition	Average Recognition rate
49	Absent	50.14
49	Present	66.00

Table 6: The recognition performance with and without the pre-recognition phase

- Discarding all the 0's in the code is a tempting thought, but there can be a region of genuine no presence in some Bengali characters, so this is not feasible. The presence of one or more 0 in the code actually points to a character with a possible multiple separate parts.
- Due to the fact that we are dealing with highly variable handwritten characters, the presence of horizontal and vertical lines are not really very reliable. Rather, they are best described as a staircase structure. In such cases, there can be a single occurrence of a portion of a character which is a wrongly recorded presence. Judging from the resolution of the figure, it was estimated that a particular occurrence that has occurred only once in the generated code must be a misinformation and should be discarded.

Algorithm 6 (see Appendix) presents an approach that produces a further compressed row and column code. This new code encompasses all the relevant information and is optimised in the sense that the relevant information is stored within the smallest code size. It is to be noted here the character representation technique illustrated in this section has some relation to a well-established method known as "characteristic loci"[Glu71]. The present formulation, although developed on that theme, has extended the character representation in a compressed form to a much more sophisticated level and does represent an essentially new technique.

9 Performance of the OCR

From the small number of character samples collected for each class, it is evident that no statistical classification system will be able to generalise the solution of recognising a 49 class problem. That is why syntactic methods were employed to generate character signatures from the samples. Table 6 presents the results of classification with and without using the pre-recognition phase. It is seen that just over 50% of the samples are recognised by the method without a

Number of Classes	Pre-recognition	Average Recognition rate	
		Top 2	Top 3
49	Absent	54.39	59.55
49	Present	71.00	75.34

Table 7: The recognition performance with multiple top choice

pre-recognition phase and on the other hand, over 66% of the samples are correctly recognised when a pre-recognition phase is introduced. The results presented so far assumed a top class matching technique, so that a character is assigned to a class, whose prototype is closest to its own signature. Foe various reasons, the top class choice often fails to produce the best class. But if the top 2 and 3 classes are now considered, it is seen that a huge improvement in the recognition rate is achieved. Table 7 presents these results.

It is noted that the size of the sample characters was 7"X4". The handwritten samples were isolated characters collected from various sources. It is true that a huge scanning resolution was employed to digitise the samples, but this was required because of the large sizes of the original samples. It is also true that a relatively small dataset has been used for this investigation. It is also noted that there exits no standard collection of handwritten Bengali characters in digitised form as is readily available for Roman or Chinese characters. The current investigation, although reporting an evaluation of a complete OCR for handwritten Bengali characters, does not claim to be a candidate to be directly incorporated in practical implementations of industry standard OCR engine.

It is also imperative to elaborate on the training and testing of the proposed OCR. Due to the fact that a small number of samples were made available for training and testing, the Leave-One-Out Estimator or the U Method was applied to evaluate the overall average performance of the OCR. This method can be described as below:

- Take one pattern sample, say X_i, θ_i out of X, θ.
- Train the classifier using $(X, \theta)_i$, where, $(X, \theta)_i = (X_1, \theta_1; ...; X_{i-1}, \theta_{i-1}; X_{i+1}, \theta_{i+1};$

Case Number	Actual Class	Collision Class
1		
2	ঙ	ঙ
3	ভ	ঙ
4	ক্ষ	০
5	ঞ	ন

Figure 8: The most frequent confusion cases

Considered Feature	% Division of the Alphabet	
	Feature Present	Feature Absent
Matra	63.26(31/49)	36.74(18/49)
Upper Part	12.24(6/49)	87.76(43/49)
Disjoint Parts	14.29(7/49)	85.71(42/49)

Table 8: The statistical separation of the character set based on one feature only

$...; X_N, \theta_N)$.

- Test the classifier on X_i, θ_i. If X_i is classified correctly, set $e_i = 0$, otherwise set $e_i = 1$.
- Repeat steps 1-3 for $i = 1, 2, 3, ..., N$ and set e_i, for $i = 1, 2, 3, ..., N$.
- Compute $\hat{P}_e(U)$ as:

$$\hat{P}_e(U) = (1/N) \sum_{i=1}^{N} e_i$$

This method has been found experimentally to be approximately unbiased (Toussaint[Tou74]).

A major source of error was the confusion cases. These are the mutual mis-recognitions of very similar classes with each other. Figure 8 shows some of the most common confusion cases. As the recognition structure is a serial structure[RF97d], any character wrongly classified by the pre-recognition stage has a reduced chance of being correctly recognised by the subsequent recognition phase. That is why some of these mis-recognitions are directly attributed to the failure of the pre-classification stage. But on the other hand, the pre-classifier stage drastically reduces the number of possible choices, and the chance of a character correctly placed in the correct sub-group is greatly increased.

Considered Feature Pairs	% Division of the Alphabet	
	Feature Combination Present	Feature Combination Absent
Matra-Upper Parts	12.24(6/49)	87.76(43/49)
Upper Parts-Disjoint Parts	0.00(0/49)	100.00(49/49)
Disjoint Parts-Matra	8.16(4/49)	91.84(45/49)

Table 9: The statistical separation of the character set based on one pair of features only

10 Discussion

In recent years, there has been a renewed attempt to reformat the classification approaches to the recognition of difficult character sets, specially handwritten ones[RF97c, RF97e, RF97a, HS93, AS88]. It has been found that successful separation of the test samples at an early stage can enhance the recognition performance of an existing OCR[FR97, CAS95, KRRG93, RF97b]. It has also been realised that a multiple stage multiple expert character recognition scheme has the potential of outperforming individual stand-alone classifiers because of its ability to handle extreme variance in the training and testing samples[RF97d, FR96, SGL94, HHS92]. The present study reported in this paper follows this recent trend in building a multiple stage hierarchical character recognition configuration. The identification of the high level features, as described in this paper, helps to build an early pre-recognition stage in an existing OCR, presented in [RKS98a]. An analysis of the Bengali character symbols reveals that it is possible to build smaller sub-groups based on the detection of these high-level features, which ultimately helps in producing a better overall recognition process. Table 8 shows how the character set can be sub-divided based on a single feature only. Analysing the contents of this table demonstrates that successful detection of a feature like the *upper part* can help create as small a sub-set as only containing just over 12% of the total classes. This can help immensely in picking up the really easy classes much earlier and it would also be possible for them to be removed so that the subsequent search becomes relatively easier for the samples belonging to the other classes. The same observation is true for the *disjoint part* feature. On the other hand, if more that one feature is considered, then it is possible to create a pair-wise separation scheme to divide the character set. Table 9 shows how this results in formation of extremely smaller sub-groups which

Presence/Absence of Features			% Division of the Alphabet	
Matra	*Upper Part*	*Disjoint Parts*	Feature Combination Present	Feature Combination Absent
Absent	Absent	Absent	30.61(15/49)	69.39(34/49)
Absent	Absent	Present	6.12(3/49)	93.88(46/49)
Absent	Present	Absent	0.00(0/49)	100.00(49/49)
Absent	Present	Present	0.00(0/49)	100.00(49/49)
Present	Absent	Absent	42.86(21/49)	57.14(28/49)
Present	Absent	Present	8.16(4/49)	91.84(45/49)
Present	Present	Absent	12.24(6/49)	87.76(43/49)
Present	Present	Present	0.00(0/49)	100.00(49/49)

Table 10: The statistical separation of the character set based on the combination of all three features

is very important in isolating some classes very early in a hierarchical classification configuration.

A very interesting information can also be derived from Table 9. It is seen that there is no valid character satisfying the criteria of the second row in that table, which can be interpreted as understanding that there is no symbol in Bengali character set which has both the feature of a *upper part* and *disjoint part*. So employment of this criterion does not help in divining the database. Similar results can be achieved if all three features are grouped together and different combinations of their presence and absence are considered. Table 10 shows how this can be exploited further to generate multiple sub-groups within the dataset. It show that five different sub-groups can easily be formed based on this information. Studying Table 10 clearly shows that five effective sub-groups can now be formulated very easily. The other three combinations have proved to be invalid, as no symbol within the Bengali alphabet belongs to those categories. This sub-grouping has now been incorporated in designing an improved Bengali OCR for handwritten characters[RKS98a].

The observations from Table 7 implies that a multiple stage multiple classifier configuration will be very effective in recognising such a complicated and large handwritten character set. It has recently been shown[FR97, RF97c] that a multiple expert approach can go a long way in achieving a very robust recognition system, especially in the case of handwritten characters. It

is hoped that adoption of such schemes will help enhance the recognition rate of this Bengali character recognition system. Another important source of errors was the confusion cases apparent between closely resembling characters. It has been shown that an efficient way of solving these frequently arising confusion cases can help boost recognition performance in an existing system[RF97f, FR96, RF96]. Adoption of such systems in the case of the Bengali character recognition system can also result in improvements.

It is noted, however, that although a multiple expert approach is likely to enhance the ultimate recognition rates, it is the *matra*, *cluster* and *upper part* detection algorithms that need special care. Work is underway to devise more sophisticated and robust algorithms for the detection of these characteristic features, as the final recognition depends on the success of these algorithms significantly. Future research is directed in developing a representative database of handwritten Bengali characters, formulating more efficient character representation techniques and more effective multi-prototyping in successfully recognising this character set.

11 Conclusion

A novel hybrid recognition scheme for recognition of handwritten Bengali characters have been presented. It has been shown that a simplistic approach can be modified to develop a very efficient recognition system with the help of multiple stage recognition. An efficient pre-recognition scheme has helped to boost the final recognition performance. An investigation for detecting some basic high level features from the Bengali character set is presented. It has been demonstrated that detection of these features can help form small sub-groups of characters and can act as a pre-classification stage if this information is properly utilised. Extracting high level information in the form of *a priori* knowledge is now considered to be a very important aspect of practical character recogniser design. It is hoped that successful application of the information extracted from the database in the form of high level feature detection will help in future recogniser design especially in the case of handwritten Bengali character recognition. The results obtained should

be considered to be indicative rather than conclusive because of the very small size of the character database. When tested on the train dataset, the system produces a 100% recognition rate, but as completely unseen samples are tested, the recognition was up to 75% when considering the top three choices, which is a very encouraging results for this very complicated task. Discussions about the possible improvement of the system in future have also been incorporated.

Appendix

Algorithm 1 *Procedure Find_matra(Rownum, Colnum, up, down, matra_row)*

BeginProcedure

A is an array containing the bits of one row at a time in the image file, up and down are the upper and lower boundaries(rows) of the image. Rownum and Colnum are the maximum number of rows and columns in the image and matra_row is the row number where a matra is detected.

```
up ← 400
down ← -1
present ← 0
matra ← false
for i = 0 to Rownum do
        j = 0
        while(j ≤ Colnum) do
                if (A(i,j) == 0) then
                        present ← present + 1
                        if(j < up) then
                                up ← j
                        elseif(j > down) then
                                down ← j
                        endif
```

endif

j ← j + 1

endwhile

enddo

row_avg ← present / (down - up + 1)

for i = up to (up + (down - up + 1)/3) do

total ← 0

for j = 0 to Colnum do

if (A(i,j) == 0) then

total ← total + 1

endif

endfor

if (total > 3(row_avg)) then*

matra ← true

matra_row ← i

break

endif

if (matra) then

break

endif

endfor

endprocedure.

Algorithm 2 *Procedure Find_upper_part(Rownum, Colnum, up, down, matra_row, upper_part)*

BeginProcedure

A is an array containing the bits of one row at a time in the image file, up and down are the

upper and lower boundaries(rows) of the image. Rownum and Colnum are the maximum number of rows and columns in the image and matra_row is the row number where a matra is detected.

Find_matra(Rownum, Colnum, up, down, matra_row)

if(4(up + matra_row) ≥ (down - up + 1)) then*

upper_part ← 1

else

upper_part ← 0

endif

endprocedure.

Algorithm 3 *Procedure Find_disjoint_parts(Rownum, Colnum, no_of_parts)*

BeginProcedure

A is an array containing the bits of one row at a time in the image file. Rownum and Colnum are the maximum number of rows and columns in the image and no_of_parts is the number of disjoint parts. ncl is the value assigned to each occurrence of a new part and cln is the revised value assigned to the parts.

i ← 1

ncl ← 4

read row 0

read row 1

do while (i < Rownum - 1)

read row i + 1 into A

for j = 0 to Colnum - 1 do

if (A(i,j) = 1) then

call Searchn(i, j, A, large)

if(large < 100) then

```
                                    A(i, j) ← large
                            else
                                    ncl ← ncl + 1
                                    A(i, j) ← ncl
                            endif
                    endif
            endfor
            cln ← 5
            do while (cln ≤ ncl)
                    changed ← false
                    for m = 0 to 2 do
                            for j = 0 to Colnum - 1 do
                                    if(A(m, j) == cln)) then
                                            Call Change(m, j, cln, changed)
                                            neg ← neg + 1
                                    endif
                            endfor
                    endfor
                    if .not. changed then
                            cln ← cln + 1
                    endif
            endwhile
            Transfer_row(1, 0)
            Transfer_row(2, 1)
            i ← i + 1
endwhile
no_of_parts ← ncl - 4 - neg
```

```
endprocedure


Procedure Searchn(i, j, A, large)
BeginProcedure
        large ← 100
        for l1 = -1 to 1 do
                for l2 = -1 to 1 do
                        i1 ← i1 + l1
                        j1 ← j1 + l2
                        if (A(i1, j1) > 1) then
                                if (A(i1, j1) < large) then
                                        large ← A(i1, j1)
                                endif
                        endif
                endfor
        endfor
endprocedure


Procedure Change(i, j, cln, changed)
BeginProcedure
        for l1 = -1 to 1 do
                for l2 = -1 to 1 do
                        i1 ← i1 + l1
                        j1 ← j1 + l2
                        if (A(i1, j1) > cln) then
                                A(i1, j1) ← cln
                                changed ← true
```

```
                    endif
                endfor
            endfor
endprocedure
```

Algorithm 4 *Procedure Row_column_code(A, N, Row, Col, RC)*

BeginProcedure

A is an array containing the bits of the rows in the image file, Row and Col are the maximum number of rows and columns in the image and RC is the row or column code that is to be generated.

```
for i = 0 to Row(Col)-1 do
        max = 0
        read Row(Col)-1
        run ←false
        for j = 0 to Col(Row)-1 do
            if(A(i,j) == 0) then
                if(run == false) then
                    run = true
                    max = max + 1
                endif
            else
                run = false
            endif
        endfor
        RC(i) = max
endfor
```

endprocedure.

Algorithm 5 *Procedure Refined_row_column_code(RC, max1, max2, RCRefined)*

BeginProcedure

RC is the row or column code, RCRefined is the refined row or column code that is to be generated. max2 is the length of the original row or column code RC.

```
max1 ← 0
test ← RC(0)
RCRefined(max1) ← RC(0)
for i = 1 to max2-1 do
        if(RC(i) ≠ test) then
                max1 = max1 + 1
                RCRefined(max1) = RC(i)
                test = RC(i)
        endif
endfor
```

endprocedure.

Algorithm 6 *Procedure Cleaned_refined_row_column_code(Row, Col, RC, RCCleaned, RowCodeColCode, left, right, max)*

BeginProcedure

RC is the row or column code, RCCleaned is the cleaned row or column code, left and right indicate the leftmost and rightmost nonzero value in a row or column code.

```
left ← 0
do while (RC(left) == 0)
        left ← left + 1
```

```
        enddo
        right ← max - 1
        do while (RC(right) == 0)
                right ← right - 1
        enddo
        j ← 0
        for i = left to right do
                RCCleaned(j) ← RC(i)
                j = j + 1
        endfor
        j ← 0
        k ← 0
        for i = 1 to RowCol - 2
                if((RowCodeColCode(i-1) ≠ RowCodeColCode(i))
                .and. (RowCodeColCode(i) ≠ RowCodeColCode(i+1)) then
                        k ← k + 1
                endif
                RCCleaned(j) ← RCCleaned(k)
                test = RowCodeColCode(i)
                i = i + 1
                while(RowCodeColCode(i) == test) do
                        i = i + 1
                endwhile
                j = j + 1
                k = k + 1
        endfor
endprocedure.
```

References

[AS88] P. Ahmed and C. Y. Suen. A decision-making method and its application in unconstrained handwritten character-recognition. In *Proc. 1st Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-88), Vols. 1 and 2*, pages 638–644, Baltimore, MD, USA, 1988.

[AU87] P. Shankar A. Unnikrishnan, Y. V. Venkatesh. Connected component labelling using quadtrees - A bottom-up approach. *The Computer Journal*, 30(2):176–182, 1987.

[Bur97] H. E. Burdick. *Digital Imaging: Theory and Applications.* McGraw-Hill, New York, Lisbon, Mexico City, Sydney, 1997.

[CAS95] J. Cao, M. Ahmadi, and M. Shridhar. Recognition of handwritten numerals with multiple feature and multistage classifier. *Pattern Recognition*, 28(2):153–160, 1995.

[CCS84] R. Chandrasekaran, M. Chandrasekaran, and G. Siromoney. Computer recognition of Tamil, Malayalam and Devanagari characters. *J. Inst. Elec. Telecom. Engg.*, 30:150–154, 1984. India.

[CK80] P. Chinnuswamy and S. G. Krishnamurthy. Recognition of handprinted Tamil characters. *Pattern Recognition*, 12:141–152, 1980.

[CP97] B. B. Chaudhuri and U. Pal. An OCR system to read two Indian language scripts: Bangla and Devanagari. In *Proc. 4th Int. Conference on Document Analysis and Recognition, ICDAR97*, volume 2, pages 1011–1015, Ulm, Germany, 1997.

[DC93] A. K. Dutta and B. B. Chaudhuri. Bengali alpha-numeric character recognition using curvature features. *Pattern Recognition*, 26:1757–1770, 1993.

[DST92] M. B. Dillencourt, H. Samet, and M. Tamminen. A general approach to connected-component labelling for arbitrary image representations. *JACM*, 39(2):253–280, 1992.

[FR96] M. C. Fairhurst and A. F. R. Rahman. A new multi-expert architecture for high performance object recognition. In *Proc. Int. Sym. on Intelligent Systems and Advanced Manufacturing, Machine Vision Applications, Architectures, and Systems Integration V, SPIE 2908*, pages 140–151, 1996.

[FR97] M. C. Fairhurst and A. F. R. Rahman. A generalised approach to the recognition of structurally similar handwritten characters. *IEE Proc. on Vision, Image and Signal Processing*, 144(1):15–22, 1997.

[Glu71] H. A. Gluskman. Multicategory classification of patterns represented by high-order vectors of multilevel measurements. *IEEE Trans. Comput.*, 20:1593–1598, 1971.

[HHS92] K. H. Ho, J. J. Hull, and S. N. Srihari. *Combination of Decisions by Multiple Classifiers in Structured Document Image Analysis*, pages 188–202. S-V, 1992. H. S. Baird, H. Bunke and K. Yamamoto(Eds.).

[HS93] Y. S. Huang and C. Y. Suen. The behavior-knowledge space method for combination of multiple classifiers. In *Proc. IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition (CVPR 93)*, pages 347–352, New York, USA, 1993.

[KRRG93] Zs. M. Kovács, R. Ragazzoni, R. Rovatti, and R. Guerrieri. Improved handwritten character recognition using second-order information from training set. *Electronics Letters*, 29(14):1308–1310, 1993.

[Mia92] M. A. K. Mia. Recognition of printed Bangla characters by syntactic method of pattern recognition. Master's thesis, Dept. of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh, 1992.

[PC95] U. Pal and B. B. Chaudhuri. Computer recognition of printed Bangla script. *International Journal of Systems Science*, 26(11):2107–2123, 1995.

[RC84] K. Ray and B. Chatterjee. Design of a nearest neighbour classifier system for Bengali character recognition. *J. Inst. Elec. Telecom. Engg.*, 30:226–229, 1984. India.

[RF96] A. F. R. Rahman and M. C. Fairhurst. A new approach to handwritten character recognition using multiple experts. In *Proc. 5th Int. Workshop on Frontiers of Handwriting Recognition*, pages 283–286, University of Essex, UK, 1996.

[RF97a] A. F. R. Rahman and M. C. Fairhurst. A comparative study of decision combination strategies for a novel multiple-expert classifier. In *Proc. 6th Int. Conference on Image Processing and Its Applications*, volume 1, pages 131–135, Dublin, Ireland, 1997.

[RF97b] A. F. R. Rahman and M. C. Fairhurst. Exploiting second order information to design a novel multiple expert decision combination platform for pattern classification. *Electronics Letters*, 33(6):476–477, 1997.

[RF97c] A. F. R. Rahman and M. C. Fairhurst. A new hybrid approach in combining multiple experts to recognise handwritten numerals. *Pattern Recognition Letters*, 18(8):781–790, August 1997.

[RF97d] A. F. R. Rahman and M. C. Fairhurst. Introducing new multiple expert decision combination topologies: A case study using recognition of handwritten characters. In *Proc. 4th Int. Conference on Document Analysis and Recognition, ICDAR97*, volume 2, pages 886–891, Ulm, Germany, 1997.

[RF97e] A. F. R. Rahman and M. C. Fairhurst. Multi-prototype classification: Improved modelling of the variability of handwritten data using statistical clustering algorithms. *Electronics Letters*, 33(14):1208–1209, 1997.

[RF97f] A. F. R. Rahman and M. C. Fairhurst. Selective partition algorithm for finding regions of maximum pairwise dissimilarity among statistical class models. *Pattern Recognition Letters*, 18(7):605–611, July 1997.

[RKS98a] A. F. R. Rahman, M. Kaykobad, and M. A. Sattar. A novel hybrid approach to handwritten Bengali character recognition. In *Proc. Int. Conf. on Computational Linguistics, Speech and Document Processing, February 18-20, pages A5-A10*, Calcutta, India, 1998.

[RKS98b] A. F. R. Rahman, M. Kaykobad, and M. A. Sattar. Detection of some characteristic features in handwritten Bengali characters: An approach to enhance recognition performance of a Bengali OCR. *Pattern Recognition Letters*, 1998. Submitted.

[Sat90] M. A. Sattar. Recognition of printed Bengali characters by decision theoretic method of pattern recognition. Master's thesis, Dept. of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh, 1990.

[SC76] K. Sethi and B. Chatterjee. Machine recognition of handprinted Devanagari numerals. *J. Inst. Elec. Telecom. Engg.*, 22:532–535, 1976. India.

[Set77] K. Sethi. Machine recognition of constrained handprinted Devanagari. *Pattern Recognition*, 9:69–75, 1977. India.

[Set87] K. Sethi. Role of context in Devanagari script recognition. *J. Inst. Elec. Telecom. Engg.*, 33:86–91, 1987. India.

[SGL94] C. Y. Suen, J. Guo, and Z. C. Li. Analysis and recognition of alphanumeric handprints by parts. *IEEE Trans. on Systems Man and Cybernetics*, 24(4):614–631, 1994.

[SR89] M. A. Sattar and S. M. Rahman. An experimental investigation on Bangla character recognition system. *Bangladesh Computer Society Journal*, 4(1):1–4, 1989. Bangladesh.

[Tou74] G. T. Toussaint. Bibliography on estimation of misclassification. *IEEE Trans. Information Theory*, 20(4):472–479, 1974.