5th Copy

ON THE DEVELOPMENT OF AN OPTICAL CHARACTER RECOGNITION (OCR) SYSTEM FOR PRINTED BANGLA SCRIPT

UMAPADA PAL

Computer Vision and Pattern Recognition Unit
Indian Statistical Institute
203 B. T. Road
Calcutta 700035
INDIA

A thesis submitted to the ***Indian Statistical Institute***
in partial fulfilment of the requirements
for the degree of
DOCTOR OF PHILOSOPHY

Acknowledgments

At the very beginning, I would like to thank my supervisor Prof. B. B. Chaudhuri for introducing me to the topic of OCR, his advice, guidance and his insistence on being meticulous in different aspects. This being a sort of pioneering work in the field of OCR on Bangla, naturally, I could not get many authorities to fall back upon. Prof. Chaudhuri filled the void to the best of my advantage, advising, discussing, providing counter-examples to my earlier and less mature hypothesis. His constant encouragement helped me in making research objectives clear and keeping myself motivated throughout the years of my tenure as a research fellow. His ideas in the subject left a long lasting influence on my own thinking. He was always alert regarding the quality of our publication.

Prof. D. Dutta Majumder, Emeritus Professor, allowed me to use the resources in the Knowledge Based Computer Systems (KBCS) during the early years of my research.

I was benefited from the knowledge, help, friendship and encouragements of many people in the course of my work. Prof. S. K. Parui deserve special thanks for his encouragement and advice in the research work. I thank Dr. Dipti Prasad Mukherjee, Dr. Probal Sengupta, Tamaltaru Pal, Ujjwal Bhattacharya, Pulak Kundu, Anirban Roy Chaudhuri, Umesh P. S. Adiga, Niladri Sekhar Das and Anil Kumar Shukla for all the help they have extended.

I wish to thank Council of Scientific and Industrial Research (CSIR), India for financial support throughout my tenure, and Photo Type Setting department of Saraswaty Press Ltd. for supplying different kinds of text for digitization.

I express gratitude towards every member of our family - my mother, uncle, aunt, brothers and others for their encouragements. My father left this world and with due respect, I dedicate this thesis towards the memory of my late father Suphal Chandra Pal.

June 30, 1997

Umapada Pal

UMAPADA PAL
CVPR Unit
ISI, Calcutta

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

This thesis concerns OCR development of machine printed text in an Indian language, *Bangla* (Bengali) which is the fourthmost popular language in the world and the secondmost popular language in India.

1.1 Optical Character Recognition

Optical Character Recognition (OCR) is a process of automatic computer recognition of characters in optically scanned and digitized pages of text. OCR is one of the most fascinating and challenging areas of pattern recognition with various practical applications. It can contribute tremendously to the advancement of an automation process and can improve the interface between man and machine in many applications, including office automation and data entry. The input of an OCR system consists of text on paper. The output is a coded file with an ASCII (American Standard Code for Information Interchange) or other character code representation as well as special symbols for unrecognized or doubtfully recognized patterns. The output can be reformatted for input to a word processing or page layout systems, or used directly in information retrieval, image filling, remittance

processing, document sorting, or other downstream applications.

Some practical application potentials of OCR system are as follows:

- Reading aid for the blind (OCR + speech synthesis).
- Automatic text entry into the computer, desktop publication, library cataloging, ledgering etc.
- Automatic reading for sorting of postal mail, bank cheques and other documents.
- Document data compression: from document image to ASCII format.
- Language processing.
- Multi-media system design.

There are several factors which can create difficulties in the development of OCR system. Some of those pertaining to machine printed documents are listed below.

- **Shape similarity :** Different characters can have similar shapes. Some examples of similar shaped characters in Bangla are GH(ঘ) and J,(য), KH(খ) and TH(থ), B(ব) and R(র), U(উ) and U, (ঊ) etc. (here GH; J,; KH etc. are ASCII codes of Bangla characters used for representing them in the computer. In Chapter 2 a complete set of Bangla characters with their ASCII code are shown). Because of the shape similarity one character may be recognized as another character. Also, due to font, size and style variations, a single character may have different shapes, which also contribute to recognition problem.
- **Deformation of image :** Noise is the main cause of image deformation. Isolated dots, disconnected line segments, holes and breaks in lines are caused

due to noise. Noise may be caused due to improper scanning and poor paper quality. Photocopied documents may also have some noises.

- **Document skew :** There may be natural skew or skew imposed during scanning of the document that may create difficulty in recognition.

- **Effect of resolution:** If a document is digitized at inappropriate Dots Per Inch (DPI), the character recognition rate may suffer.

- **Presence of graphics, tables, halftone pictures and mathematical symbols :** A document may contain graphics, tables, halftone pictures, mathematical symbols etc. which should be separated from the basic text materials before recognition. Some of the materials like tables, mathematical and chemical expressions may need special treatment for recognition.

- **Complex document structure :** A document page may have a complex physical layout, which should be segmented before subjecting each segment to recognition.

- **Complex background texture and colour :** The background over which the text is printed may not have uniform white texture. For example, a currency note has a complex and coloured background on which textual matter is printed.

- **Script complexity :** Complexity of the alphabet and writing rules may accelerate the OCR error rate. For example, there are 11 vowels, 40 consonants and 10 numerals in the Bangla alphabet. Vowels take modified (allographic) shapes in connection with consonants except at the beginning of the words and when two vowels sit side by side in the words. In addition, two, three or four characters together can generate a new complex shape called a *compound (cluster) character* (see chapter 2). These factors add to the OCR error.

Handwritten characters pose more problems for recognition. However, our

concern is machine-printed text recognition and hence we omit discussion about hand-written character recognition problems.

1.2 Character recognition approaches

The class of techniques applied to character recognition is not, in principle, different from that applied in any general image pattern recognition problem. However, depending on the features used, the character recognition techniques can be broadly classified as follows:

1. Template matching and correlation techniques.
2. Feature analysis and matching techniques.

Template matching and correlation techniques :

In this class of techniques an input character is compared pixel by pixel with previously stored character prototypes. The stored prototypes are known as *templates* and the basic operation of character comparison is called *template matching* [165]. This technique can be divided into two parts: (i) superimposing an input shape on templates and (ii) measuring the degree of coincidence between the input shape and the templates [102]. The prototype that matches most closely with the input character provides recognition. A template matching technique is slow and costly because it requires proper registration of the input character in terms of location, size and orientation. Also, such a technique suffers from sensitivity to noise and is not automatically adaptive to font, style and size variations.

Feature analysis and matching techniques : These are among the most frequently used techniques for character recognition. Here, significant features are extracted from the input character and compared with the feature descriptions of

the ideal characters. The description that matches most closely provides recognition. Many feature analysis techniques have been developed and applied for character recognition. Most of them are examples of traditional pattern recognition methods, and are usually suitable for application to constrained domains. Suen *et al.* [158] have presented a very useful survey of various feature matching techniques.

The features used in various approaches of OCR can be grouped as follows:

1 **Global transformation and series expansion :** Global transformation and series expansion techniques help to reduce dimensionality of the feature vector and provide features invariant to some global transformations like translation and rotation. Researchers have used Fourier [143], Walsh [70], Haar [177], Hadamard series expansions [178], Karhunen Loeve expansion [85], chain-code transform [30] etc. for the purpose. The feature extraction and mask making processes are easy but time consuming.

2 **Geometrical and topological features :** These features are most popular among the researchers. Global and local properties of the characters may belong to these features. They include strokes and bays in various directions, end points, intersection of line segments, loops and their positions in the character bounding box, stroke relations, angular properties, sharp protrusion etc. [174]. Compared to other techniques, the main advantage of taking geometrical and topological features is their high tolerances to distortion and style variations, and moderate tolerances to rotation and translation as well as relatively high speed of computation.

3 **Distribution of points :** This category comprises features extracted from statistical distribution of points. Moments, crossing count, zoning, n-tuples, characteristic loci etc. belong to this category of features [15,20,167]. Implementation complexity of these features is very low and hence the computation speed is high. These features are tolerant to distortion and take care of style variations to some extent.

1.3 Historical background and present status of OCR

The origin of character recognition [97] can be found in 1870 when Carey invented the retina scanner - an image transmission system using a mosaic of photocells. Later, in 1890, Nipkow invented the sequential scanner, which is a major breakthrough both for modern television and reading machines. However, character recognition was initially considered as an aid to the visually handicapped and the early successful attempts were made by the Russian scientist Tyurin in 1900.

In 1929, when Tausheck [162] obtained a patent on OCR in Germany, many people started dreaming about a machine that would read characters and numerals. This dream appeared feasible in the 1950's with the development of digital computers. Tausheck's patented work was based on the principle of template/mask matching. This principle, which used optical and mechanical templates matching, reflects the technology at that time. Light passing through mechanical masks is captured by a photodetector. When an exact match occurs, light fails to reach the detector and so the machine recognizes the characters. Since naive template matching is slow and costly, a modified logical matching scheme was designed. This scheme was called the *peehole* method. In this method, some appropriate pixels are chosen from both the black and the white regions of a character so that the selected pixels can distinguish the input character from characters belonging to other classes. The first OCR based on peehole method was announced by Solatran Electronics Group Limited in 1957 and was called *Electronic Reading Automation* (ERA) [102]. Later on an OCR was designed based on the same scheme by Iijima *et al.* [71] at *Electro Technical Laboratory* (ETL), in 1958. This system was more efficient and systematic than ERA system.

In the early 1960's, OCR researchers who started work on hand-printed character recognition, realized that it is difficult to create templates of hand-printed

characters because of the large variation of their shapes. This led to the proposal of new methods based on structural analysis. Since a structure can be broken up into parts, the features of these parts and the relationship between these parts can describe the structure. The problem therefore reduces to choosing appropriate sets of features and their inter- relationships such that they can identify each character clearly.

It has been explained by Mori *et al.* [102] that both the approaches of OCR namely, template matching and structural analysis are coming closer to each other and it seems that they might merge into one big stream.

Commercial OCR systems:

The commercial OCR systems can be divided into three generations depending on their versatility, robustness and efficiency [102].

The first generation of the OCR system can be characterized by the constrained letter shapes which the OCRs read. Such machines appeared in the beginning of the 1960's. The first widely commercialized OCR was the IBM 1418, which was designed to read a special IBM font, 407 [102]. The recognition method was logical template matching where the positional relationship was fully utilized.

The next generation can be characterized by the recognition capabilities of a set of regular machine printed characters as well as hand-printed characters. At the early stages the scope was restricted to numerals only. Such machines appeared in the middle of 1960's to early 1970's. In this generation, the first and famous OCR system was IBM 1287, which was exhibited at the 1965 New York world fair [102]. In terms of hardware configuration, the system was a hybrid one. It combined analog and digital technology. The first automatic letter-sorting machine for postal code numbers of Tosiba [56] was also developed during this period. The methods were based on the structural analysis approach.

The third generation can be characterized by the OCR of poor print quality

characters, and hand-printed character for a large category character set. Commercial OCR systems with such capabilities appeared roughly during the decade 1975 to 1985.

Present status of OCR:

At present, more sophisticated optical readers can be obtained [102,116,153]. These readers can process multi-font documents which can be typewritten, typeset, and printed by dot-matrix, line and laser printers. They can recognize characters with fonts sizes from 8 to 24 points at least, spaced between 4 and 12 lines per inch, in different formats including intermixed text and graphics. With the introduction of narrow range scanners, measuring 3 to 6 inches wide, columnar scanning is now available. With these, an optical reader can recognize multiple columns or sections of a page or mailing lists. Some are equipped with software for spell checking, and for flagging suspicious characters or words. See Table-1.1 for examples of some optical readers.

One challenging area of OCR is the recognition of hand-written documents [100,158]. However, in spite of many years of intensive research, commercial readers with good hand-written character recognition capabilities are rare. For the time being, recognition of hand- written characters, signatures and cursive script seems to belong only to on-line products where typically writing tablets are used to extract real-time information and features to aid recognition.

Among other commercial products postal address readers are also available in the market. In the United States, about 60% of the mail (mostly machine printed) is sorted automatically [111].

At present, reading aid for the blind is also available in the market. Integration of OCR with speech output has been marketed by Xerox-Kurzweil for English language [90].

Presently, OCR systems of printed Roman, Chinese and Japanese text are

Table 1.1: Examples of some optical readers.

Model	Company	Recognized characters	Scan speed	Resolution	Form sizes
CSL 2610	Computer Gesellschaft Konstanz mbh	Omnifont of typewriteRs and correspondence printers, upper-case alphanumeric handprinting	450 forms per hour (5.83 X8.27") 900 forms per hour (4.13 X 5.83")	----	From 2.70 X 5.10" to 5.90 X 8.20"
SLAM	Elsag (Genova, Italy)	Do.	200 cps for handwriting 500 cps for machine-printing	----	From 3.00 X 3.00" to 12.00 X 14.00"
Allfont 2710	Siemens	Do.	Up to 1,100 forms per hour up to 240 cps	230 dpi	From 3.00X 4.00" to 8.50 X 12.00"
Data-entry Polyform	AEG Olympia (Konstanz, Germany)	Do.	Up to 850 forms per hour (4.13 X 5.83")	----	From 3.94 X 5.83" to 12.52 X 8.54"
Compound Document Processor	Palantir Co. (Santa Clara, CA)	Omnifont of typewriters and correspondence printers	Up to 100 cps	300 dpi	From 3.00 X 5.00" to 8.50 X 14.00"
Kurzwell 4000	Kurzwell Computer Prod. (Cambridge, MA)	Do.	30 sec per form	300 dpi	Up to 11.00 X 14.00"
CBL 2310	Computer Gesellschaft Konstanz mbh	6 fonts: Prestige Elite 10, OCR B1, Pica 10, Courier 10, Courier 12 Ltter Grothic 10	300 forms per hour 300 cps	-----	Up to 8.27 X 11.69"

available in the market. However, to the best of our knowledge, no OCR system is available for Indian language scripts.

1.4 Major research work on OCR : a survey

At first, we discuss research work on foreign language script recognition. Next, recognition of Indian language characters is reviewed in some detail.

1.4.1 Work on non-Indian language scripts recognition

A notable early work in the area of character recognition was done by Grimpdate *et al.* [61] in 1958. Here, the input characters pattern obtained by a flying spot scanner is analysed for shape by a digital computer. The character pattern is described in terms of length, slope of the straight line segments and curvature of curved segments. The description is compared with that of the prototype stored in the computer in order to identify the unknown character.

In the early 1960's an important work is the analysis-by- synthesis method suggested by Eden [43] at M.I.T. Eden put forward the idea that all Latin script characters can be formed by 18 strokes, which in turn, can be generated from a subset of 4 strokes, namely, hump, bar, hook and loop. Somewhat similar studies were made by Blesser *et al.* [11], Cox *et al.* [36], Shillman *et al.* [142], Yoshida and Eden [184] and Berthod [7]. A theoretical approach based on phenomenological attributes has been proposed by Blesser *et al.* [11]. To take care of variability in type fonts Cox *et al.* [36] presented two main groups of grammar- like rules. Shillman *et al.* [142] mentioned three experimental techniques for studying ambiguous characters and for investigating relationship between physical and functional attributes. By extracting stroke sequences from the input pattern and using a look up dictionary of strokes, Yoshida and Eden [184] proposed a recognition system of

Chinese characters. Berthod [7] used primitives of Eden's method for cursive script analysis.

In 1968, using the statistical properties of the language, Casey *et al.* [18] developed an autonomous reading machine for printed Roman text. Here, the known character-pair occurrence frequencies of the language are used to identify the printed symbols without *a-priori* information about the structure of the characters. More about the use of n-gram statistics in the text processing has been described by Suen [157].

Graph theory is also used for OCR experiments. Kahan *et al.* [74] presented a system for printed Roman alphabet, where thinning and shape extraction are performed directly on a Graph of the run length of a binary image and statistical Bayesian classifier has been used for recognition. Later, in 1995 Rocha and Pavlidis [134] presented a segmentation-free approach for OCR. It was based on the recognition of subgraphs homomorphic to prototypes of characters.

Feng and Pavlidis [50] suggested a feature generation technique for syntactic pattern recognition by approximation on the basis of concavity. Later, in 1975, a split-and-merge algorithm has been utilized by Pavlidis and Ali [121] for the polygonal approximation of characters for the recognition of numerals.

In 1979, Agui and Nagahashi [1] suggested a description method for hand-printed Chinese characters. Here a Chinese character is represented by partial patterns using three relations, namely, concatenate, cross and near. The relation of relative location among partial patterns are used for categorization of the partial patterns. Using topological, spatial (zonal), curvature and cusp- shape properties, Bozinovic and Srihari [13] described an off-line cursive script word recognition system.

Fourier descriptors were first used for character recognition by Granlund [60]. Mahmoud [96] used Fourier descriptors and character contour encoding for the

recognition of Arabic characters. He computed Fourier descriptors and curvature features of the primary part of the character. These features of the training set are used as the model features. The features of an input character are computed and compared to the model features using a distance. The model with minimum distance is taken as the class representing the character. Also, El-Sheikh and Guindi [49] used fourier descriptor for Arabic cursive script recognition.

Two dimensional moments have been used by many researchers for the purpose of character recognition [20,48]. Here, the features used are the moments of black points about a chosen center. Later, El-dabi *et al.* [47] presented a system for cursive type-written Arabic character recognition using accumulative invariant moments.

In 1988, Sekita *et al.* [139] presented a method of extracting features of hand-written Japanese characters using spline approximation. The method represents a character by contours expressed by well approximating functions and stable break-points which characterize the connection of the strokes so that it provides proper feature for recognition with relaxation matching. Lam and Suen [91] proposed a method for totally unconstrained hand-printed zip-code number recognition. Here, at first they used a structural classifier that identifies majority of the samples and then they applied robust relaxation algorithm which classified the rest of the data. A new relaxation method based on feature reflecting structural information for hand- printed Chinese character recognition was introduced by Xie and Suk [180]. They defined a distance measure based on matching probabilities computed by relaxation technique for distinguishing similarly shaped characters within a cluster produced by pre-classification.

Hidden Markov model based approach and Viterbi algorithm have been used in the field of character recognition. Kundu *et al.* [89] described a hand-written word recognition system using first and second order hidden Markov model based approach. Using the existing statistical knowledge of English, the calculation

scheme for model probabilities are immensely simplified and after establishing the model they used Viterbi algorithm to recognize letter sequence of the word. Similar techniques were also used by Hull and Srihari [69] as well as Singhal and Taussiant [144] for character recognition. In 1996, Mohamed and Gadev [99] presented an approach for hand-printed word recognition. Their approach was the combination of segmentation-free hidden Markov model and segmentation- based dynamic programming techniques.

Hough transform based techniques have been applied to character recognition by many researchers. In 1983, Kushnir *et al.* [53] applied Hough transform for the recognition of printed Herbrew characters. They employed a two-stage approach. In the first stage, a coarse classification is performed and in the second stage the classifier is divided into two parts. In one part the decision is based on matching features in the Hough space and in the other part a structural classification technique is employed.

Among other transforms, Karhunen-Loeve expansion has been used by Kimpan *et al.* [85] for classification of printed Thai characters. Huang and Lung [70] applied Walsh transform for complex Chinese character recognition.

Recognition via neural networks was studied in late 1980's. Compared to other methods, the advantage of neural network is that it offers self learning of the system from examples [137]. Neural nets have been used for postal zipcode, bank cheque and credit card reading [82]. Self-organizing maps and fuzzy rules have been used by Chi *et al.* [32] for hand-printed numeral recognition. A two-phase hand-printed word recognizer that combines a neural net recognition technique with a probabilistic correction technique has been presented by Burr [14]. Wang and Jean [171] used neural network for resolving multi-font character confusion. Later, they proposed a method for segmentation of merged characters by combining shortest path criterion with neural net [172].

Among OCRs of non-Roman scripts, major research has been done on Chinese

and Japanese characters. The earliest reported attempt on printed Chinese character recognition was perhaps by Casey and Nagy [17] in 1966. For the large number of characters of the Chinese alphabet, they developed a two stages approach. The first stage was for the initial grouping of similar shaped characters and the final stage was for resolving individual character identity. The various other techniques employed for the recognition of Chinese characters can be found in the review work of Stallings [156], Mori *et al.* [101] and Nagy [107] etc.

1.4.2 Work on Indian language character recognition

Studies on Indian character recognition are few. Majority of them are concerned about Devnagari and Tamil script characters. Devnagari is the most popular Indian script used for writing Hindi, Sanskrit and some other languages. Some studies are also reported on Telugu and Bangla. They are briefly reviewed below. It should be mentioned that all the studies deal with isolated basic character recognition where the large set of compound characters is ignored. To the best of our knowledge no report other than ours deals with a complete OCR system of printed document.

Studies on Devnagari character recognition:

On the basis of presence or absence of some basic primitives, namely, horizontal line segment, vertical line segment, left and right slant, D-curve, C-curve etc. and their positions and interconnections, Sethi and Chatterjee [140] presented a Devnagari numeral recognition system based on binary decision tree classifier. They [141] also used a similar technique for constrained hand-printed Devnagari character recognition.

Among others, Sinha and Mahabala [148] presented a syntactic pattern analysis system with an embedded picture language for Devnagari script recognition. The system stores structural descriptions for each symbol of the script in terms of primitives and their relationships. The recognition involves a search for primitives

on the labeled pattern based on the stored description and context. Sinha also reported knowledge based contexual post-processing systems for Devnagari text recognition [145,146]. He demonstrated how the spatial relationship among the constituent symbols of Devnagari script plays an important role in the interpretation of Devnagari words.

Studies on Tamil character recognition:

Siromony *et al.* [150] described a method for recognition of machine printed Tamil characters using an encoded character string dictionary. The scheme employs features in the form of strings which are extracted by row-wise and column-wise scanning of character matrix. The features in each row and column are encoded suitably depending upon the complexity of the script to be recognized. Chandrasekaran *et al.* [21] used similar approach for constrained hand-printed Tamil character recognition. Later on, Chandrasekaran *et al.* [22] applied the same technique for multi-font Tamil, and special sets of printed Malayalam and Devnagari character recognition.

Chinnuswamy and Krishnamoorthy [33] proposed an approach for hand-printed Tamil character recognition employing Directed Labeled Graphs (DLG) to describe structural composition of characters in terms of primitives and the relational constraints satisfied by them. Primitives are obtained after thinning the character. Recognition has been done using correlation measure of the directed labeled graph of the input character with those of the prototypes.

Studies on Telugu character recognition:

For Telugu alphabet, a two-stage recognition system has been presented by Rajasekaran and Deekshatulu [127]. In the first stage they applied a knowledge based search to recognize and remove the primitive shapes present in the input character. A directed curve-tracing method is used for the purpose. In the second stage, the pattern obtained after the removal of primitives is coded by tracing

along points on it. On the basis of the knowledge of primitives and basic characters present in the input pattern, classification is achieved by means of a decision tree.

Studies on Bangla character recognition:

Ray and Chatterjee [128] presented a nearest neighbor classifier employing features extracted by using a string connectivity criterion for Bangla character recognition.

Dutta [40] presented a generalized formal approach for generation and analysis of all Bangla and Hindi characters.

Dutta and Chaudhuri [41] reported a work on recognition of isolated Bangla alphanumeric characters. The characters have been represented in terms of the primitives and structural constraints between the primitives imposed by the junctions present in the characters. The primitives have been characterized on the basis of the significant curvature events like curvature maxima, curvature minima and inflectional points observed along their extends. A two stage feed forward neural net, trained by the well-known back-propagation algorithm, has been used for the recognition. The structural constraints imposed by the junctions have been encoded in the topology of the network itself.

1.5 Scope and layout of the thesis

The work of the present thesis is mainly development motivated. It is a major part of a more elaborate project of developing a reading system of Bangla for the visually handicapped. The work described here is a development of a complete OCR system on single font printed Bangla text.

At first, this thesis deals with the analyses of the Bangla language characteristics and how these characteristics influence the design and accuracy of OCR system. Next, it deals with OCR development on single font printed Bangla script.

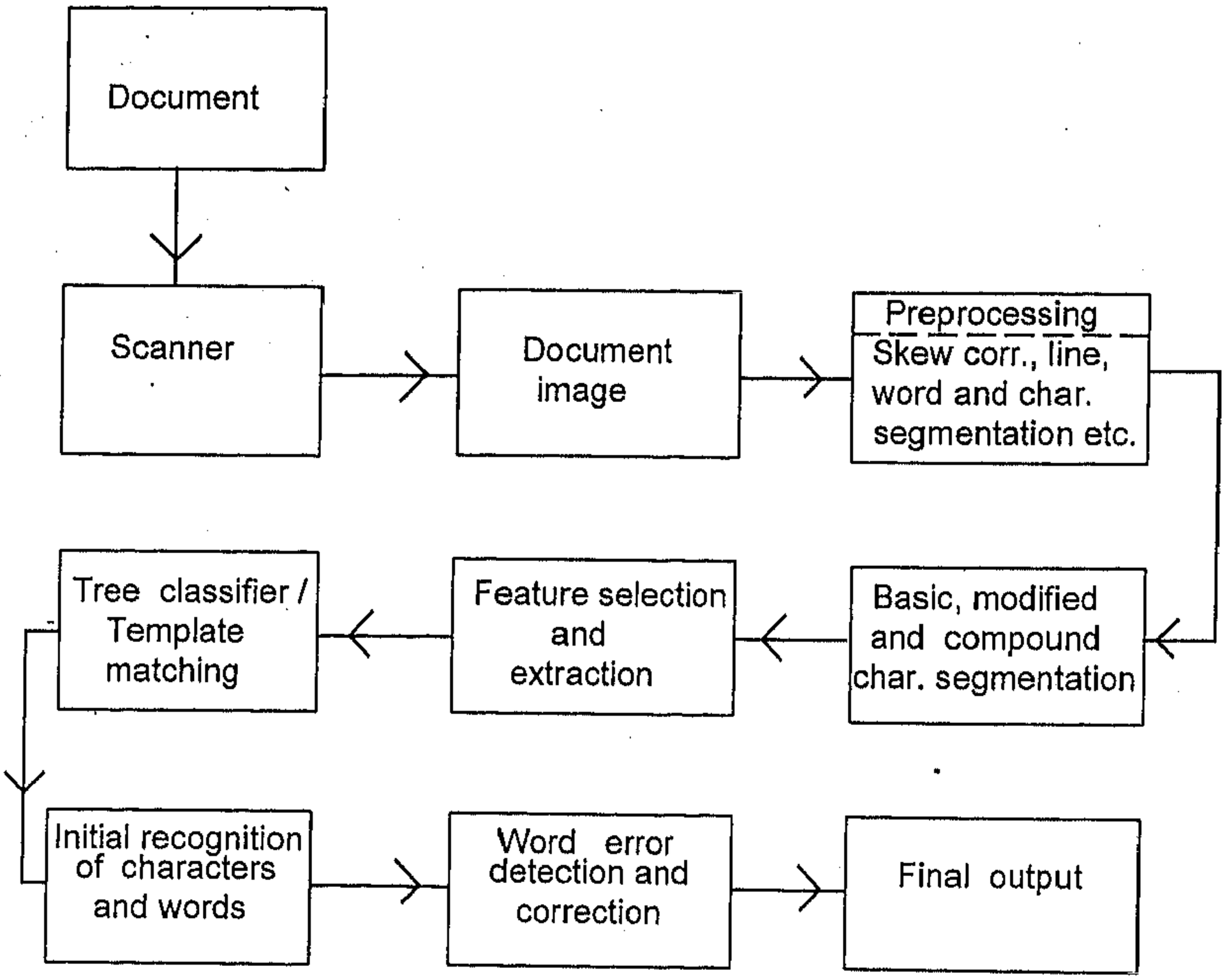


Figure 1.1: Block diagram of the OCR system.

Block diagram of the OCR system is shown in Fig.1.1.

The contents of the thesis may be broadly divided into three major parts. They are:

Part I: **Preprocessing Division.** This part consists of two chapters, Chapter-2 & 3.

'art II: **Recognition Division.** This part consists of two chapters, Chapter-4 & 5.

Part III: **Postprocessing Division.** This part consists of one chapter, Chapter-6.

Preprocessing Division:

Here, at first some statistical studies have been made on Bangla script characters. The application potentials of these studies are also described. Next, binarization, smoothing as well as skew detection and correction techniques have been elaborated. Our skew correction method is simple, fast and robust. The method has been developed using inherent characteristics of the script form. Finally, techniques for zone detection, segmentation of text into lines, words as well as separation of characters from words have been described.

Recognition Division:

Here, initially feature selection and detection procedure as well as some analysis on binary tree classifier has been presented. Next, we have described automatic character recognition procedure using a hybrid method. Combination of a feature based tree classifier and a run length based template matching approach is employed for the purpose. We have grouped the characters into three classes namely, basic, modifier and compound character. The basic and modifier class characters are recognized by a feature based tree classifier. The compound character recognition involves a template matching approaches preceded by a feature based subgrouping.

Postprocessing Division:

Here, we have proposed an OCR error detection and correction technique for an highly inflectional language like Bangla. Using two separate lexicons of root words and suffixes, we detect candidate root-suffix pair of each word and test their grammatical agreements, and note the root/suffix part in which the error has occurred. The correction is done on the corresponding error part of the input string by a fast dictionary access technique.

Chapter-wise break-up:

The chapter-wise break-up is given below.

hapter 2 This chapter deals with the study of graphemic character occurrences in words of Bangla (Bengali) language. The occurrence statistics of characters are presented for words collected from newspapers, novels, popular magazines and juvenile literature as well as dictionary. Some of the computed statistics are the occurrences of vowel/consonant (unigram) percentage, their positionwise occurrences, percentage of compound characters, bi-gram, tri-gram etc. Their application potentials are also discussed.

hapter 3 Different methods of preprocessing like document image data collection, binarization, smoothing, skew detection and correction, text and graphics separation, line segmentation, zone detection, word and character segmentation using some conventional and some newly developed techniques have been presented in this chapter.

hapter 4 At first this chapter describes separation procedure of basic, modified and compound characters from zonal information and shape characteristics. Next, feature selection and detection procedure as well as some analyses on binary tree classifier have been presented.

hapter 5 Character recognition procedure is considered in this chapter. The basic and modified characters, which are about 75 in number and which occupy about 94% of the text corpus, are recognized by a structural feature based tree classifier. Compound characters are recognized by a tree classifier followed by a sophisticated template matching approach.

Chapter 6 This chapter describes an OCR error detection and correction technique which will be useful for the inflectional language scripts. At first, the OCR recognized text error pattern is analysed. Next, using two separate lexicons of root words and suffixes, candidate root-suffix pair of each input word are detected, their grammatical agreements are tested and the root/suffix part in which the

error has occurred is noted. Finally, the correction is made on the corresponding error part of the input string by a fast dictionary access technique. For an erroneous word, some candidate strings are generated and for each such candidate string a weight is assigned. Among the candidate strings, which are in grammatical agreements, the one with maximum weight is selected as correct word.

The thesis concludes with the scopes of future works.

PREPROCESSING

DIVISION

Chapter 2

BANGLA SCRIPT STATISTICS

2.1 Introduction

There are four categories of languages in India [164] : (1) Indo European (Hindi, Marathi, Gujrati, Panjabi, Kashmiri, Bengali, Assamese and Oriya etc.); (2) Dravirian (Telegu, Tamil, Kannada and Malayalam etc.); (3) Sino Tibetan; and (4) Austric, Hittite etc.

Bangla, the secondmost popular language in India and the fourthmost popular language in the world [169], originated from Sanskrit, an ancient Indo-Aryans language. About 200 million people in the eastern part of Indian subcontinent speak in this language. Bangla script alphabet is used in texts of Bangla, Assamese and Manipuri languages. Also, Bangla is the national language of Bangladesh.

Bangla is a phonetic language, because spelling of a character remains largely unaltered in the word. The Bangla graphemic character shape is more complicated than English character shape because of the presence of compound characters (see next section). Also, Bangla is a highly inflectional language where word roots (words having no *suffix* may be called *root words*) are more frequently associated with

prefix and *suffix*. Compounding of several roots in the form of *euphony (sandhi)* and *assimilation (samasa)* are also quite common. More about Bangla script and language characteristics are discussed in Section 2.2.

The study of character occurrence frequencies in any language is important and useful in problems related to spelling correction [44,157], speech recognition [42,67,73], natural language processing [163], OCR [44,133], cryptography [54] etc. The primary aim of our study is to make a complete OCR system. So graphemic character statistics are presented here though we have also computed both graphemic and phonemic statistics of Bangla. For phoneme statistics see the paper [25].

To compute the statistics, two sets of data were collected. Section 2.3 deals with data collection, computed statistics and their application potentials. Some of the computed statistics are the occurrences of vowel/consonant (unigram) percentage, their positionwise occurrences in the usual text, percentage of compound character, word length, bigram, trigram etc. Their importance in our OCR design is also discussed. Some statistics are computed to judge the efficiency of a dictionary look up method [157] for OCR word error correction. The results are described and explained in Section 2.4.

There exist many papers on the study of character occurrence in European language scripts [44,133,157]. Among Indian languages, some studies on *Hindi* are reported [84,164] (Hindi is the most popular language in India). *Devnagari* alphabet is used to write or print Hindi text. In Bangla, a few studies of statistics computed from small corpus of words are reported. Das *et al.* [38] computed global character occurrence in Bangla script. The motivation of that work was to design keyboard for Bangla typewriter. Their study did not take care of the compound character occurrence and other higher order statistics. Bhattacharia [8] computed word length and sentence length in unit of syllable number for different Bangla novels. His aim was to make a study on literary style of authors based on these statistics. Some

analysis on Bangla phoneme occurrence in connection with the study of shifts from Sanskrit to Bangla was reported in the classic book of Chaterjee [23].

2.2 Characteristics of Bangla script and language

2.2.1 Properties of Bangla script

All major Indian scripts including Bangla are mixtures of syllabic and alphabetic scripts [40]. They are varied in character and form. Around 250 BC two different scripts took shape in India. They are *Indo-Bactrian* or *Khorosthi* script and the *Brahmi* script. The Bangla script is derived from the Brahmi script through various transformations. The genetic structure of Bangla script is given in Fig.2.1

Charles Wilkins did the first punch-cutting for printing books in Bangla alphabet. The first printed book containing Bangla alphabet is perhaps the Bangla grammar due to N. B. Halhed published in 1778 [159]. The punch-cutting of Bangla typeface was further advanced by Panchanan Karmakar and his family successors at Serampore Mission Press, Hooghly. The Missonaries of Serampore did a great service to the development of printing and publishing of Bangla books and documents.

Book printing industry in Bengal advanced quite fast with time and some font variations were also introduced. Of them *Lino* is the most popular font used by mass printing media. Bangla Lino font was started in 1935 by the management of Bangla newspaper *Ananda Bazar Patrika.* Mr. S. K. Bhattacharia created the Lino font with the help of Mr. J. K. Sen. Mr. Suresh Chandra Majumder and Mr. Rajsekhar Basu were instrumental in this font development project. Several other font styles are used by different publishing industry including Monotype and intertype. Some other recently reported fonts are Vivek, Geetanjali, Anand, Prakash etc. but they are yet to be popular.

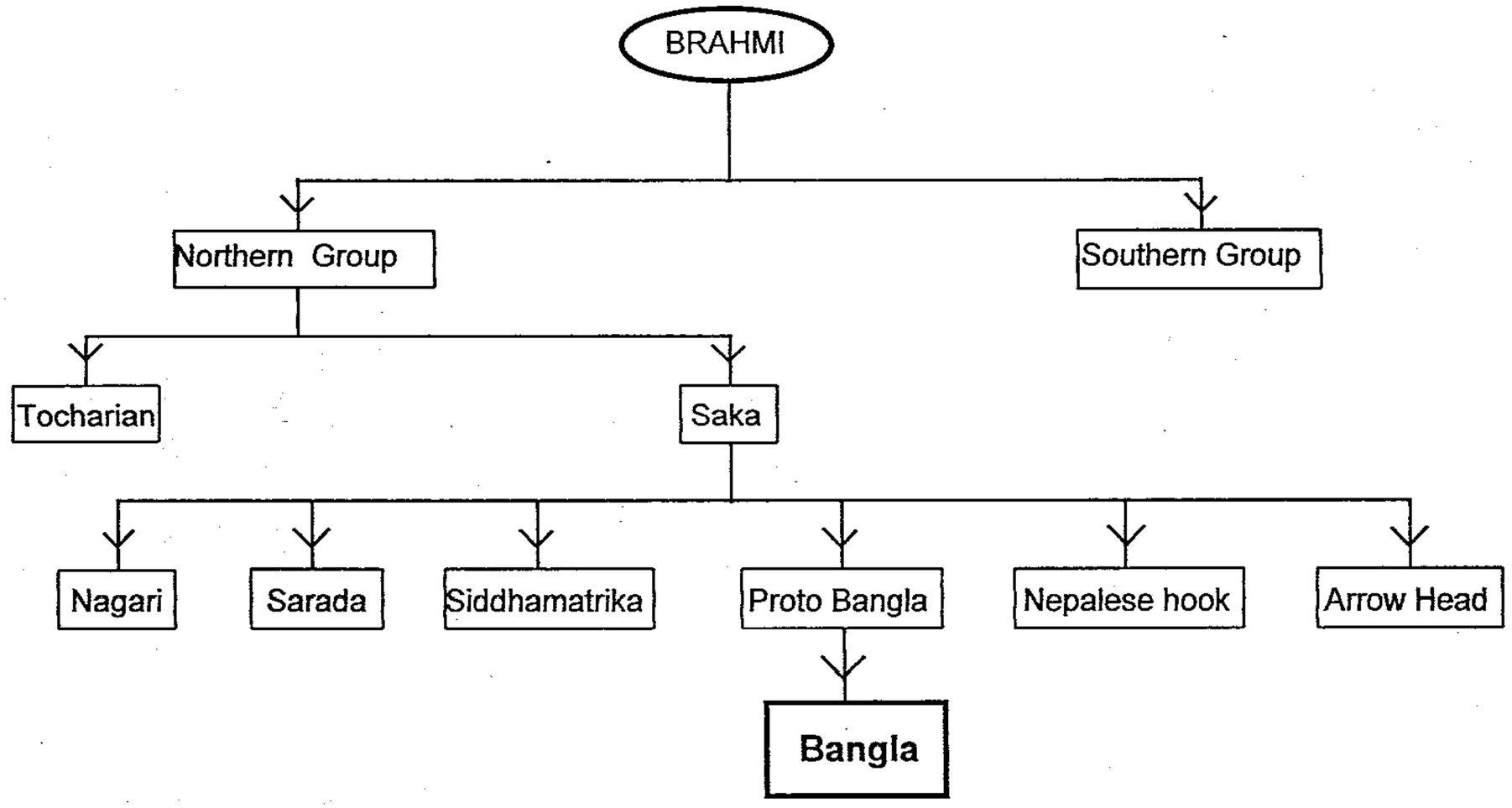


Figure 2.1: The genetic structure of Bangla script.

The main features of Bangla script are as follows:

- There are 11 vowels, 40 consonants and 10 numerals in Bangla alphabet. They are called *basic characters*. The basic characters and ASCII codes used to represent them in computer are shown in Fig.2.2. Out of 40 consonants in Bangla, two characters have same shape (see consonant character number 23 and 29 of Fig.2.2(b)). Thus, we have to recognized 60 basic characters. In Bangla, characters are not alphabetical, like English-Roman where the characters largely have one sound one symbol characteristics. It is a mixture of syllabic and alphabetic characters. However, the characters are well organized in terms of place and manner of articulation. The concept of upper/lower case is absent in Bangla. Writing style of Bangla is from left to right in a horizontal manner. There are several punctuation marks in Bangla. They are shown in Fig.2.2(d).

- Many Bangla characters have a horizontal line at the upper part. This line is called *matra or headline* (See Fig.2.4). It plays an important role in our Bangla OCR system. It helps us to detect the skew angle (defined in the next chapter) of a digitized document, to locate the script line, to segment the characters in a word and to classify the characters. Some characters have an arc-like extension above the headline (e.g. ই , ঈ etc.). Presence of this line is also useful for character classification.

- A vowel (other than A(অ)) following a consonant takes a modified (allographic) shape, which depending on the vowel is placed at the left, right(or both) or bottom of the consonant. These are called *vowel modifier*. For example, see Fig.2.3(a). Some vowel modifiers have two parts. One sits to the left and other to the right of a consonant (or compound) character. For example, the 9th character of Fig.2.3(a) has two parts. For the recognition of this modifier we have considered it as two characters. Interestingly, the part that sits to the left is the modifier of another vowel E(এ) and the other

অ	আ	ই	ঈ	উ	ঊ	ঋ	এ	ঐ	ও	ঔ
A	A,	I	I,	U	U,	R.	E	(AI)	O	(AU)

(a)

ক	খ	গ	ঘ	ঙ	চ	ছ	জ	ঝ	ঞ
K	KH	G	GH	N,	C	CH	J	JH	N+
ট	ঠ	ড	ঢ	ণ	ত	থ	দ	ধ	ন
T,	T,H	D,	D,H	N*	T	TH	D	DH	N
প	ফ	ব	ভ	ম	য	র	ল	ব	শ
P	PH	B	BH	M	J,	R	L	B	S.
ষ	স	হ	ড়	ঢ়	য়	ৎ	ং	ঃ	ঁ
S,	S	H.	R,	R*	Y	T.	M,	H,	N.

(b)

১	২	৩	৪	৫	৬	৭	৮	৯	০
1	2	3	4	5	6	7	8	9	0

(c)

।	,	ঃ	;	?
.	,	:	;	?

(d)

Figure 2.2: Basic characters and punctuation marks in Bangla with their ASCII codes.
(a) vowels (b) consonants (c) numerals (d) puctuation marks.

part which sits to the right is the modifier of another vowel Á,(আ). Clearly, Bangla script is not strictly a left to right alphabetical sequence. Some of the vowel modifiers have portion that goes above the matra.

- If the first character of the word is a vowel then it retains its basic shape. For two consecutive vowels in a word, the second one retains its basic shape. Occasionally, as in বই the vowel ই retains its basic shape, though the first character is not a vowel. This is to indicate that the word is actually ব + অ + ই and not ব + ই . *Hence, the second of the consecutive* vowels retains its shape. The vowels U(উ), U,(ঊ) and R.(ঋ) may take different modified shapes when attached to some consonant characters. They also change the shape of some consonant characters to which they are attached. For illustration see Fig.2.3(b).

- Compounding of a consonant following another consonant is sometimes represented by an allograph which we call as *consonant modifier*, as shown in Fig.2.3(c). If the first consonant is R(র) then its modified symbol, called *ref* (◌র্) is placed in the upper part of the basic shape of the second consonant. Sometimes, the first consonant basic shape is retained, while the second consonant shape is modified. This is so when the second consonant is R(র) or J,(য).

- A word may be partitioned into three zones. The upper zone denotes the position above the headline, the middle zone covers the portion of basic (and compound) characters below headline and the lower zone is the portion where some modifiers can reside. A typical zoning is shown in Fig.2.4.

- More formally speaking, modifiers are those symbols which do not disturb the shape of the basic characters (in middle zone) to which they are attached. If the shape is disturbed at the middle zone, we call the resultant shape as *compound character (clustered character)* shape. Compound characters can be combinations of consonant with consonant as well as consonant with vowel.

Vowel	আ	ই	ঈ	উ	ঊ	ঋ	এ	ঐ	ও	ঔ
Modified Shape	া	ি	ী	ু	ূ	ৃ	ে	ৈ	ো	ৌ
with number	1	2	3	4	5	6	7	8	9	10
and code	a,	i	i,	u	u,	r.	e	(ai)	o	(au)
When attached to consonant ক	কা	কি	কী	কু	কূ	কৃ	কে	কৈ	কো	কৌ

(a)

Consonant	গ	র	শ	হ	র	হ
Vowel	ঊ	ঊ	ঊ	ঊ	ঊ	ঋ
Combined shape	গু	রু	শু	হু	রূ	হৃ

(b)

Consonant	য	র	র
Modified shape	্য	্র	্র
with number	1	2	3
and code	j	r	r*
When attached to consonant দ	দ+য = দ্য	দ+র = দ্র	র+দ = র্দ

(c)

Figure 2.3: Examples of vowel and consonant modifiers (a) vowel modifiers (b) exceptional cases of vowel modifiers (c) consonant modifiers.

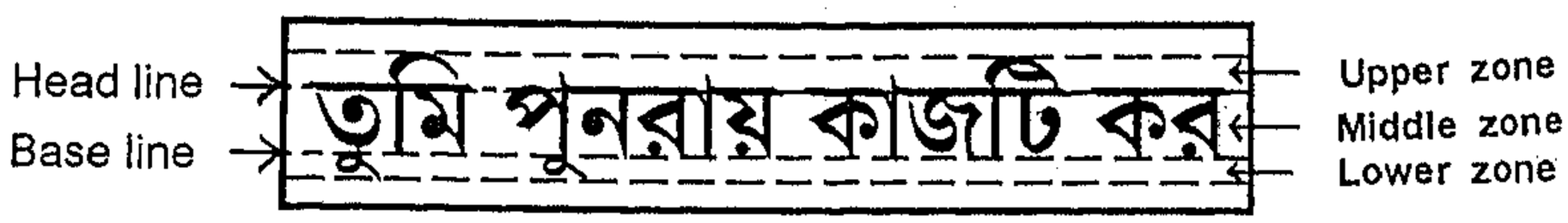


Figure 2.4: An example of a typical zoning of a Bangla text line.

Compound characters formed by consonant-vowel combination are few. A subset of 100 compound characters is shown in Fig.2.5. Compounding of three consonants is also possible. The vowel and consonant modifiers may be attached to compound characters.

Some examples of compound character formation is shown in Fig.2.6. Occasionally, combination of two characters forms a new shape as shown in the first two rows of Fig.2.6. In the third row of Fig.2.6 one character partially retains its shape. The other character has changed its shape completely and sits below the other. In the fourth row, two characters sit side by side in compounding, but the size of the first character is reduced. In the compound character depicted in the fifth row a portion of one character and a portion of the other is visible. The other compound characters of Fig.2.6 can be understood in a similar way.

- The total number of basic, modifier, compound character and numeral is in Bangla is about 300. Although the writing style is from left to right, a word may not be partitioned vertically into a left-right character sequence (remember the case of modifier for vowel O(ৌ), of which one part sits to the left and one to the right of the consonant). Thus, a delayed decision is necessary during recognition of characters.

2.2.2 Characteristics of the Bangla language

We describe here the basic linguistic characteristics of Bangla. As stated before, Bangla is an inflectional language. In the text, a Bangla word may consist of an optional prefix, a simple or compounded stem of root words, zero or more internal affixes and an optional termination declension. The set of words that can be used in Bangla text may be called *surface words*. The major affix classes in Bangla are verbal and nominal declensions called *Bibhaktis*, verbal and other internal af-

ল্ক	স্ক	ক্ষ	ঙ্ক	জ্ঞ	স্থ	ল্ম	ঙ্ঘ	দ্ঘ	দ্গ
চ্চ	ঞ্চ	শ্চ	চ্ছ	ঞ্ছ	শ্ছ	ঞ্জ	জ্জ	জ্ঝ	স্ট
ক্ট	ণ্ট	প্ট	ল্ট	ষ্ঠ	ষ্ঠ	ড্ড	ল্ড	ল্ল	ক্ম
ক্ত	স্ত	ন্ত	প্ত	স্থ	দ্দ	ন্দ	ব্দ	দ্ধ	ন্ধ
ব্ধ	গ্ন	ত্ন	হ্ন	প্ন	শ্ল	প্ল	ম্প	ম্প	ম্প
স্ফ	স্ফ	ল্ফ	ক্ব	জ্ব	ট্ব	ধ্ব	ল্ব	শ্ব	স্ব
ঙ্ক	ত্ম	ঘ্ম	ন্ন	ম্ম	হ্ম	ক্ষ	ন্ত	ঙ্ক	জ্জ
হ্ম	ট্ম	ঠ্ম	ন্ম	গ্ম	ঙ্ম	ঞ্চ	ল্ল	ম্ল	ক্ল
গ্ল	প্ল	গ্ম	ণ্ম	ন্ম	দ্ম	ভ্র	ত্ত	স্ফ	ন্ত্র
ন্ত্র	ন্দ্ব	ঙ্ক্ষ	ঙ্ক্ষ	জ্জ্ব	স্প্র	ত্ত্ব	ক্ষ্ণ	ক্ষ্ণ	ন্ত্ব

Figure 2.5: A subset of 100 compound characters.

fixes called *Pratyayas*, nominal suffixes and prefixes called *anusargas* and *upasargas* respectively. The root words are classified into several clusters like verb-root, noun-root etc. The morpho-syntactic rewriting rules are based on these classes. For example, a verb word and a noun word in the text may be conjoining of the form

VERB WORD – – –– > verb root [causational affix] verb declension

NOUN WORD – – –– > noun root [definiteness affix] case declension.

More complex conjoining of the form

Noun root – – –– > verb root [causational affix]

is also possible. The conjoining is not always a simple concatenation. Morphological distortion of the root word is also possible. The morpho-phonemic or morpho-graphemic restructuring of symbols at the boundaries of two conjoining morphemes are governed by the *spelling rules* of generative morphology. The spelling rules make the job of detecting the morphemic boundaries more difficult. A spelling rule may

Consonant characters		Compound characters
ক + র	=	ক্র
ক + ষ	=	ক্ষ
গ + ধ	=	গ্ধ
চ + চ	=	চ্চ
জ + ঞ	=	জ্ঞ
প + ত	=	প্ত
শ + চ	=	শ্চ
স + ক	=	স্ক
স + থ	=	স্থ
ক + ষ + ণ	=	ক্ষ্ণ
ঙ + ক + ষ	=	ঙ্ক্ষ
জ + জ + ব	=	জ্জ্ব
ন + ত + ব	=	ন্ত্ব
ন + ত + র	=	ন্ত্র

Figure 2.6: Some examples of compound character formation.

affect (or be affected by) the decision of the rule at some other morpheme boundary. Moreover, conjoining of two or more roots or words in the form of *sandhi* and *samasa* is a common phenomenon in Bangla word formation. In sandhi, morphological distortion of the constituent words are made according to some set of rules. In samasa, the conjoining is often simple concatenation of the constituent words.

The detail of linguistic properties of Bangla sentence is beyond the scope of this thesis. However, the following basic characteristics can be noted:

- The verb in a sentence sits usually at the end of the sentence.

- The word (or phrase) order in a sentence is not restrictive. Change of phrasal position often keeps the sentence syntactically valid and carries the same meaning as the original.

- The words in the sentence are related to the verb by *Karaka* relation, marked by the suffixes called *Bibhaktis*.

- A long sequence of events can be described in a single sentence by using non-finite verbs.

- Almost all common nouns, pronouns, adjectives, verbs and adverbs can appear more than once in a sentence.

There exist two forms of the language namely *Sadhu* (formal) and *Chalit* (informal) which differ mainly in the representation of verb and suffix. Sadhu, implying 'better' or 'pure', is more traditional in form. The Sadhu form is nearly obsolete these days and our statistics are based on script written in Chalit form. About fifty years ago Sadhu form was more popular for writing. Sentences in old Sadhu form usually contain words with larger number of compound characters. Mixture of two forms in a single text is considered to be a *Guruchandali error*.

Because of inflectional nature, a huge number of surface words can be formed out of a moderate number of root words. For example, in a dictionary of 50,000 root words there are about 800 single worded verbs. There are about 125 verb suffixes (neglecting dialectical variations) of which a particular verb can usually accept about 100 suffixes to form valid *surface words*. Thus, about 80,000 surface verb words can be formed. The number of non-verb suffixes is also of similar order. Naturally, the storage and search time could be reduced if we could have separate *root word dictionary* and *suffix dictionary* files. However, surface word formation from root and suffixes, as well as surface word parsing into root words and suffixes are both difficult because a large number of rules and their exceptions should be considered. For our OCR error detection and correction, which is based

on dictionary search, we simplified the procedure to a large extent by making the dictionary such that:

1. If a root word can accept a prefix then the prefixed root word is considered as another root word. Conjoining of root words by *Sandhi* and *Samasa* are also considered as new root words.

2. If morphological deformation is caused in a root word in order to accept a set of particular suffixes, then the deformed root word is considered as another root word.

Thus, we have our root words and suffixes such that any surface word is a simple concatenation of a single root word string followed by (optional) single suffix string. In doing so, we can avoid a large number of rule verification during OCR error detection and correction without appreciable increase in the size of root word dictionary.

Each language has a small set of abstract linguistic units, called phonemes, to describe its sounds. A language can have about 20-50 phonemes, which provide an alphabet of sounds to uniquely pronounce words in the language. It has been noted that 43 phonemes are adequate to speak in Standard Colloquial Bangla (SCB) [40]. Of these 43 phonemes, there are 7 oral vowels, 7 nasal vowels, 1 glide and 28 consonants. There exists controversy on the actual number of *Dipthongs* in Bangla. A large number of them do not occur in any word but are found to be associated with continuous speech, particularly when the preceding word has a terminal vowel and the following word has an initial vowel. Bangla, like most of the Indian languages, is very rich in consonants, particularly *plosives* and *affricates*. The plosives and affricates arise from five different articulatory positions ranging between velar to bilabial. Also, every articulatory position has the following four manners of production : voiced unaspirated, unvoiced unaspirated, voiced aspirated

and unvoiced aspirated. The large number of consonants, along with 7 oral vowels distinguish the spoken form of Bangla from European languages.

2.3 Data collection, computed statistics and their application potentials

2.3.1 Data collection

For our experiment, two sets of data were collected. The first set of data were gathered from popular Bangla books, newspapers, magazines, novels and juvenile literature using the method of random sampling. This data set containing nearly 2.5 million words were obtained from the Centre for Applied Language and Translation Studies (CALTS), University of Hyderabad, India. The second set of data contains the words collected from a Bangla dictionary [9]. Nearly 60,000 words belong to this set of data. Data were stored in the ISCII (Indian Standard Code for Information Interchange) format using GIST (Graphics and Indian Script Terminal) multilingual card.

2.3.2 Computed statistics and their application potentials

At first, global occurrence frequencies of vowels, consonants and compound characters were computed. When represented as a fraction of total number of characters in the data set they give estimate of apriori probabilities of these classes of characters. These statistics provide an idea of which characters should be recognized correctly for higher recognition rate. Also, these statistics help us to design a tree classifier for recognition. In Chapter 4, the tree classifier will be discussed in detail. The average wordlength as well as the suffix statistics of words were computed. They are useful for the design of error detection and correction model of the system.

Next, the positionwise occurrence statistics of the characters and their ranks with respect to their occurrence in different position are found and tabulated. Positionwise occurrence statistics help OCR system for detection of error. For example, suppose the probability of occurrence of a character X in the first position of a word is zero. If in an OCR output the first character of a word is X then we conclude that an error has occurred in the first position.

The frequency of occurrence of words are computed for various wordlengths and the results are plotted in a graphical form for both sets of data. These graphs may be called the wordlength histograms of the data set. They provide a pictorial view about human trend of using words of various wordlengths.

We have also computed the probability that a Bangla word will have at least one character with headline. This statistics has been applied in modeling the skew detection approach.

The occurrence of a character preceded by another in a word is represented in the form of bigram. If there are n characters in an alphabet then the bigram contains $n \times n$ entries in which the (i,j)-th entry represents the frequency of occurrence of i-th character followed by j-th character. When computed from a large data set, it gives an estimate of conditional probability of occurrence of j-th character given that i-th character has occurred in a word. If it is assumed that the words are the outcome of a Markov process, then this conditional probability can be used in various problems including error detection and correction. For example, suppose the occurrence probability of the bigram RX is zero. Now, if R appears before X in an OCR output then we can assume that an error has occurred.

As compared to bigram, the global frequency of occurrence of characters may be called as uni-gram statistics. On the other hand, higher order statistics like trigram and in general, k-gram can be computed. However, a k-gram contains n^k entries and its computation needs huge data and computer time. Usually, people do not go beyond trigram in OCR problems. N-gram statistics has been used by

many authors for OCR word error detection and correction. More about n-gram technique is described in Chapter 6.

We have computed the number of valid words obtained by replacing a character with its graphemically similar character in a word. From this statistics we can judge whether dictionary look- up technique can be effective in OCR error detection and correction. For example, the word Ga,L (গাল) has been recognized by a valid word Ga,N (গান) although the the character L(ল) has been wrongly recognized as N(ন). In this case, dictionary look-up cannot correct the error.

The various computed statistics have, in general, many application potentials. Some of the possible applications are given below in a tabular form.

Statistics computed	Application potentials
Global percentage of vowels, consonants, compound characters etc.	Character recognition, Cryptography.
Global character statistics.	Character recognition, Cryptography, Keyboard setting.
Suffix statistics.	Linguistics, Cryptography. Error correction.
Position statistics, Bi-gram, Tri-gram.	Character recognition, Speech synthesis and analysis, Cryptography.
Replacement characteristics.	Spell checking by dictionary, Character recognition etc.

2.4 Results and discussion

Table-2.1 represents the global statistics of two sets of data and shows some interesting comparisons. In commonly used language (as in current newspapers, books, novels, magazines and juvenile literatures) the vowel percentage is larger and the compound character percentage is smaller than that in the dictionary. In other words, we have a tendency of using words which are easy to spell and mellowed to listen. Table-2.1 also confirms that Bangla is a highly *inflectional language*, where nearly 70% of the words have suffixes attached to them. Since the second set of data contains the *root words* from dictionary, suffix statistics is absent. We have not computed the prefix statistics, and assumed that the words made by concatenating prefix with a root word is also a root word.

Table 2.1: Global characteristics of 1st and 2nd set of data.

	Global characteristics	1st set of data (Corpus)	2nd set of data (Dictionary)
1.	Vowel characters	39.82%	37.70%
2.	Consonant characters	54.01%	51.71%
3.	Compound characters	6.17%	10.59%
4.	Average word length	4.56 (chars)	5.12(chars)
5.	Words with suffix	69.91%	0.00%

We have noted that in present day Bangla text the compound character percentage is very small. This confirms the tendency of using words which are easy to spell and mellowed to listen. The occurrence of compound characters in the popular documents is 6.17%, though there are more than 200 compound characters in Bangla. Hence, we can put secondary importance on compound character recognition compared to basic and modifier character recognition.

To test the likelihood of headline in a word we note that out of the 50 basic characters in Bangla there are 32 characters with headline. From the character occurrence statistics we have noted that out of all the 14 most frequent characters have headline. So, it is likely that most Bangla words will have a headline. We can make a better quantitative analysis of the presence of headline, as follows. According to our computed statistics on first set of data, the average length of Bangla words is about 5 characters. About 30%- 35% of characters are vowel modifiers which being small in size, contribute very little to the headline of the word. Most basic characters are consonants, as vowels in basic form can appear at the beginning of the word or when two vowels appear side by side. But in practical situation, the latter case is very rare. Also, we have seen that compound characters are very infrequent, occurring in 6.17% cases only.

For simplicity, we assume that on an average 4 basic characters only contribute to the headline of the word. We also assume that each character is equally likely in a word. In Bangla only 41 characters can appear in the first position of a word. Out of these 41 characters 30 characters have headlines. Hence, probability of getting a character with headline in the first position of a word is $P_1 = \frac{30}{41}$. Then probability of getting a character without headline in the first position is $1 - P_1 = \frac{11}{41}$. As argued above, the characters which can contribute to the headline in the other positions of a word are mostly consonants. Since 28 out of 39 Bangla consonants have headlines, the probability of getting a consonant with headline for other positions in a word is $P_2 = \frac{28}{39}$. Then probability of getting a character without headline in other positions is $1 - P_2 = \frac{11}{39}$.

Thus, probability of all four characters without headline in a word is $(1 - P_1)(1 - P_2)^3 = 0.00601$ (assuming that all characters are equally likely and independently occurring in a word). Hence, probability that a word will have at least one character with headline is $1 - 0.00601 = 0.99399$.

Out of the character set in Bangla, the top 60 popular characters are presented

in Table-2.2. From the table it can be noted that the modified character a,(া) has the highest occurrence percentage in both set of data. From this table we can say that characters like a,(া); e(ে); R(র); i(ি); K(ক); etc. need more attention for recognition as their occurrence percentage is high. Also, it can be noted that the rank of e(ে) is two in the first set of data whereas it is four in the second set. One of the reasons is that the second set data are the root words which do not contain suffix and most of the suffixes contain the character e(ে). From the tables it can be seen that the compound character /PR/(প্র) takes first rank among all compound characters in both data sets. These global statistics will help us to generate the tree classifier of our OCR system.

Table-2.3 denotes positionwise occurrence of first 60 frequent characters for the first set of data. It may be noted from Table-2.3 that occurrences of the vowel a,(া) occupy first rank in most of the positions in both the data sets. Note that the vowel A(অ) rarely occurs at second and higher positions. Also, it is understood that there is no Bangla word starting with characters a,(া), Y(য়), u(ু), j(্), i,(ী), M,(ং), N*(ণ) etc. as percentage of occurrences in the first position of these characters are zero. These statistics can be used in the error detection module. For examples, if the OCR wrongly recognizes a character as a,(া) in the first position then we know that it is an error because a,(া) cannot occur in the first position.

The occurrence rank of first 60 frequent characters among compound characters are shown in Table-2.4. It is understood that machine recognition of compound characters is more difficult than that of the basic characters and modifiers. These ranks give an idea about which compound characters should be recognized by an efficient OCR system with more accuracy.

It is interesting to note how simple dictionary look-up can be used for error correction in OCR of Bangla. To do so, we have grouped the similar shaped characters. The OCR errors lie mostly within these groups. Now, if a character in a word is misrecognized as another character of its group and if such a character makes

a valid word, then we cannot detect and correct the error by dictionary look-up. Out of the total data set, Table-2.5 shows percentage of valid words obtained by substituting a character by its similar shaped character in the word. It can be noted from the Table-2.5.(a) that if we replace the character P(প) of the words in the first set of data by G(গ) (প and গ are two similar shaped characters) one by one then we will get 1.737% valid words by this substitution. Also, it can be noted from the table that misrecognition of JH(ঝ) as R.(ঋ) cannot generate valid words. Since the percentage of obtaining valid words due to this substitution is small, the dictionary look-up technique can be quite effective in OCR error detection and correction.

The bigram statistics of Bangla characters of both data sets are presented in Table-2.6. Note that the entries which have high ranks in the first set of data are *portions of valid* Bangla suffixes. This happens because Bangla is a highly inflectional language. The trigram statistics of Bangla characters of both data sets are presented in Table-2.7. Here too, the suffix rank is high.

2.5 Conclusion

A statistical analysis of Bangla character occurrence in popular newspapers, books, novels, magazines and juvenile literature as well as dictionary has been reported for the first time. The computed statistics will be helpful in OCR development, Cryptography, Linguistics and efficient Data Base Management. However, our goal in this study is to use some of these statistics for the development of Bangla OCR system.

Table 2.2(a)

First 60 global character occurrence (1st set of data)

Rank	% of occur.	char.	Rank	% of occur.	char.	Rank	% of occur.	char.
1	13.202	া	21	1.170	জ	41	0.316	ঁ
2	11.466	ে	22	1.127	য	42	0.312	ক্ষ
3	7.582	র	23	1.068	ট	43	0.295	ং
4	5.772	ি	24	1.063	ছ	44	0.271	ন্ত
5	4.798	ক	25	1.061	ী	45	0.265	ফ
6	4.767	ন	26	1.005	শ	46	0.238	ঠ
7	3.960	ব	27	0.917	খ	47	0.210	ত্র
8	3.615	ত	28	0.794	ও	48	0.201	ঙ্গ
9	2.906	ল	29	0.775	গ	49	0.187	ঘ
10	2.603	স	30	0.770	চ	50	0.155	ড
11	2.565	ম	31	0.768	ভ	51	0.151	ষ্ট
12	2.376	য়	32	0.665	থ	52	0.149	স্থ
13	2.049	অ	33	0.643	প্র	53	0.132	শু
14	1.957	দ	34	0.579	ধ	54	0.130	ক্ত
15	1.955	প	35	0.510	ং	55	0.122	স্ত
16	1.782	ঁ	36	0.509	উ	56	0.121	শু
17	1.691	হ	37	0.502	ড়	57	0.120	স্ব
18	1.408	ই	38	0.470	ণ	58	0.117	গ্র
19	1.379	এ	39	0.364	ষ	59	0.109	ন্দ
20	1.250	;	40	0.327	'	60	0.102	দ্ধ

Table 2.2(b).

First 60 global character occurrence (2nd set of data).

Rank	% of Occur.	Char.	Rank	% of Occur.	Char.	Rank	% of Occur.	Char.
1	16.366	া	21	1.285	জ	41	0.358	ত্র
2	6.372	ি	22	1.130	ট	42	0.333	থ
3	5.866	ন	23	1.116	গ	43	0.323	ড়
4	5.332	ে	24	0.951	্য	44	0.321	ঙ্গ
5	4.907	র	25	0.886	ভ	45	0.284	ঠ
6	4.498	ক	26	0.810	খ	46	0.283	ন্ত
7	3.899	ত	27	0.795	ধ	47	0.257	ও
8	3.438	ব	28	0.741	ঁ	48	0.229	ঘ
9	3.227	়	29	0.724	ণ	49	0.217	ী
10	2.944	ম	30	0.640	থ্র	50	0.210	ষ্ট
11	2.803	ল	31	0.594	য	51	0.200	ৈ
12	2.735	ী	32	0.545	ু	52	0.197	ক্ত
13	2.681	প	33	0.530	উ	53	0.194	এ
14	2.673	স	34	0.527	ৃ	54	0.187	ন্দ
15	2.023	অ	35	0.442	ষ	55	0.183	ণ্ড
16	1.976	দ	36	0.436	ছ	56	0.179	ত্ত
17	1.514	য়	37	0.432	ং	57	0.174	স্ব
18	1.430	শ	38	0.400	ফ	58	0.172	ণ্ড
19	1.390	চ	39	0.389	ক্ষ	59	0.172	স্ত
20	1.344	হ	40	0.388	ই	60	0.162	ড

Table 2.3

Positionwise occurrence of 60 most frequently occurred characters (1st set of data)

	position 1		position 2		position 3		position 4		position 5		position 6	
Char.	% of occr.	rank	% of occr.	rank	% of occr.	rank	% of occr.	rank	% of occr.	rank	% of occr.	rank
†	0.000		21.011	1	9.775	3	13.859	2	15.937	1	12.846	2
ে	3.987	10	9.880	2	9.972	2	17.044	1	12.295	3	14.004	1
র	1.791	18	5.322	3	10.131	1	6.255	3	12.628	2	12.307	3
ি	10.182	1	3.965	6	6.839	4	3.416	8	4.368	6	3.367	9
ক	7.673	3	5.105	4	2.398	11	4.303	5	2.951	7	3.471	8
ন	3.023	12	3.944	7	4.643	6	5.038	4	6.611	4	5.389	4
ব	6.353	5	4.948	5	3.313	7	2.748	11	2.745	10	2.390	12
ত	3.861	11	2.809	11	2.709	10	4.019	7	4.496	5	5.042	5
ল	0.760	26	2.354	12	4.779	5	4.052	6	2.846	8	3.553	7
স	6.616	4	2.050	13	1.762	14	1.353	17	1.207	18	1.166	18
ম	4.098	9	2.939	10	3.198	8	2.658	12	2.365	12	1.939	14
য়	0.000		1.504	16	3.102	9	3.197	9	2.419	11	3.793	6
অ	9.057	2	0.000		0.006	60	0.004	59	0.011	58	0.006	58
দ	2.207	15	3.242	9	1.112	23	1.240	20	1.461	15	2.719	10
প	5.095	7	1.464	17	1.148	22	1.641	15	1.189	19	1.088	20
ঁ	0.000		3.837	8	1.655	15	2.817	10	0.840	25	1.048	21
হ	4.464	8	1.164	20	0.984	24	0.898	24	0.458	30	0.378	32
ই	0.768	25	1.752	14	1.894	13	0.654	29	1.513	14	0.925	22
এ	6.075	6	0.005	59	0.022	58	0.004	60	0.004	60	0.002	60
্	0.000		1.331	19	2.181	12	1.660	14	0.844	24	1.460	16
জ	2.282	14	0.566	34	1.457	16	1.162	21	1.239	17	0.880	23
য	1.983	16	1.372	18	0.301	42	0.426	34	0.439	31	0.313	35
ট	0.373	34	0.536	35	1.301	19	1.818	13	1.729	13	1.628	15
ছ	0.527	31	0.889	26	0.620	32	1.300	19	2.826	9	2.123	13
ী	0.000		1.006	24	0.567	33	1.462	16	1.093	20	2.485	11
শ	1.083	23	1.062	23	1.156	21	1.323	18	0.903	21	1.137	19
খ	0.772	24	1.068	22	1.372	18	0.910	23	0.334	36	0.306	36
ও	1.666	19	0.122	45	0.641	31	0.534	32	0.666	26	0.570	27
গ	1.100	22	0.952	25	0.671	30	0.930	22	0.555	28	0.740	24
চ	1.37	21	1.108	21	0.761	26	0.893	25	0.860	23	0.649	26

(Contd.)

Table 2.3 (Contd.)

Positionwise occurrence of 60 most frequently occurred characters (1st set of data)

	position 1		position 2		position 3		position 4		position 5		position 6	
Char.	% of occr.	rank	% of occr.	rank	% of occr.	rank	% of occr.	rank	% of occr.	rank	% of occr.	rank
ভ	1.561	20	0.777	30	0.558	34	0.729	27	0.546	29	0.470	28
থ	0.665	28	1.550	15	0.382	39	0.598	31	0.418	33	0.466	29
প্র	2.704	13	0.021	57	0.052	57	0.101	48	0.105	49	0.056	51
ধ	0.742	27	0.731	32	0.723	28	0.670	28	0.599	27	0.344	33
ং	0.000		0.800	29	1.185	20	0.120	47	0.110	47	0.393	31
ঊ	1.870	17	0.073	50	0.277	46	0.020	58	0.028	55	0.049	52
ড়	0.000		0.828	28	0.935	25	0.456	33	0.228	37	0.172	40
ণ	0.000		0.358	39	0.480	36	0.841	26	1.244	16	1.311	17
ষ	0.016	43	0.093	48	0.702	29	0.611	30	0.884	22	0.725	25
'	0.000		0.119	46	1.400	17	0.196	38	0.016	57	0.029	54
়	0.000		0.834	27	0.082	54	0.159	43	0.349	35	0.237	37
ক্ষ	0.327	35	0.387	36	0.352	40	0.291	35	0.184	40	0.120	46
ং	0.000		0.769	31	0.235	47	0.235	37	0.174	42	0.212	38
ত্ত	0.000		0.269	40	0.751	27	0.286	36	0.437	32	0.161	42
ফ	0.546	30	0.380	38	0.159	50	0.123	46	0.102	50	0.085	49
ঠ	0.063	41	0.383	37	0.415	37	0.187	40	0.108	48	0.398	30
ত্র	0.041	42	0.070	51	0.525	35	0.097	49	0.192	39	0.208	39
ঙ্গ	0.000		0.582	33	0.140	51	0.043	55	0.045	53	0.121	45
ঘ	0.577	29	0.114	47	0.096	53	0.063	51	0.151	43	0.019	56
ড	0.255	37	0.162	44	0.056	56	0.193	39	0.387	34	0.320	34
ষ্ট	0.003	44	0.056	52	0.409	38	0.145	44	0.205	38	0.115	47
স্থ	0.182	39	0.015	58	0.284	43	0.186	41	0.055	52	0.040	53
গু	0.210	38	0.027	54	0.108	52	0.132	45	0.177	41	0.142	43
ক্ত	0.000		0.214	42	0.281	44	0.058	52	0.117	46	0.162	41
স্ত	0.129	40	0.217	41	0.342	41	0.026	56	0.087	51	0.085	50
শু	0.424	33	0.024	56	0.077	55	0.021	57	0.007	59	0.015	57
স্ব	0.447	32	0.025	55	0.018	59	0.050	53	0.023	56	0.004	59
গ্র	0.272	36	0.034	53	0.185	49	0.045	54	0.043	54	0.022	55
ন্দ	0.000		0.172	43	0.281	45	0.165	42	0.146	44	0.122	44
দ্ধ	0.000		0.088	49	0.227	48	0.070	50	0.140	45	0.102	48

Table 2.4(a).

Rank of first 60 global compound character's occurrence (1st set of data).

Rank	Char.	Rank	Char.	Rank	Char.	Rank	Char.	Rank	Char.
1	প্র	13	গ্র	25	ন্ন	37	ন্ট	49	স্ম
2	ক্ষ	14	ন্দ	26	দ্ব	38	ল্ল	50	ক্ল
3	ন্ত	15	দ্ধ	27	শ্র	39	ব্র	51	শ্ন
4	ত্র	16	চ্ছ	28	শ্ব	40	শ্চ	52	ন্থ
5	ঙ্গ	17	ক্র	29	স্ব	41	স্প	53	ষ্ক
6	ষ্ট	18	ন্ন	30	ত্ত	42	ষ্ক	54	ল্প
7	স্থ	19	ত্ত	31	ঞ্চ	43	ণ্ড	55	ন্থ
8	গু	20	জ্ঞ	32	ষ্ঠ	44	চ্চ	56	ঙ্গ
9	ক্ত	21	গু	33	স্ক	45	ঞ্জ	57	ড্ড
10	স্ত	22	ম্প	34	ন্ত	46	ত্ম	58	ম্ম
11	গু	23	দ্র	35	স্ট	47	দ্দ	59	ট্ট
12	স্ব	24	হ্ম	36	ব্দ	48	ন্ম	60	জ্ঞ

Table 2.4(b).

Rank of first 60 global compound character's occurrence (2nd set of data).

Rank	Char.	Rank	Char.	Rank	Char.	Rank	Char.	Rank	Char.
1	প্র	13	ন্ত	25	ষ্ক	37	ন্ত	49	দ্দ
2	ক্ষ	14	গু	26	দ্ব	38	ল্ল	50	ক্ক
3	ত্র	15	দ্ধ	27	হ্ম	39	গ্ন	51	ন্ন
4	ঙ্গ	16	ঞ্চ	28	জ্ঞ	40	ভ্র	52	ক্ল
5	ন্ত	17	ক্র	29	শ্ব	41	স্প	53	হ্ম
6	ষ্ট	18	স্থ	30	ঞ্জ	42	স্ক	54	ল্ল
7	ক্ত	19	শ্র	31	ল্ল	43	জ্জ	55	জ্ঞ
8	ন্দ	20	ম্ব	32	চ্ছ	44	হ্ম	56	ন্থ
9	গু	21	দ্র	33	ষ্ঠ	45	ট্ট	57	ড্ড
10	ত্ত	22	গ্র	34	ত্ব	46	স্প	58	হ্ম
11	স্ব	23	গু	35	হ্ম	47	ম্র	59	গ্ন
12	গু	24	ন্ন	36	ত্ম	48	শ্চ	60	ষ্ক

Table 2.5 (a)

Percentage of valid word obtained by replacing a character by its similar shaped (confusion) character in the word (1st set of data).

Original char.	Confused char.	Percentage of valid word	Original char.	Confused char.	Percentage of valid word
প	গ	1.737	ক	ফ	0.475
ল	ন	1.423	ত	ভ	0.422
ঘ	য়	1.286	শ	ণ	0.169
ঘ	ষ	0.857	ষ	য়	0.127
ম	য়	0.718	ষ	ম	0.126
ঘ	য	0.643	ঝ	ঋ	0.000
ম	য	0.612	য	ষ	0.000
খ	থ	0.507	য	য়	0.000
ম	ঘ	0.506			

Table 2.5 (b)

Percentage of valid word obtained by replacing a character by its similar shaped (confusion) character in the word (2nd set of data).

Original char.	Confused char.	Percentage of valid word	Original char.	Confused char.	Percentage of valid word
ল	ন	2.851	ত	ভ	0.588
ঘ	য	2.312	শ	ণ	0.564
প	গ	1.930	ক	ফ	0.435
খ	থ	1.907	য	ষ	0.292
ঘ	য়	1.851	য	য়	0.292
ম	য়	1.559	ষ	ম	0.287
ঘ	ষ	0.925	ষ	য়	0.242
ম	য	0.902	ঝ	ঋ	0.000
ম	ঘ	0.697			

Table 2.6 (a)

First 100 frequently occured bigrams with their ranks (1st set of data).

Rank	Bigram	Rank	Bigram	Rank	Bigram	Rank	Bigram	Rank	Bigram
1	ার	21	বে	41	সা	61	মন	81	শি
2	রে	22	কি	42	াব	62	জন	82	থে
3	রে	23	তি	43	াম	63	াও	83	এব
4	কর	24	ায়	44	ভা	64	খা	84	শে
5	লে	25	য়ি	45	ছে	65	এই	85	রত
6	ান	26	নি	46	হয়	66	খে	86	াপ
7	বা	27	পা	47	কি	67	াজ	87	শি
8	কে	28	ছে	48	হা	68	মে	88	বং
9	না	29	তি	49	টি	69	জা	89	াড
10	তা	30	াক	50	টা	70	অন	90	রণ
11	য়ে	31	সে	51	কে	71	দা	91	বি
12	কা	32	াল	52	সম	72	লে	92	াগ
13	রা	33	রি	53	বল	73	কট	93	খন
14	মা	34	াত	54	যা	74	ুর	94	জে
15	তে	35	দি	55	থা	75	শে	95	খে
16	নি	36	পর	56	াদ	76	লি	96	স ু
17	নে	37	রি	57	াস	77	ীর	97	ধা
18	দে	38	লা	58	াই	78	চা	98	বর
19	নে	39	লি	59	মি	79	াহ	99	বস
20	বি	40	য়া	60	ছি	80	গে	100	দ ু

Table 2.6 (b)

First 100 frequently occured bigrams with their ranks (2nd set of data).

Rank	Bigram	Rank	Bigram	Rank	Bigram	Rank	Bigram	Rank	Bigram
1	ান	21	সা	41	রী	61	খা	81	রণ
2	ার	22	াম	42	ায়	62	মি	82	দি
3	কা	23	নী	43	ভা	63	গা	83	মন
4	তি	24	নি	44	াজ	64	বর	84	র্
5	তা	25	াস	45	শা	65	ধা	85	সি
6	নে	26	াব	46	্ল	66	শি	86	বয়
7	রা	27	্র	47	কর	67	শি	87	্ন
8	বি	28	দা	48	বে	68	দ্	88	কি
9	বা	29	হা	49	স্	69	লি	89	লে
10	মা	30	রি	50	াগ	70	দি	90	রক
11	লা	31	াক	51	ম্	71	রত	91	লী
12	নি	32	ক্	52	সম	72	পি	92	টি
13	াল	33	চা	53	চি	73	ীয়	93	াণ
14	কি	34	াঁ	54	অন	74	াশ	94	বন
15	না	35	াদ	55	্ত	75	হি	95	ত্
16	পা	36	াপ	56	াহ	76	্ত	96	াড
17	াত	37	টা	57	্ক	77	াট	97	দে
18	তি	38	পর	58	মি	78	ন্	98	্র
19	রি	39	লি	59	প্	79	পি	99	াচ
20	যা	40	জা	60	বি	80	সি	100	সং

Table 2.7(a)

First 75 frequently occured trigrams with their ranks (1st set of data).

Rank	Trigram	Rank	Trigram	Rank	Trigram	Rank	Trigram	Rank	Trigram
1	করে	16	ানে	31	ওয়া	46	খান	61	বেশ
2	দের	17	মান	32	কিন	47	বিশ	62	কাল
3	কার	18	দেখ	33	রতে	48	দিন	63	দেশ
4	ছেে	19	ররে	34	লিে	49	ানা	64	দিক
5	তার	20	কার	35	নিয়	50	করত	65	শেষ
6	বার	21	য়েছ	36	নি	51	কটা	66	কিছ
7	ছিল	22	ভাব	37	বলে	52	করি	67	হবে
8	লেন	23	থেক	38	াতে	53	াকা	68	াছে
9	নের	24	াবে	39	ারণ	54	সেই	69	াহা
10	পার	25	ারা	40	কথা	55	নির	70	য়ের
11	কেে	26	কো	41	ছেন	56	দিে	71	ারি
12	একি	27	ামা	42	তিন	57	নার	72	জান
13	াকে	28	পরি	43	হার	58	মনে	73	সার
14	হয়ে	29	তছি	44	কটি	59	লেে	74	ভার
15	এবং	30	থাক	45	লের	60	রাজ	75	াছে

Table 2.7(b)

First 75 frequently occured trigrams with their ranks (2nd set of data).

Rank	Trigram	Rank	Trigram	Rank	Trigram	Rank	Trigram	Rank	Trigram
1	ানে	16	ানা	31	অনু	46	কুল	61	খান
2	কার	17	ারা	32	বিদ	47	নীয়	62	রিক
3	কান	18	দার	33	াতা	48	াতি	63	বিক
4	ারি	19	ওয়া	34	চান	49	পুর	64	াদা
5	মান	20	াকা	35	রিত	50	জ্ঞান	65	মহা
6	লান	21	ামা	36	বান	51	হীন	66	াপা
7	ারী	22	কাল	37	দুর	52	দান	67	মাত
8	রান	23	চার	38	ায়া	53	পাল	68	জীব
9	নির	24	রাজ	39	ানী	54	ধার	69	ভাব
10	যান	25	হার	40	নান	55	মুখ	70	লো
11	ালা	26	পান	41	সান	56	চিত	71	ানু
12	টান	27	বাদ	42	সার	57	রিণ	72	মাল
13	পরি	28	কৃত	43	বার	58	পুর	73	াটা
14	ানি	29	াবা	44	মার	59	বাহ	74	াগা
15	ালি	30	তান	45	পাত	60	হিত	75	সমা

Chapter 3

PREPROCESSING AS WELL AS LINE, WORD AND CHARACTER SEGMENTATION

3.1 Introduction

Preprocessing is an important phase of the OCR system. This phase takes the raw image as an input and produces processed image as an output which is easier for further handling. The main preprocessing stages of our OCR model are : Binarization, smoothing, skew angle estimation and correction, text and graphics separation, and line, word and character segmentation.

Binarization is a process by which the foreground object is separated from its background. It transforms the image to a two-tone one where the object and background pixels are usually represented by 1 and 0, respectively.

Smoothing performs filling and cleaning of the image. Filling eliminates small breaks, gaps and holes. Cleaning takes care of the noise in the image.

When a document is fed to the optical sensor either mechanically or by a human operator for image acquisition, a few degrees of skew (tilt) is unavoidable. Skew angle is the angle that the text lines in the digital image make with the horizontal direction. Skew angle estimation and correction are important preprocessing steps of document layout analysis and OCR systems. There exist various skew detection approaches [3,5,66,92,122,181] which are briefly reviewed in Section 3.4. We have developed a new approach that is suitable for Indian scripts like Bangla and Devnagari. This approach is also described in Section 3.4.

For a document containing multiple columns of text lines as well as graphics and halftone pictures, it is necessary to separate the text blocks before feeding it for the recognition. There exist various techniques of text block segmentation [79,80,116]. One approach is based on the use of Gabor filterbanks [72] where the response of the filterbank in text regions is different from that in graphics regions. Another approach is based on the 'docstrum' analysis of the document [115]. Fletcher and Kasturi [51] applied Hough transform for text string separation from mixed text/graphics images. Pavlidis and Zhou [122] used a correlation based technique to segment the text regions as well as to detect the logical text blocks. Wang *et al.* [173] performed a non-linear run-length smoothing technique for the text/graphics separation. We have used the last mentioned technique whenever necessary and obtained satisfactory results.

In our approach the two tone image is subject to further processing stages namely line, word and character segmentations. While line and word segmentations are done by projection profile approach, the character segmentation is done by deletion of the headlines from the middle zone of words. For the convenience of the recognition algorithm, a line of Bangla text is partitioned into three zones (shown in Fig.2.4).

This chapter is organized as follows. An overview of text digitization and image data collection are presented in Section 3.2. Grey tone to two tone conversion (thresholding) and smoothing are described in Section 3.3. Section 3.4 deals with the skew detection and correction techniques. Script line segmentation and zone detection aspects are presented in Section 3.5. Also, word segmentation from line and character segmentation from word have been elaborated in this section.

3.2 Text digitization and image data collection

3.2.1 Text digitization

For script recognition and document processing there exist a number of input devices such as the tablet and stylus, the hand- held scanner, sheet-fed scanner, overhead scanner, flatbed scanner etc. [106]. In some applications, such as in interactive computer graphics, the tablet and stylus as well as light pen digitizer are used.

The tablet digitizer is a two-level digitizer, where the position of a point on the tablet is sensed by a coil in the stylus and the co-ordinates are transferred to the processor. Hand-held scanners typically have lower resolution ranges. The sheet-fed scanners resemble a FAX machine in operation. Rollers pull the document from the feeder bin and move it past the stationary scan head. In overhead scanners a camera like optical mechanism passes above the document which is face up on an open air bed and hence good quality image may not be obtained.

We have used a flatbed scanner, the most common type used for OCR, (manufactured by Hewlett Packard, Model ScanJet 4C) for our digitization purpose. This is because the output of hand-held scanner is affected by hand movement creating local fluctuations and the mechanical motion of the original document during scanning by sheet-fed scanner can easily cause it to be misaligned and result in

skewed scan lines. Flatbed scanners, on the other hand, scans the document uniformly. Also, most commercial OCRs have provision for flatbed scanning input device [111]. The resolution of the flatbed scanner varies from 100 to 900 dots per inch (dpi).

3.2.2 Image data collection

Nowadays various Desk Top Publishing (DTP) software company are developing packages with different font names. Some of the available fonts for Bangla are Lino, Geetanjali, Vivek, Satyajit, Bankim, Anand, Sidhartha, Prakash etc. But most of the popular Bangla newspapers, magazines and printed books follow the Lino font. Our government presses also follow the Lino font. Geetanjali is the most recent font supplied by Abacus company. See Fig.3.1, where examples of texts in different fonts are given. The point size of these text is 16. (1 inch = 72 points). We can note from Fig.3.1 that there is negligible difference between Lino and Geetanjali fonts. We have chosen Lino font images for our experiment.

We have gathered several pages of Lino font text from books, newspapers, juvenile literatures etc. for testing our system. These pages contain approximately 20,000 words (equivalent to about 91,000 characters).

3.3 Grey tone to two tone conversion and smoothing

3.3.1 Grey tone to two tone conversion

The main purpose of converting grey tone images into two tone binary images is to make the processing easier. This process of conversion is done by thresholding the grey value of each pixel. Global, local and dynamic thresholding are three types of approaches for thresholding.

তখন শিক্ষক কহিলেন, তোমার এই কথা শুনিয়া সন্তুষ্ট হইলাম। কিন্তু তুমি, যে পাখীকে লক্ষ্য করিয়া ডেলা ছুড়িয়াছিলে, উহার গায়ে ঐ ডেলা লাগে নাই। নিকটে একটি বালক দাঁড়াইয়া ছিল, ডেলা তাহার মাথায় লাগিয়া রক্তপাত হইয়াছে। চক্ষুতে লাগিলে সে এ জন্মের মত অন্ধ হইয়া যাইত। বালকটি কাতর হইয়া কত রোদন করিতেছে। অতএব দেখ, ডেলা ছোড়ায় কত দোষ। (a)

তখন শিক্ষক কহিলেন, তোমার এই কথা শুনিয়া সন্তুষ্ট হইলাম। কিন্তু তুমি, যে পাখীকে লক্ষ করিয়া ডেলা ছুড়িয়াছিলে, উহার গায়ে ঐ ডেলা লাগে নাই। নিকটে একটি বালক দাঁড়াইয়া ছিল, ডেলা তাহার মাথায় লাগিয়া রক্তপাত হইয়াছে। চক্ষুতে লাগিলে সে এ জন্মের মত অন্ধ হইয়া যাইত। বালকটি কাতর হইয়া কত রোদন করিতেছে। অতএব দেখ, ডেলা ছোড়ায় কত দোষ।— (b)

তখন শিক্ষক কহিলেন, তোমার এই কথা শুনিয়া সন্তুষ্ট হইলাম। কিন্তু তুমি, যে পাখীকে লক্ষ্য করিয়া ডেলা ছুড়িয়াছিলে, উহার গায়ে ঐ ডেলা লাগে নাই। নিকটে একটি বালক দাঁড়াইয়া ছিল, ডেলা তাহার মাথায় লাগিয়া রক্তপাত হইয়াছে। চক্ষুতে লাগিলে সে এ জন্মের মত অন্ধ হইয়া যাইত। বালকটি কাতর হইয়া কত রোদন করিতেছে। অতএব দেখ, ডেলা ছোঁড়ায় কত দোষ। (c)

তখন শিক্ষক কহিলেন, তোমার এই কথা শুনিয়া সন্তুষ্ট হইলাম। কিন্তু তুমি, যে পাখীকে লক্ষ্য করিয়া ডেলা ছুড়িয়াছিলে, উহার গায়ে ঐ ডেলা লাগে নাই। নিকটে একটি বালক দাঁড়াইয়া ছিল, ডেলা তাহার মাথায় লাগিয়া রক্তপাত হইয়াছে। চক্ষুতে লাগিলে সে এ জন্মের মত অন্ধ হইয়া যাইত। বালকটি কাতর হইয়া কত রোদন করিতেছে। অতএব দেখ, ডেলা ছোড়ায় কত দোষ। (d)

তখন শিক্ষক কহিলেন, তোমার এই কথা শুনিয়া সন্তুষ্ট হইলাম। কিন্তু তুমি, যে পাখীকে লক্ষ্য করিয়া ডেলা ছুড়িয়াছিলে, উহার গায়ে ঐ ডেলা লাগে নাই। নিকটে একটি বালক দাঁড়াইয়া ছিল, ডেলা তাহার মাথায় লগিয়া রক্তপাত হইয়াছে। চক্ষুতে লাগিলে সে এ জন্মের মত অন্ধ হইয়া যাইত। বালকটি কাতর হইয়া কত রোদন করিতেছে। অতএব দেখ, ডেলা ছোড়ায় কত দোষ। (e)

Figure.3.1: Examples of Bangla text in different fonts. (a) Lino (b) Geetanjali (c) Vivek (d) Satyajit (e) Bankim.

In its simplest form, a threshold selection involves choosing a gray level 't' in such a way that any gray level less than 't' is mapped into 'object' label. Otherwise a grey level is mapped into the 'background' label. Such a threshold is a constant or global threshold applied everywhere in the image.

In a more general form, a threshold operator can be viewed as a test involving a function T in the form T(x,y, N(x,y), g(x,y))

where g(x,y) is the gray level of the image point (x,y) and N(x,y) denotes some local properties of the point (x,y), e.g., the average gray level over some neighborhood. For each point (x,y) in the image, if $g(x,y) < T(x,y,N(x,y),g(x,y))$ then (x,y) is labeled as an object pixel; otherwise, (x,y) is labeled as a background pixel [179].

When T depends only on g(x,y), the threshold may be called *global.* If T depends on both g(x,y) and N(x,y), it is *local* threshold. If T depends on the coordinate values (x,y) as well as on g(x,y) and N(x,y), then the process may be called *dynamic* thresholding.

In local thresholding, the original image is partitioned into smaller subimages and a threshold is determined for each of the subimages. This technique gives a threshold image with grey level discontinuities at the boundary of two different subimages.

A global thresholding technique is one that thresholds the entire image with a single threshold value. Global thresholding techniques can be of two types: point dependent and region dependent techniques. If the threshold value is determined solely from the gray level of each pixel, the thresholding method is point-dependent. If the threshold value is determined from the local property (e.g., the local gray level distribution) in the neighborhood of each pixel, the thresholding method is region- dependent. P-tile method, mode method, the method due to Ostu, histogram concavity analysis method, entropy method, moment preserving method and minimum error method are point dependent techniques while histogram trans-

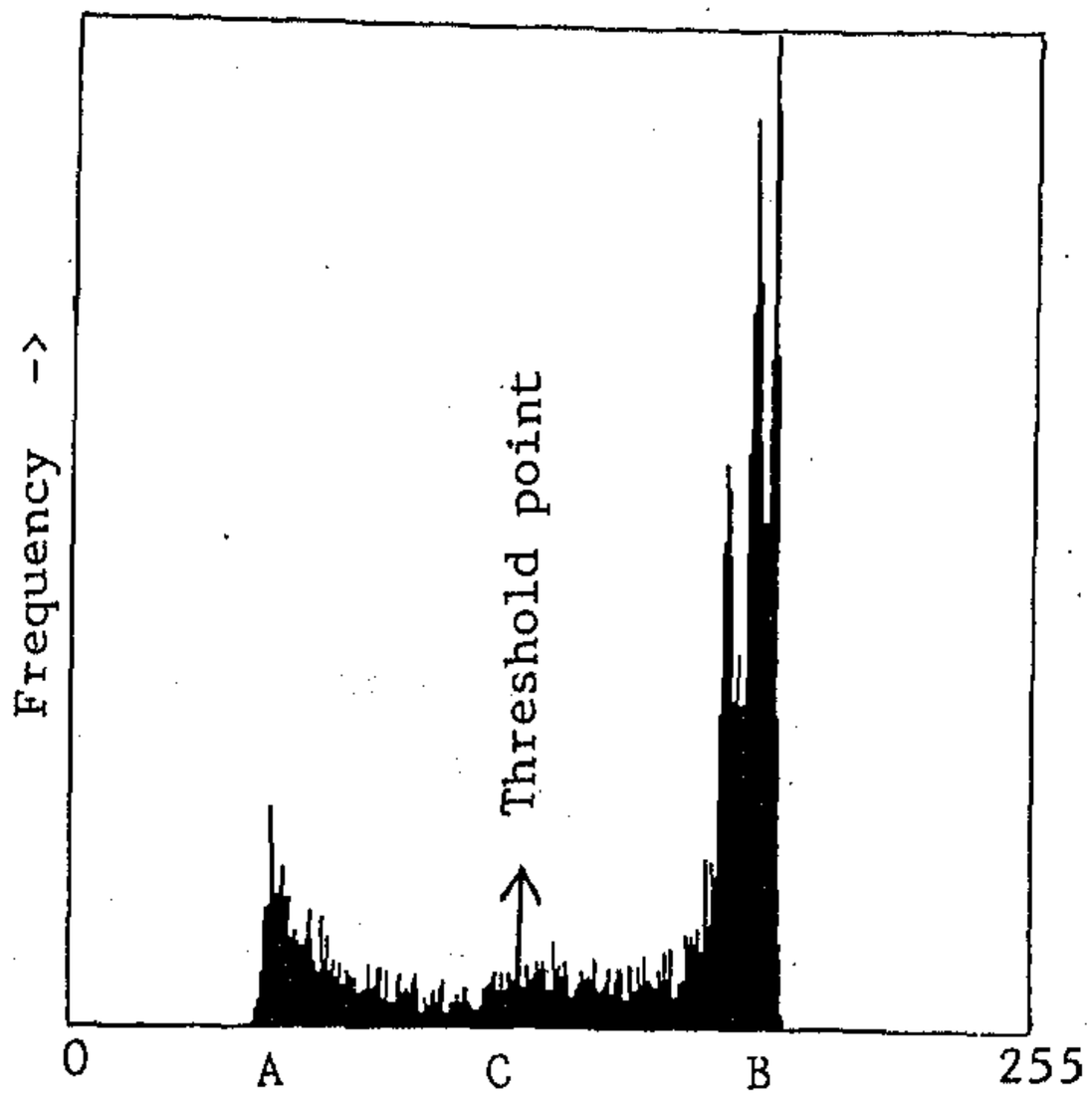
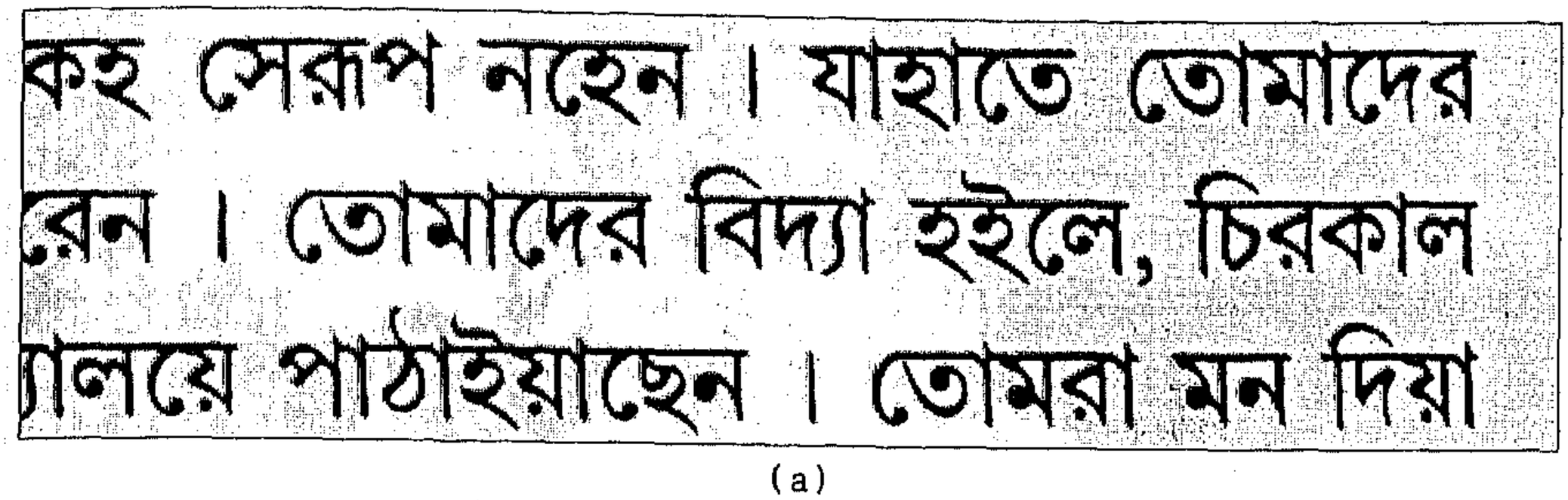


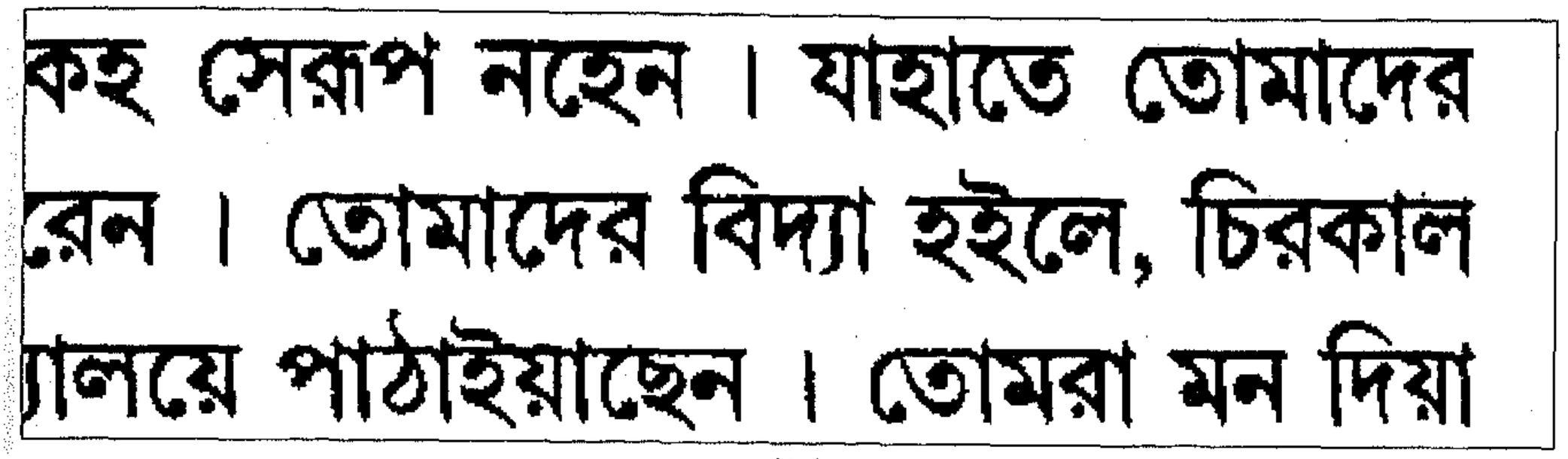
Figure 3.2: A typical histogram for threshold selection.

formation method, method based on second order grey level statistics, and gradient relaxation method are region dependent thresholding techniques [81,179].

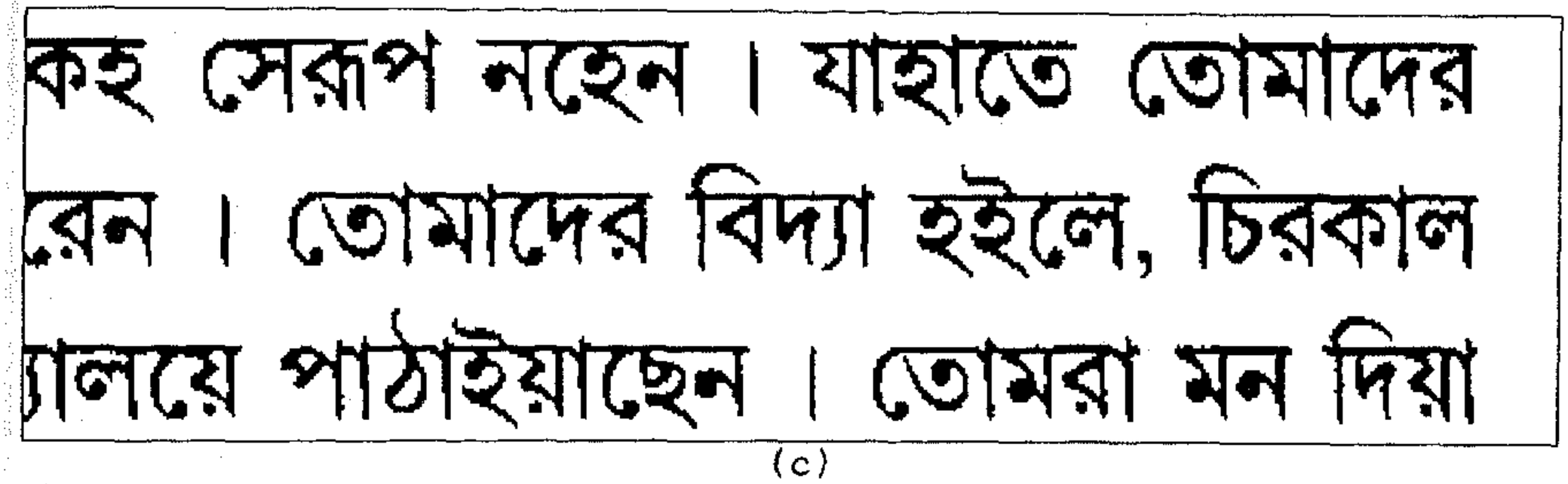
For images with distinct object and background regions the histogram of the grey levels will be bimodal. The grey level corresponding to the valley of the histogram can be selected as a threshold value. In our case we have chosen threshold value as the midpoint of the two histogram peaks. For example see Fig.3.2. Here, A and B are two peaks of object and background regions of a document. C, the mid-point of AB, is chosen as a threshold value. This threshold value has been chosen at the training phase by observing a large number of images. The two-tone image is represented by 0-1 labels where the label 1 represents the darker pixels (objects). For a comparison the output obtained by Ostu's [117] method and the method due to Kapur *et al.* [81] are shown in Fig.3.3. It can be noted that our simple approach gives result similar to or better than these methods.



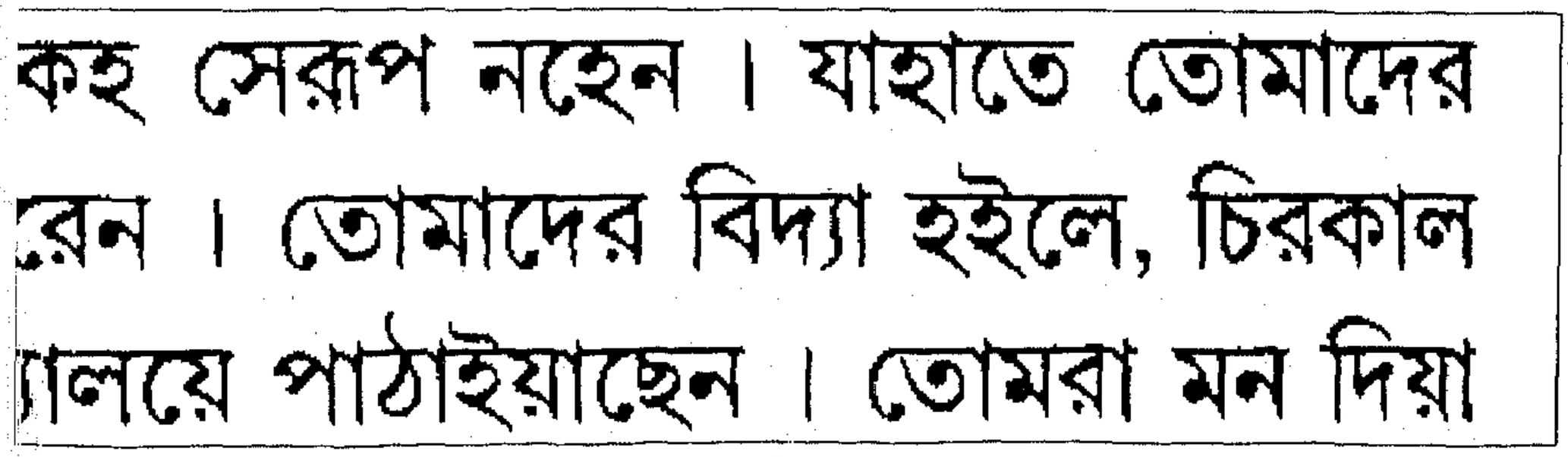
(a)



(b)



(c)



(d)

Figure 3.3: Results of different thresholding methods. (a) Original image (b) Method due to Kapur *et al.* (c) Method due to Ostu (d) Our method.

3.3.2 Smoothing

The digitized image may contain spurious noise pixels and add irregularities to the outer boundary of the characters and may have undesired effects on the recognition system. For removing these noise pixels we have used a simple and efficient technique due to Mahmoud [96]. This technique reduces the noise of the binary image by eliminating small areas and filling little holes, that results in the regularization of the character contour. For a given text, the algorithm modifies each pixel according to its initial value and to those of its neighborhood. Pixel modification rule is as follows.

If $P_0 = 0$ then

$$P'_0 = \begin{cases} 0 & : \quad if \quad \sum_{i=1}^{8} P_i < T \\ 1 & : \quad otherwise \end{cases}$$

else

$$P'_0 = \begin{cases} 1 & : \quad if \quad P_i + P_{i+1} = 2 \quad for\ at\ least\ one\ i = 1, ...7 \\ 0 & : \quad otherwise \end{cases}$$

where P_0 is the current pixel value, P'_0 is the modified pixel value and T is the threshold.

Table 3.1: The current pixel P_0 and its neighbors.

P_4	P_3	P_2
P_5	P_0	P_1
P_6	P_7	P_8

We have noted that a threshold of value 5 gives acceptable results. A lower threshold results in filling characters holes and concave boundaries consequently

changing the topology of the characters. Higher thresholds result in a very little or no smoothing. The labeling scheme of these pixels is shown in Table-3.1.

3.4 Skew estimation and correction

Skew correction can be achieved in two steps namely (a) estimation of skew angle θ_s and (b) correction of skew by rotating the image in an opposite direction by θ_s. so that script lines becomes horizontal. Here, we discuss about an efficient and robust skew detection technique which is suitable for the most of the Indian languages. Before going to the actual methodology of skew correction, a brief review of the earlier works is presented here.

3.4.1 A brief review of earlier works

Some popular skew estimation techniques are based on the projection profile of the documents. The horizontal/vertical projection profile of a document [68] is a histogram of the number of black pixels along horizontal/vertical scan lines. The projection profile is computed at a number of angles and for each angle, a measure of difference of peak and trough height of the profile is made. The maximum difference corresponds to the best alignment with the text line direction which, in turn, determines the skew angle. Baird [5] proposed a modification for quick convergence of this iterative approach. Akiyama and Hagita [3] described a fast and less accurate approach where the document is partitioned into vertical strips. The horizontal projection profiles are calculated for each strip and the skew angle is determined from the correlation of the profiles of the neighboring strips. Pavlidis and Zhou [122] proposed a method based on vertical projection profile of horizontal strips. Another method using cross correlation between the lines at a fixed distance has been proposed by Yan [181].

Techniques based on Hough transform are also popular for skew estimation

[66,92,154]. Modifications of Hough transform based approach are proposed to discard the irrelevant data pixels and to reduce the amount of data so that less computation is necessary. For example, Hinds *et al.* [66] made some modifications based on run length of black pixels to reduce the amount of data to be processed by Hough transform. Le *et al.* [92] found connected components in a documents and considered only bottom pixels of each component for Hough transform. Their method reduces the data due to consideration of bottom pixels only. An improvement of this approach is proposed by Pal and Chaudhuri [118]. One general limitation of Hough transform is that correct identification of peak in Hough space may be difficult for sparse text.

Fourier transform has also been used for the detection of skew angle. Postl [126] proposed an approach where the direction for which the density of Fourier transform is largest gives the estimated skew.

Another class of approaches is based on component nearest neighbor clustering. Hashizume *et al.* [64] proposed nearest neighbor clustering for skew detection. They found all the connected components in the document and for each component they computed the direction of its nearest neighbor. A histogram of the direction angle is computed, the peak of which indicates the document skew angle. O'Gorman [115] generalized the approach in so called 'docstrum' analysis. In principle, these approaches are not limited to any range of skew angle.

3.4.2 The proposed technique

It should be pointed out that the success of some of the above methods depends upon the characteristics of the alphabet and text lines of a particular script. For example, the component nearest neighbor clustering methods [64,115] will not work well for documents containing Bangla script. This is so because the characters in a word of this script get connected through headlines and a complete word can be a single component unlike in English (Roman) where usually each character is

a separate component. Since word length can vary to a great extent, the nearest neighbor of a word (computed in terms of centroid) may not be another word to its left or right and thus the direction of the neighbor may not indicate the skew angle properly.

As another example, consider the modified Hough transform method due to Le *et al.* [92]. This technique is also unsuitable for Bangla script because of the connecting nature of characters in a word. A few bottom pixels of words will make the data very sparse and the peak of the Hough transform accumulator will be flattened.

The above examples suggest that it may be possible to find skew angle easily and quickly if some inherent characteristics of the script form are employed. Our skew detection is based on the headlines of the characters. From a statistical analysis (discussed in chapter 2) we have found that the probability that a Bangla word will have at least one character with headline is 0.99399. This high probability implies that we can use headline for the skew detection purpose.

Since the characters in a word are usually connected through a headline, a complete word in a (skewed or unskewed) document can often be detected by the method of connected component labeling [24]. For skew angle detection at first the connected component labeling in the document is carried out and the bounding box (minimum upright rectangle containing the component) of each labeled component is defined. The mean b_m and standard deviation b_s of the bounding box width are also computed. Next, components having bounding box width greater than or equal to b_m and less than b_m+3b_s are selected. By thresholding at b_m the small components like dots, punctuation marks, isolated characters and characters without headline etc., are mostly filtered out. By thresholding at b_m+3b_s big components that may represent graphs and tables are also filtered out. Because of these filtering processes the irrelevant components cannot create error in skew estimation. Also, longer headlines of the words are often retained by this process. See Fig.3.4(b) for

the selected components of Fig.3.4(a), obtained using this approach.

Next, we find the *upper envelope* of the selected components. Consider a component with label G. From each pixel of the uppermost row of the component bounding box, we perform a vertical scan and as soon as the pixel labeled G is encountered, we convert its label to U. The set of pixels labeled U obtained in this way denotes the upper envelope of the component. See Fig.3.4(c) where all the upper envelopes of components of Fig.3.4(b) are shown. From Fig.3.4(c), we note that in most of the cases the upper envelope contains the headline. In this way we could filter out the irrelevant data for further processing. Hough transform technique may be applied on the upper envelopes for skew estimation, but this is a slow process. We have used the following approach which is fast, accurate and robust.

The idea is based on the detection of *Digital Straight Line* (DSL) [135] segments from the upper envelope. Although the upper envelope image contains headline information, it may contain non-linear parts due to some modified and basic characters. To delete these non-linear parts, we find and select DSL segments in the upper envelope image. A DSL is a digital arc which results from the digitization of a straight line segment. There are two ways of digital arc representation namely, *crack code* and *chain code*. In a chain code representation, two digital arc pixels can be 8-neighbors while in crack code representation two digital arc pixels can be 4-neighbors. We consider chain code representation of the upper envelope digital arcs.

The following conditions are necessary for straightness of a chain code digital arc [136].

1 There exist runs of pixels in at most two directions which differ by 45 deg.

2 For runs of two directions, the run lengths in one of the directions is always one.

Figure 3.4: Skew angle detection approach. (a) An example of skewed image. (b) Selected components of (a). (c) Upper envelopes of the components of (b). (d) SDSL components of (c).

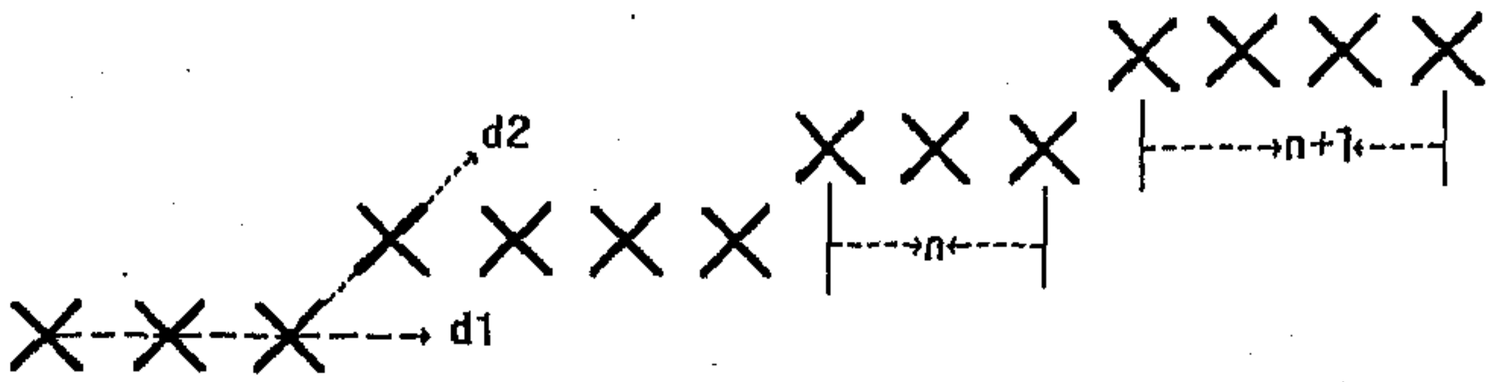


Figure 3.5: An example of a digital straight line (DSL).

3 The run length in the other direction can have at most two values differing by unity (e.g. n and n+1).

An example of a DSL is shown in Fig.3.5. Here, the angle between two directions d_1 and d_2 is 45 deg and run length in the direction d_2 is one whereas run length in d_1 direction is two (n) or three (n+1), occurring alternately.

It should be noted that a digital arc satisfying the above three conditions may not always represent a digital straight line. The arc should satisfy the chord property for being a DSL [135]. Arcs satisfying the above three properties belong to a superset of DSL and may be called SDSL. We considered SDSLs only because they are easy to detect and they represent straightness almost always in the data considered.

Note that better estimates of skew are obtained if the envelopes are long so that they have long headline regions. Hence, from upper envelope of each component, we can find the largest digital arc which satisfy the three conditions stated above. These SDSL components are considered for skew detection (see Fig.3.4(d)). Let the set of such components be R_1 and let the largest SDSL of all members of R_1 be C_L. For example, see Fig.3.4(d). Rejection of other arcs results in a kind of noise cleaning.

If we take average of angles with the horizontal direction, obtained from the lines joining the first and last pixel of every SDSLs, we get an approximate estimate of the skew angle. To get a better estimate we have clustered the members of R_1 corresponding to individual text lines. The accurate estimate of skew angle is obtained by averaging of angles between the horizontal line and the lines joining the first pixel of the leftmost SDSL and the last pixel of the rightmost SDSL of each individual text line (cluster).

The clustering is done as follows. From C_L or its continuation find the normal distances to a reference (say, leftmost) pixel of other SDSLs in R_1. The SDSLs of a particular text line will have nearly equal normal distances, and hence they can be clustered in terms of their normal distances. The co-ordinates of leftmost (x_{lc}, y_{lc}) and rightmost (x_{rc}, y_{rc}) pixels of each cluster C are also maintained for future convenience. Actually, (x_{lc}, y_{lc}) and (x_{rc}, y_{rc}) are the co-ordinates of leftmost pixel of leftmost SDSL and rightmost pixel of rightmost SDSL of the cluster C.

The above approach may be viewed as similar to filling of parametric space bins in the Hough transform. A bin corresponds to SDSLs having nearly equal normal distance defined above. Thus, for a SDSL N_1 of R_1, its normal distance of leftmost pixel to C_L is computed and if this distance is within $\pm H/2$ of normal distance of the leftmost pixel of the leftmost SDSL of any existing bin to C_L then N_1 is placed in that bin (H is the minimum possible distance between two text lines any document can have. Way of estimating H is given in the next paragraph). Otherwise, a new bin is created and N_1 is placed in the new bin. At the same time, the co-ordinates of the leftmost and rightmost pixels of each cluster are updated accordingly. For example, suppose a new SDSL q enters an existing cluster C. If the leftmost pixel (x_q, y_q) of q is to the left of the existing leftmost pixel (x_{lc}, y_{lc}) of this cluster i.e., if $x_q < x_{lc}$ then we make $x_{lc} \leftarrow x_q$, $y_{lc} \leftarrow y_q$. Else, no modification is done. The rightmost pixel of C is treated accordingly. The whole procedure can be executed in single scan over R_1. At the end, K bins and hence K clusters are usually produced if there are K text lines in the document. From each cluster C the

angle of the line joining (x_{lc}, y_{lc}) and (x_{rc}, y_{rc}) with horizontal direction is found. Average of such angles over all text lines gives an accurate estimation of the skew angle.

The value of H can be estimated as follows. For most documents, the character size is not smaller than 6 points. Then, minimum spacing between two headlines is 12 points. If the document is digitized at P DPI then the minimum distance (H) between the headlines of two text lines is $P \times 12/72$ pixels = P/6 pixels. (Since 72 points = 1 inch). For document digitized at 300 DPI we have H = 50 pixels.

In brief, our skew detection algorithm has the following steps:

Algorithm SKEW-EST :

Step 1: Find connected components in the binary document image and find the mean b_m and standard deviation b_s of their bounding box widths.

Step 2: Choose the set S of connected components having bounding box width greater than or equal to b_m and less than $b_m + 3b_s$.

Step 3: For each component in S find the upper envelope described above. From each envelope component, find the SDSLs. If more than one SDSL is found, choose only the longest one and form the subset R_1. Let the longest SDSL in R_1 be C_L.

Step 4: From the line C_L or its continuation find the normal distances to the leftmost pixel of other SDSLs of R_1.

Step 5: Cluster the SDSLs of R_1 corresponding to individual text lines and find the leftmost and rightmost pixel of each cluster, as described above.

Step 6: For each cluster find the angle of line joining the leftmost and rightmost pixels (e.g., A and B in Fig.3.4(d)) with horizontal direction.

Step 7: Find the average of such angles over all clusters (e.g., text lines). The average angle (θ_s) gives an accurate estimate of the skew angle.

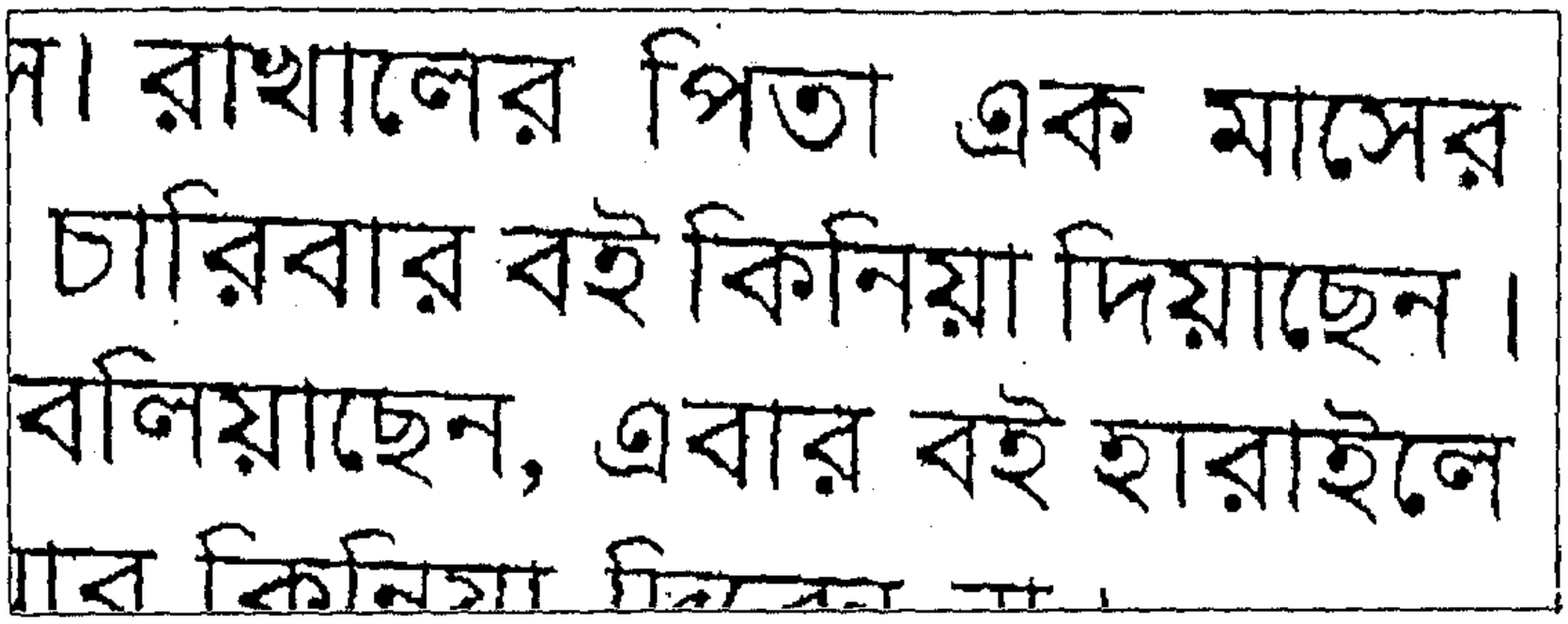
The image is then rotated by θ_s in opposite direction so that the script lines become horizontal. This is done by simple rotation transforms and truncation to the nearest integer value to get the new co-ordinates. See Fig.3.6 where a skewed image and its deskewed image are shown.

3.4.3 Results and discussion

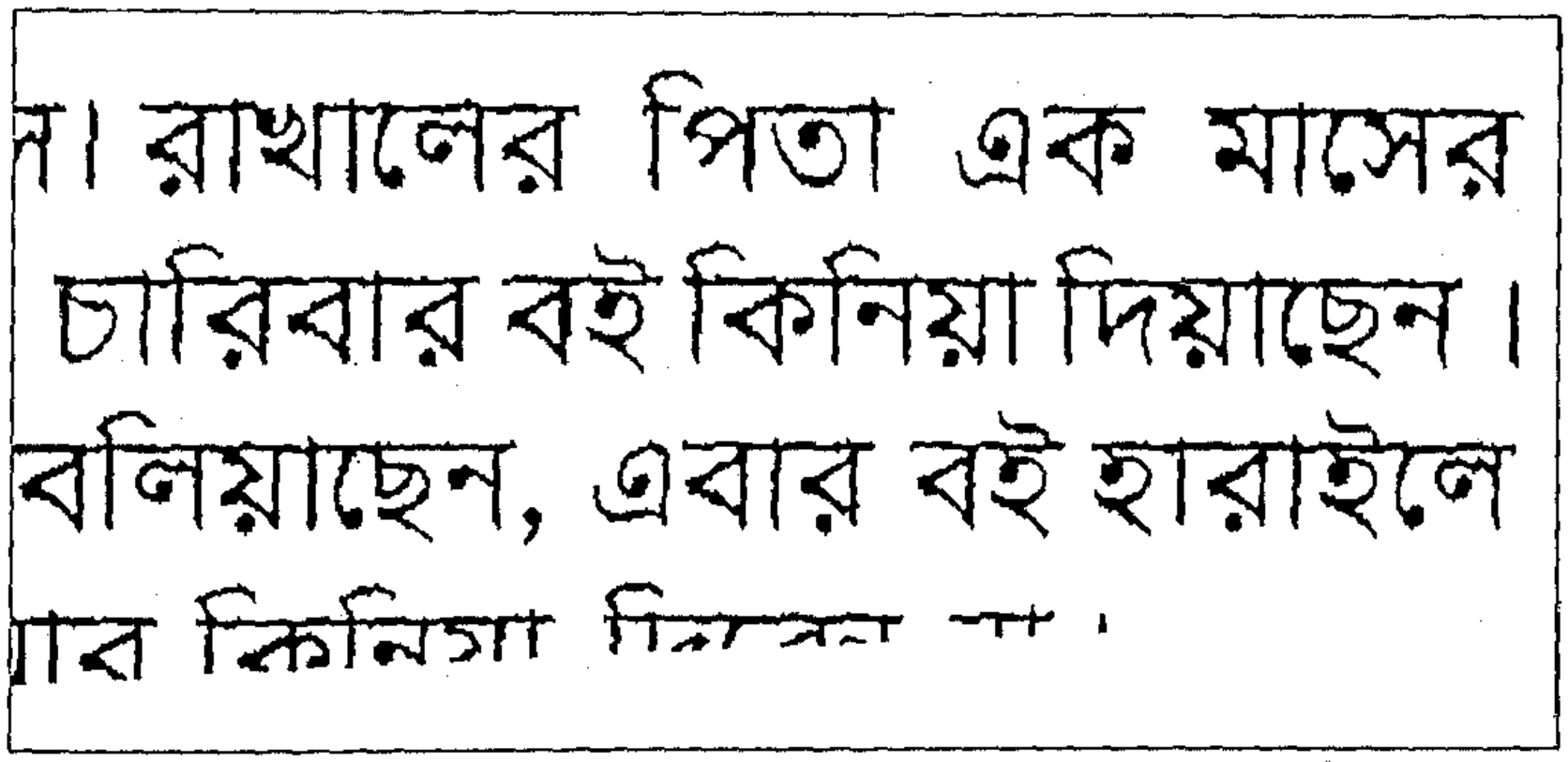
For the experiment we have considered one hundred document images from different books, magazines and newspapers. Some documents contained graphics, tables etc. They were digitized by flatbed scanner at a resolution of 300 DPI. The documents were skewed by a prespecified angle ranging between 0 and ± 45 deg. This angle is considered as true skew angle. As a comparative study, we have estimated skew angle using Hough transform over the original documents as well as using Hough transform over SDSL of upper envelope. Typical results are shown in Table-3.2.

To test the time efficiency of our method (A_3) with respect to conventional Hough transform method on original document image (A_1) as well as on SDSL (A_2), we computed the time of execution in these methods. The angular resolution used in Hough transform was 1 deg. The average execution times for a document of 512×512 pixels on a SUN 3/60 (with Microprocessor MC68020, and SUN OS version 3.0) machine are 620, 312 and 17.80 seconds for methods A_1, A_2 and A_3, respectively. Note that, Hough transform execution time would be higher if finer angular resolution (say 0.5 deg) is considered.

To test the skew angle estimation efficiency of the methods mentioned above, we have done a statistical testing of hypothesis using F-distribution obtained as follows. Let X_1, X_2, ...X_n be n independent observations from a Gaussian distri-



(a)



(b)

Figure 3.6: Example of an image (a) before skew correction. (b) after skew correction.

bution with mean ϕ and standard deviation σ i.e.,

$$\sum_{i=1}^{n} \frac{(X_i - \phi)^2}{\sigma^2} \sim \chi^2_n.$$

For independent true skew angles Y_j (j=1, 2 ..n_1) let X_{ji} and Z_{ji} (i=1, 2, ...n_2) be the independent estimated angles for i-th observation by methods A_3 and A_2, respectively. We assume X_{ji} and Z_{ji} follow Gaussian distributions with standard deviations σ_1 and σ_2, respectively. Now,

$$T1 \triangleq \frac{1}{\sigma_1^2} \sum_{j=1}^{n_1} \sum_{i=1}^{n_2} (X_{ji} - Y_j)^2 \sim \chi^2_{n_0}$$

and

$$T2 \triangleq \frac{1}{\sigma_2^2} \sum_{j=1}^{n_1} \sum_{i=1}^{n_2} (Z_{ji} - Y_j)^2 \sim \chi^2_{n_0}$$

Where $n_0 = n_1 \times n_2$.

Our null hypothesis is $H_0 : \sigma_1 = \sigma_2$ under which

$$T \triangleq \frac{T_1}{T_2} \sim F_{n_0,\, n_0}.$$

Where $F_{n_0,\, n_0}$ is F-distribution with n_0 and n_0 degrees of freedom.

From our estimated skew angles the value of T is found as 1.24, where $n_1 = 5$ and $n_2 = 20$. The 5% cut off point of the F-distribution (with degrees of freedom 100 and 100) is 1.41 (obtained from F-distribution table through interpolation). Hence, the methods A_2 and A_3 are statistically equally accurate. However, as pointed out before, method A_3 takes less execution time.

Font and size variations do not affect the proposed skew detection and correction method since the presence of headline is independent of character font and size. Note that the approach is not limited to any range of skew angle.

One additional advantage of the proposed method is that the headline is readily detected during skew estimation. The characters in a word can be separated if

the headline region is rubbed off (character segmentation procedure is discussed in the next section). The separated characters are then subjected to the OCR recognition procedure. Moreover, detection of headline separates middle zone from upper zone of a text line automatically, thus helping in zone detection. Since our method is based on connected component labeling, document structure analysis can also be readily done from the skew corrected output.

3.5 Line and zone detection, word and character segmentation

For the convenience of recognition the OCR system should automatically detect individual text lines as well as segment the words as well as the characters accurately. Since Bangla text lines can be partitioned in three zones, it is convenient to distinguish these zones. Character recognition becomes easier if the zones are distinguished because the lower zone contains only modifiers and *halant* marker, while upper zone contains modifiers and portion of some basic characters. Here, at first, we consider the text line and zone separation problem. Next, the word and character segmentation problem is tackled.

3.5.1 Detection of text lines and zones

The lines of a text block are segmented by finding the valleys of the histogram computed by row-wise sum of pixel values. Note that at the time of skew correction we know the headline part of each text line. The point between two consecutive headlines, where the histogram height is least, is noted. An imaginary horizontal line through this point may be called a boundary line. A text line can be found between two consecutive boundary lines. For example, see Fig.3.7. In the figure dotted horizontal lines denote line segmentation marking.

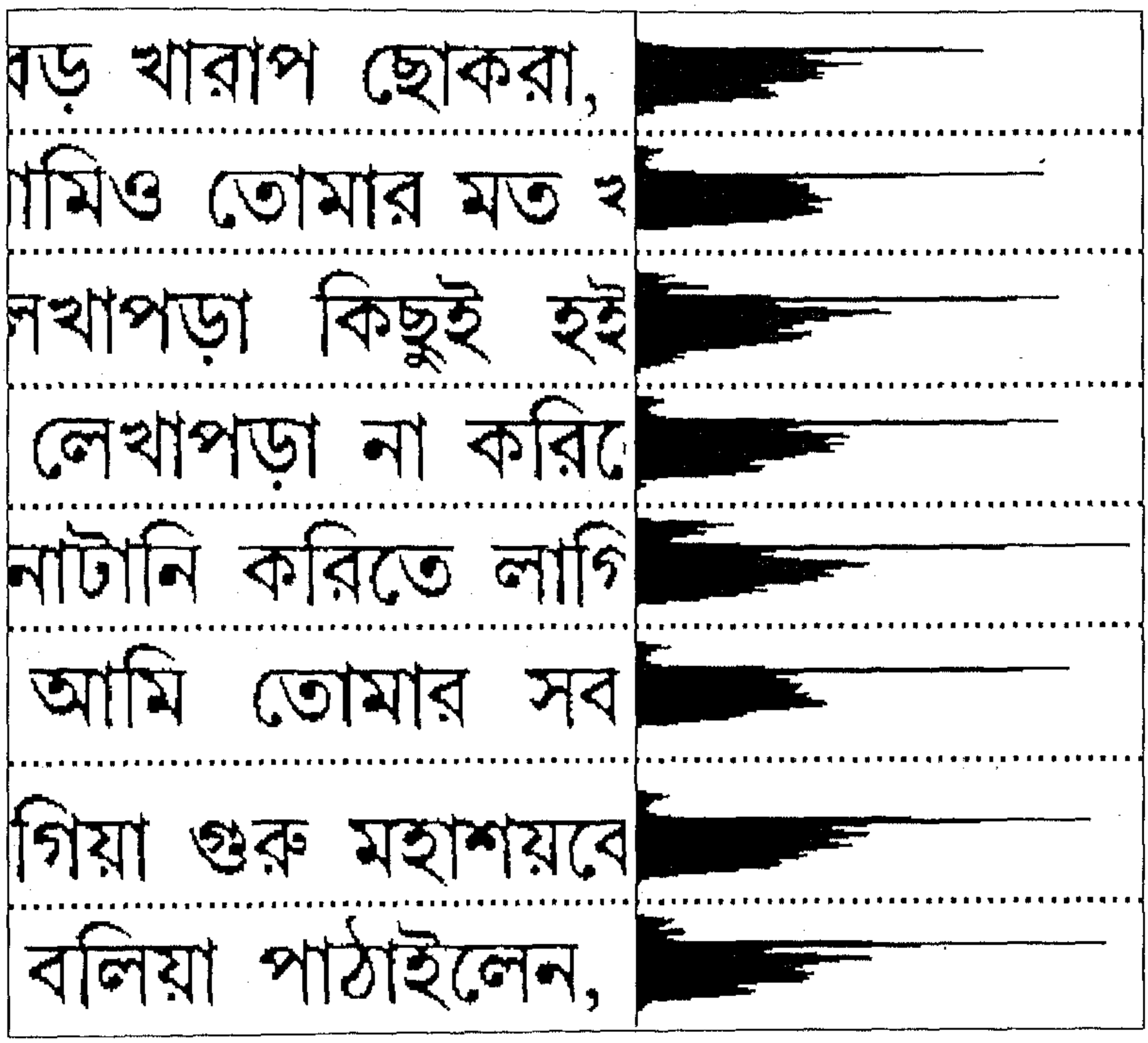


Figure 3.7: Text line segmentation approach. (Dotted lines are the separators of text lines).

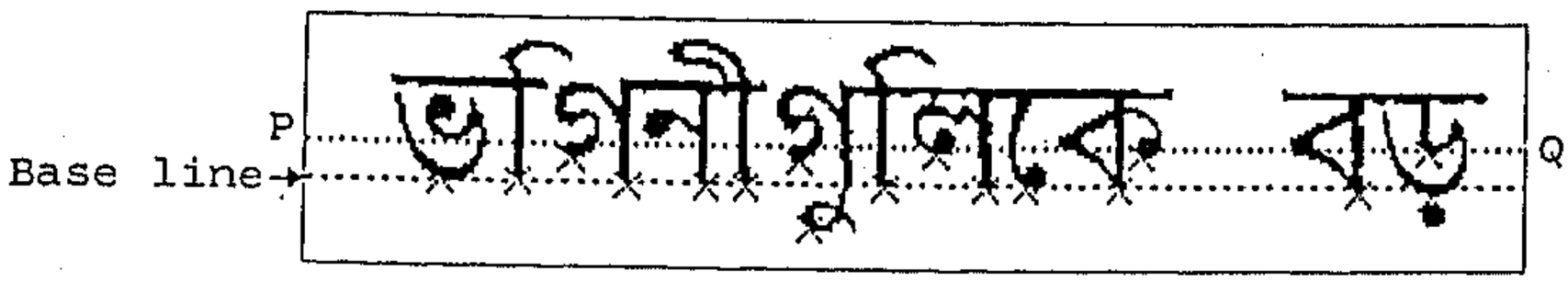


Figure 3.8: Lower boundary detection approach. Here, ×'s denote the lowermost points of the components below the line PQ

From Fig.2.4 it may be noted that the upper zone of a text line can be separated from the middle zone of a text line by the headline. Note that the headline is already detected in skew correction routine. Thus, detection of upper zone is almost instantaneous. Recognition of lower zone is relatively difficult because some text lines may not have any lower zone signature.

To separate the middle and lower zone of a text line we have applied a fast technique as follows. Consider an imaginary straight line $\overline{PQ}$ in the middle of the two consecutive boundary lines of a text line. This line partitions a text line region into equal halves. Consider only components below $\overline{PQ}$ and connected to $\overline{PQ}$. For each connected component below $\overline{PQ}$ the lowermost pixel is labeled by '×'. The horizontal line which passes through maximum number of such '×' labeled pixels is the separator line (base line) between middle and lower zone. See Fig.3.8 for an illustration.

Note that separation of middle and lower zone is relatively less accurate. Moreover, the tails of some characters fall in the lower zone. Care is needed to distinguish them from the vowel modifiers, *halant* sign etc. staying in this zone. We shall discuss this problem during vowel modifier detection. We include headline region, which is nearly 6% of the height of the text line, within the middle zone. The

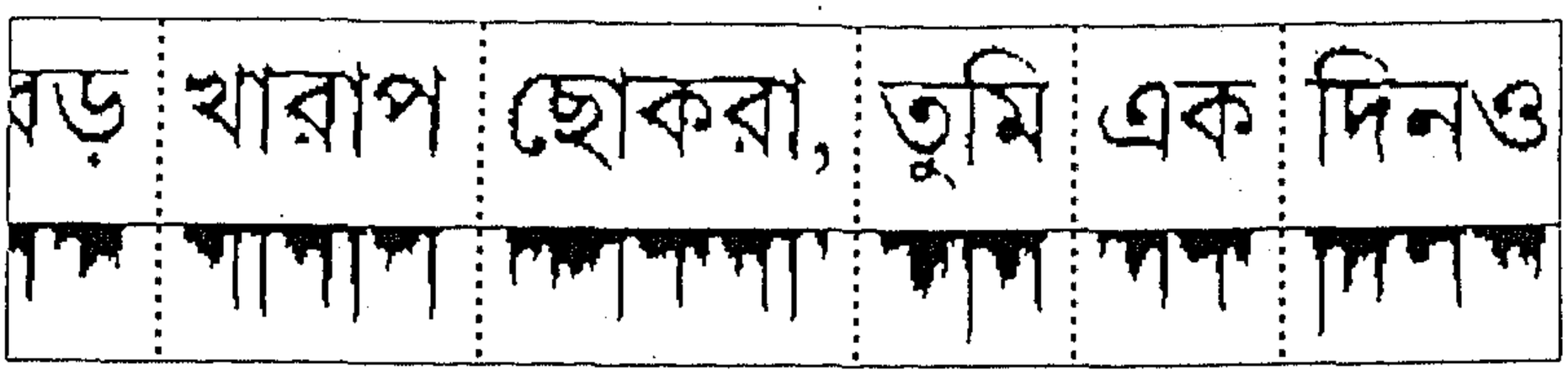


Figure 3.9: Word segmentation approach. (Dotted lines are the separators of words).

relative heights of upper, middle and lower zone in Bangla script are nearly 27%, 52% and 21%, respectively in Lino font with default style. Compressed/expanded styles have different zone height proportions.

The upper zone may contain parts of some basic and compound characters, parts of vowel modifiers, nasalization sign, apostrophe sign and consonant modifier like *ref*(´). The lower zone may contain vowel modifiers, consonant modifiers, halant sign etc.

3.5.2 Word and character segmentation

After a script line is segmented, it is scanned vertically for word segmentation. If in one vertical scan two or less black pixels are encountered then the scan is denoted by 0, else the scan is denoted by the number of black pixels. In this way a vertical scanning histogram is constructed, as shown in Fig.3.9.

Now, if in the histogram there exist at least k_1 consecutive zero values then the midpoint of that run of 0's is considered as the boundary of a word. The value of k_1 is a function of text line height.

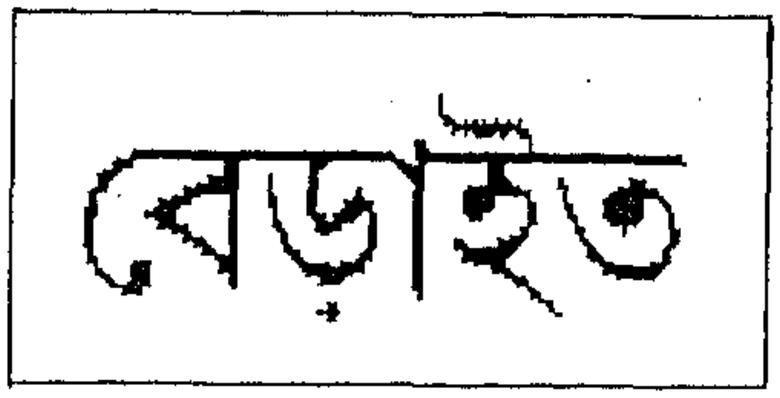
The algorithm is robust since characters of a word are mostly connected through the headline. Experiment with a large corpus of words in fair documents has produced zero word segmentation error.

Since characters in a word are mostly connected through the headline, character segmentation from word is an important task in Bangla OCR. To segment the characters we consider only the middle zone. The basic idea in character segmentation is similar to rub off the headline from the middle zone of the word so that the characters get topologically disconnected.

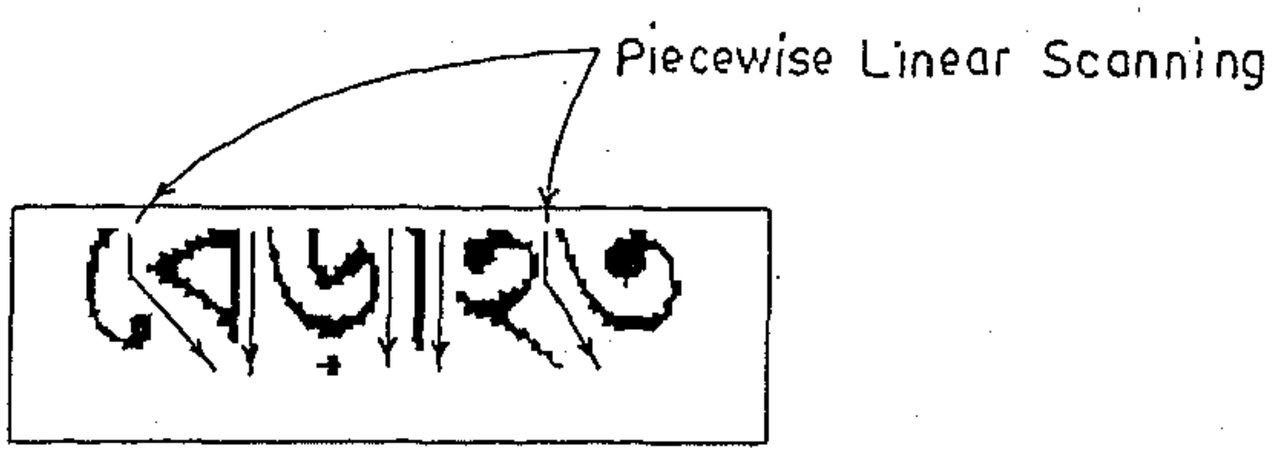
Note that the headline is already detected during skew detection. An estimate of average thickness t of character strokes is found by run-length statistics of black pixels. Now, from the onset of headline, t scan-lines can be horizontally rubbed off to make the characters topologically disconnected, as shown in Fig.3.10(b). Connected component labeling can be used to isolate these characters. But some characters like R(র), Y(য) have isolated dots which would be separated out and postprocessing would be necessary to take care of them. Instead, a quicker and more robust approach has been used where a linear scanning in the vertical direction, starting from the lower most row of headline, is initiated. If during a scan, one can reach the base line without touching any black pixel then this scan marks a boundary between two characters. Sometimes *kerned* characters (Kerned characters are the characters who overlap with neighboring characters) [95] may be present in the script for which a piecewise linear scanning method has been invoked (see Fig.3.10(b)). Using this approach nearly 98.6% characters have been properly segmented for fair documents. The characters which could not be separated were touching (merged) characters arising out of gray-tone to two-tone conversion. The percentage of merged character is very small. For segmentation of merged characters we have used the algorithm due to Kahan *et al.* [74].

By the character segmentation method some characters like modifier numbered 9 and 10 of Fig.2.3(a) get segmented into two parts. We consider each part of these characters as separate character.

Occasionally, characters like G(গ) and N*(ণ) can wrongly be split into two sub-segments by deletion of headline. See Fig.3.11 where the character G

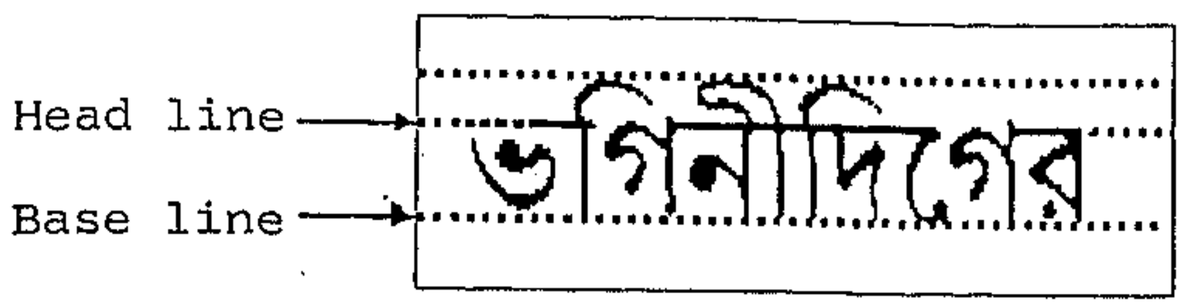


(a)

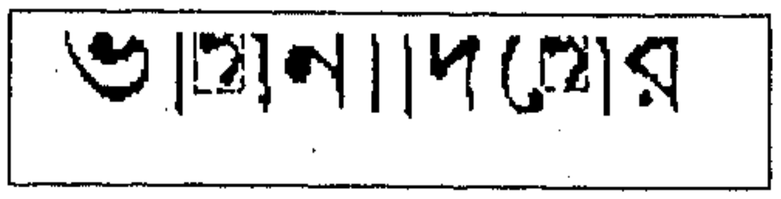


(b)

Figure 3.10: Character segmentation approach. (a) A Bangla word. (b) Scanning for character segmentation after deletion of headline from middle zone.



(a)



(b)

Figure 3.11: Detection of two parts of a wrongly segmented character. (a) A Bangla word (b) Wrongly segmented parts of the word are shown within a square box.

(গ্) has been wrongly segmented into two parts. The left sub-segment is enclosed within a box while the right sub-segment has a vertical line. They can be joined by the following heuristics. In general, all characters in the middle zone touch the boundary line between middle and lower zone (i.e. base line). Since here the left sub-segment does not touch the base line, the algorithm knows that it is not a valid character in the alphabet and combines the left and right sub-segments.

In actual practice the headline is not rubbed off to segment the characters. The lowermost row of the headline is noted. A linear or piecewise linear scan, starting from this noted row in the middle zone, segments the characters, as in Fig.3.10(b).

The combination of lines, words and characters segmentation of a Bangla text has been shown in Fig.3.12. Once the segmentation is done, simple features such as bounding box, zonal information etc. are computed on each character. The bounding box is computed for character portion in the middle zone only. In the

upper or lower zone of the bounding box, a character may have a portion of its own or portions of vowel modifiers (and occasionally consonant modifiers). They are used for both initial grouping and final recognition of the characters, as described in forthcoming chapters.

3.6 Conclusion

From the experiment we have noted that in good text documents, the lines and words are properly segmented. If the printing matters are composed manually then occasionally two consecutive text lines may not be parallel to each other. See Fig.3.13 for example. Here, the second text line is not parallel to the first one. Also, occasionally a character is vertically shifted with respect to others. Such a situation creates problems in skew correction and character segmentation.

For noisy documents, the line and word segmentation is less affected than character segmentation. The preprocessing technique described in this chapter can be applied for Hindi, Panjabi and other language scripts because of the presence of the headline. We have already experimented with Devnagari script and obtained encouraging results.

Figure 3.12: Lines, words and characters segmented image. Small vertical line segments are the character separators, while dotted lines are line and word separators.

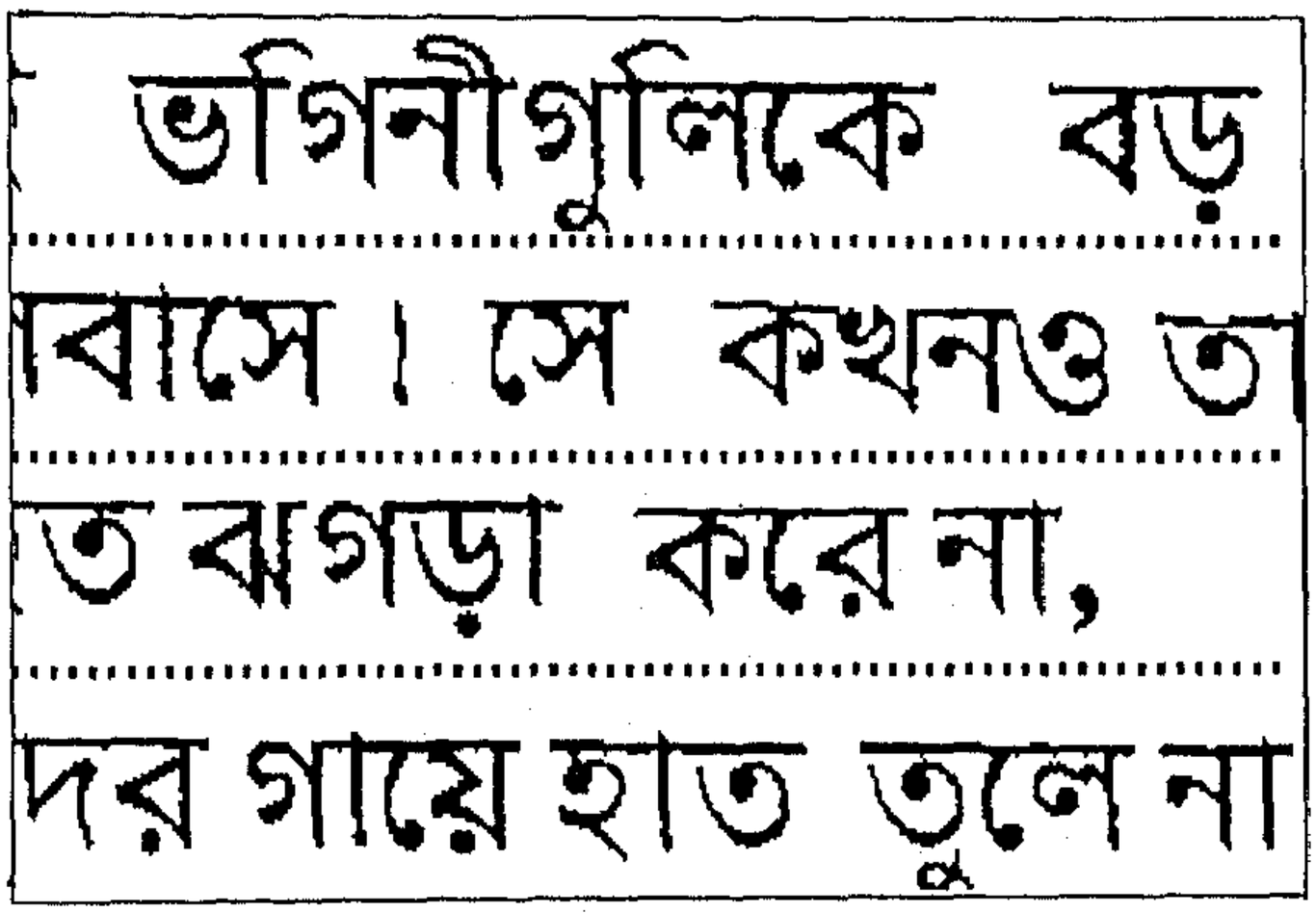


Figure 3.13: Example of a Bangla text where two text lines are not parallel. (Here second line is not parallel to first or third line)

Table 3.2: Mean and Standard Deviation (SD) of estimated skew angles obtained by different methods. (For each true skew angle the statistics is computed over 20 document images)

True skew angle (in deg) (manual)	Mean and SD of estimated skew angles using method					
	A_1		A_2		A_3	
	mean	SD	mean	SD	mean	SD
40	40.396	0.285	40.034	0.256	39.889	0.301
20	20.174	0.439	20.049	0.3162	20.047	0.242
10	10.271	0.393	10.166	0.201	10.112	0.323
5	5.064	0.458	4.962	0.213	5.188	0.233
2	1.986	0.396	2.151	0.234	2.054	0.307

A_1: Hough transform over total image.

A_2: Hough transform over SDSLs of upper envelope.

A_3: Proposed quick method

RECOGNITION

DIVISION

Chapter 4

FEATURE SELECTION AND DETECTION, INITIAL CHARACTER GROUPING AND TREE CLASSIFIER

4.1 Introduction

Feature selection is one of the most important part in OCR because features are the main keys to recognize the unknown characters. Different types of shape features have been proposed to recognize characters efficiently and accurately which we have reviewed in Section 4.2.

As we mentioned earlier, Bangla script contains large number of compound characters with small occurrence probability. If we can separate them, we can use robust, accurate and fast algorithm for more frequent basic and modified characters. With this idea, we classify the characters into one of the major three groups namely:

basic, modified and compound character groups by a feature based discriminant analysis. The approach is presented in Section 4.3. This initial grouping allows us to use separate methods for basic and compound character recognition.

We choose simple but robust stroke features of the characters those are easy to detect. For classification, we used tree classifier for basic characters and modifiers while a combination of tree classifier followed by template matching approach for compound characters.

This chapter is organized as follows. The feature selection and detection techniques have been described in Section 4.2. Section 4.3 deals with character group identification. The decision tree classifier has been elaborated in Section 4.4.

4.2 Feature selection and detection

4.2.1 Feature selection

Since a line of Bangla script can be partitioned in three horizontal zones, it is convenient to consider features in a zonewise manner. All characters do not have signatures in the upper or lower zone. The characters having signatures in the upper zone are basic characters like I(ই); I,(ঈ); U(উ); U,(ঊ); (AI)(ঐ); (AU)(ঔ); T,(ট); T,H (ঠ); the vowel modifiers numbered 2, 3, 8, 10 (see Fig.2.3(a)) and the consonant modifier numbered 3 (see Fig.2.3(c)). The vowel modifiers numbered 4, 5, 6, and marker(্) for pure consonant called *Halant* are the principal signatures in the lower zone. Since they are few in number, all of the shapes are considered as features.

Many feature-based character recognition schemes start with thinning the character outline [41]. To avoid artifacts due to thinning and to reduce computation time, we have not used thinning at any stage.

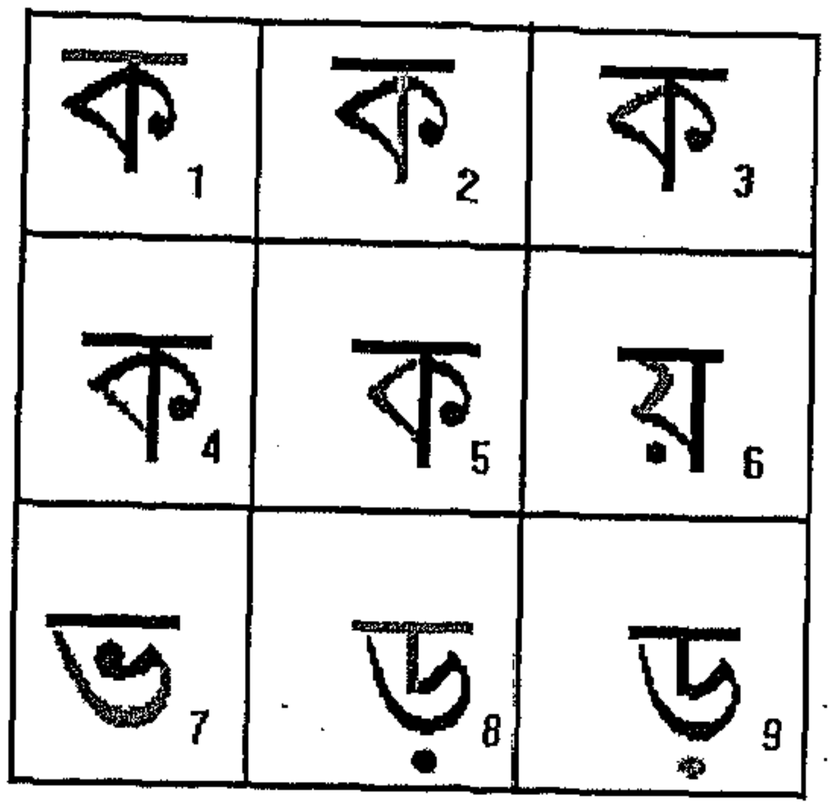


Figure 4.1: Stroke features used for the character recognition. (Less dark portions in the character represent the features. For easy identification, the features are numbered).

We have considered a few stroke features for initial classification of basic characters. The features are used to design a tree classifier where the decision at each node of the tree is taken on the basis of presence/absence of a particular feature.

In Fig.4.1 the principal set of features chosen for our classifier are shown in the context of the characters. Apart from the *principal features* some other features are also used at some nodes of the tree classifier. They are described during tree classifier design. The features are chosen with the following considerations: (a) Robustness, accuracy and simplicity of detection (b) Speed of computation (c) Independence of fonts and (d) Tree classifier design need.

It may be noted that the features of Fig.4.1 are simple (mostly linear) in structure and hence quick and easy to detect. Their distortion due to noise can be easily taken care of. They are quite stable with respect to font variation. At a

non- terminal node of the tree classifier, we use a feature that is able to subdivide the characters in that node into two groups so that the sum of the occurrence probabilities of one group is as near as possible to that of the other group. The features of Fig.4.1 serve such purpose to a reasonable extent.

Note that these features should be searched in the middle zone only. For the upper and lower zones we use some shape characteristics of the modifiers, described in the next chapter.

4.2.2 Feature detection

The methods of detecting the features of Fig.4.1 are described below. Here the stroke lengths are standardized with respect to the character middle zone height because this height is constant for characters of single font and fixed size.

The stroke numbered 1, shown in Fig.4.1 is essentially the same as the headline of a character. To detect it, we assumed that the length of the stroke l_a is at least 75% of the height of the character middle zone. Now, the upper quarter part of the character middle zone is scanned horizontally. If a scan contains continuous black pixels whose number is more than or equal to l_a then the stroke is assumed to be present in that character.

The method of detection of stroke numbered 2 is similar to that of stroke numbered 1, but here scanning area is different and scanning mode is vertical.

To detect stroke numbered 3, we assumed that the length of this stroke l_c is 40% of the height of the middle zone of the character. A diagonal scan is made (so that scanning direction makes $+45^0$ with horizontal) in the upper half of the middle zone of the character. If the number of continuous black pixels in a scan is greater than or equal to l_c then the stroke numbered 3 is present in the character. The stroke numbered 4 can be detected in a similar way. However, here the scanning is made in the lower half part of the middle zone of the character along 45^0 direction.

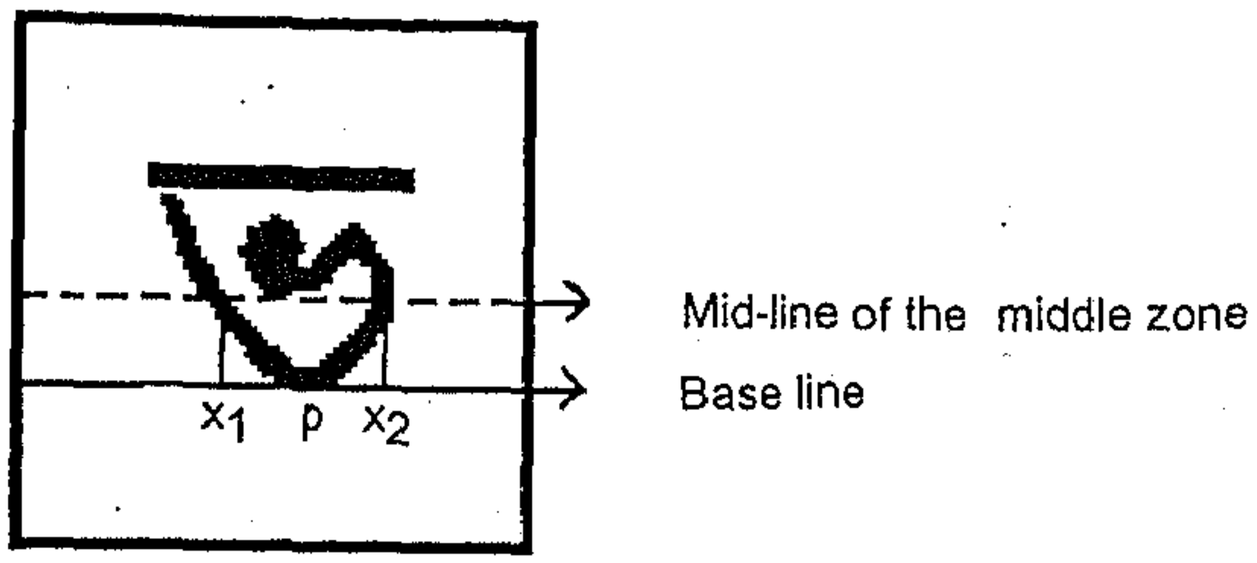


Figure 4.2: *Detection method of stroke numbered 7 of Fig.4.1*

We do not need any extra computation to detect the stroke numbered 5. If the stroke numbered 3 and stroke numbered 4 are both present in the character, we know that stroke numbered 5 is present in that character.

To detect the stroke numbered 6, the length of the arms of the stroke is assumed to be 30% of the width of the middle zone of the character and the angle between them is 315^0. A two way scan is performed where starting point of the scan is at the left hand side of the middle of the vertical line of the character. If in a scan each arm contains continuous black pixels of length greater than or equal to the length of the stroke then stroke numbered 6 is deemed present in that character.

Stroke numbered 7 is a cup-shaped feature where the bottom of the cup touches the base line. To detect it, horizontal scans are performed while the left-most and the rightmost black pixels are noted. For each horizontal scan, the x co-ordinates of these two extreme points are of nearly equal distance from P, where P is the midpoint of black run (run is a set of consecutive pixels of similar colour) where the cup touches the base line. This distance increases monotonically as we go scanning upwards till the middle of the middle zone. See Fig.4.2 where $\overline{PX_1} \simeq \overline{PX_2}$, and both $\overline{PX_1}$ and $\overline{PX_2}$ increase monotonically with scans.

To detect stroke numbered 8, at first we check whether the stroke numbered 1 is present. If yes, then the presence of a stroke like stroke numbered 2, whose

length is 40% of the height of the middle zone touching the mid-point of stroke numbered 1 is tested. If the test succeeds then presence of stroke numbered 8 is accepted.

For stroke numbered 9, it is tested in the lower half of the middle zone, whether more than 95% pixels within a circle having diameter equal to 12% of the middle zone height (that means the thickness of headline) are black. If yes, and if it is not a part of another stroke then stroke numbered 9 is assumed present.

For quick detection of circle, at first we scan horizontally. If a run of R black pixels is present then from its midpoint O we scan vertically. If a run of R black pixels is present with O as midpoint then we scan diagonally from O. If again successes are obtained for both the diagonal directions then we assume that a circle of diameter R is present.

From the experiment, we have noted that in most of the cases the strokes numbered 1, 2, 3, 4, 5 and 8 are correctly detected. Detection accuracy of different stroke features are shown in Table-4.1.

4.3 Initial grouping of characters

Before going for actual classification, the basic, modified and compound characters are separated. One motivation of this step is that although there are large number of compound characters, they occupy about 6.17% of the text corpus. If we can isolate them, then the small subset of basic characters can be recognized by a fast, accurate and robust technique while we could use a slower technique to recognize the compound characters.

The middle zone of a script line may contain the vowel modifiers that have very small width. The width of the basic characters vary from character to character, being the least for character H,(ಃ) and the most for character N+(ಞ) in almost

all fonts. On the other hand, some compound characters like /DG/(দ্গ), /R,G/ (ড়্গ) have very large width. (Here /DG/ denotes the compound character formed by the characters D(দ) and G(গ))

Another important point to note is that the compound characters are, in general, more complex in shape than the modifier and basic characters. In general, a compound character will have large border length and high curvature along the border than a basic character. Thus, using the following three features the group of a character among the three groups mentioned above may be identified.

- Feature f_1. Bounding box width of the character.
- Feature f_2. Number of outer border pixels per unit width, which is computed by dividing the total number of character outer border pixels by the width of the character.
- Feature f_3. Accumulated curvature per unit width. To compute f_3 the outer border of the character is traversed. The absolute angular change made by traversing from one pixel to the next is added over all pixels of the border. This accumulated value is divided by f_1 to get f_3.

Two discriminant planes of the form

$$a_1 f_1 + a_2 f_2 + a_3 f_3 = a_0 \quad \text{and} \quad b_1 f_1 + b_2 f_2 + b_3 f_3 = b_0$$

are used for classification. The parameters of discriminant planes are found from a large number of training data. For 12 point text digitized at 300 DPI the typical values of $a_0, a_1, a_2, a_3, b_0, b_1, b_2, b_3$ are 225.77, 0.30, 0.21, 1.99, 80.83, 0.22, 0.49, 0.67 respectively. On the test data set, an overall classification accuracy of 96.33% was obtained by this classifier.

The following cases contribute to misclassification. Because of complex structure and large width, basic characters like A(অ), J(ঙ) etc., sometimes fall into

compound character group. Also, due to smaller width, basic characters H,(ঃ) and N.(ঁ) always falls under modified character group. On the otherhand, some simple shaped compound characters like /NN/(ন্ন). /DD/(দ্দ) etc. sometimes fall under basic character group. Moreover, it is noted that the modified character like e(ে) sometimes falls under basic character group.

The misclassification results were analysed from training data and the final recognition scheme was modified to rectify the error. From the data of 20,000 words, a confusion matrix of the classification is shown in Table-4.2.

4.4 Tree classifier

In a decision tree or hierarchical classifier, an unknown pattern is assigned to a class using several decisions in a successive manner. A tree classifier consists of a root node, a number of non-terminal nodes and a number of terminal nodes. A terminal node has only an ascendant node but a non-terminal node has both ascendant and descendant node. From the classification point of view, terminal nodes of the tree represent single pattern classes and the non-terminal nodes denote group of pattern classes. In particular, the root node represents the entire set of classes into which a pattern may belong. We can associate a *level* to each node of the tree. The *level* of a node p is the number of nodes one should traverse from the root node to reach p. If the group of pattern classes corresponding to a node is disjoint from group of pattern classes of any other node with the same level then the decision tree is *non- overlapping*. Tree classifier has been used in a variety of pattern recognition problems including character recognition [175].

A tree classifier has the following advantages over single stage classifier [103].

(1) The classification accuracy may improve by using only features that are pertinent for classification at a given stage. Since the number of features used at one

stage is small, the dimensionality problem is less severe than that of a single stage classifier which uses several features at one time. Also, at each non-terminal node of the tree, we encounter the problem of a smaller number of classes than that encountered in a single stage classifier.

(2) If the tree classifier is such that the total number of features needed is not more than the number of features used in a single stage classifier, then the tree classifier is better in computational efficiency because normally not all characters require all of the features for classification.

(3) A hierarchical classification represents a more logical structure for the division of objects. For instance, the stroke feature numbered 1 of Fig.4.1 divides the characters into two groups which followed by the stroke feature numbered 2 divides these two groups into four in a rather natural way.

The main disadvantage of a tree classifier is that it is very difficult to design an optimum classifier because the number of possible tree structures for an m-class problem, even for moderately small m is astronomical. Also, the features used at each node of the tree usually are not optimal because an optimal feature subset can only be obtained by exhausting all the possible combinations of feature subsets which is overwhelming even for the moderately small number of features.

4.4.1 Tree classifier design considerations

The design of a tree classifier has three components: (1) a tree skeleton or hierarchical ordering of the class labels, (2) the choice of features at each non-terminal node, and (3) the decision rule to be used at each non-terminal node.

We have chosen the number of descendant nodes as two and number of feature at each non-terminal node as one. For the choice of features at a non-terminal node we have considered the occurrence statistics of the characters in Bangla given in Table- 2.2(a). The decision rules are mostly binary e.g. presence/absence of the

feature.

If the pattern group of any non-terminal node can be subdivided into two subgroups by examining a feature so that the sum of occurrence probabilities of one group is as close as possible to that of the other group then the resulting binary tree is optimum in time complexity, assuming that the time required to test a feature is constant. The resulting tree is identical with the Huffman code [24] tree generated using the character occurrence probabilities.

However, we may not get a set of natural features to design such an optimal tree. A semi-optimal tree is generated if out of the available features, we select the one for a given non-terminal node that separates the group of patterns best in the above sense. In this way, we used stroke feature numbered 1 and 2 of Fig.4.1 near the root nodes (please see Fig.5.3). For example, the 'vertical line' i.e. feature numbered 2 in the right side of the character boundary, separates the characters into two groups with sum of probabilities 0.59 and 0.41. Features numbered 1 and 2 partition the basic characters into four nearly equi-probable subgroups. Fortunately, they are the simplest to detect too.

Note that the features of Fig.4.1 are positional strokes. Detection of some of them (e.g. feature numbered 4 of Fig.4.1) should follow detection of others (e.g. feature numbered 2 of Fig.4.1). In other words, if the feature numbered 2 exists in a character then only we will check for feature numbered 4 in that character. Moreover, as we go down the tree, the number of features to choose from gets reduced. These problems put further constraint in designing the optimal tree classifier.

For character recognition by a tree classifier, the average number of nodes visited is $\sum_{i=1}^{n} n_i p_{c_i}$, where n_i is the number of nodes to be visited for the recognition of the character c_i and p_{c_i} is the probability of occurrence of the character c_i. This number is the smallest if Huffman tree could be used for classification [24]. For a comparative study, we have computed the average number of nodes to be visited

in Huffman tree and our classification tree for basic characters (Huffman tree was generated by the probability estimates of the basic characters). We have found that average number of nodes to be visited for a basic character recognition in Huffman tree (i.e. optimum tree) is 4.714 while that in our classification tree is 5.331. Given the constraints of selecting and using the features described above, our classification tree is quite close to the optimum one.

4.5 Conclusion

At first, feature selection and detection technique of the OCR system is presented in this chapter. The features chosen for modified and basic characters are font and size independent. Next, the purpose and techniques of initial character grouping is elaborated. Finally, a brief idea about tree classifier which we have used for classification is described. In the next chapter, the actual character recognition using a tree classifier designed mostly using these features is described.

Table 4.1: Detection accuracy of different stroke features.

Stroke number	Detection accuracy
1	99.90%
2	99.72%
3	99.65%
4	99.33%
5	99.35%
6	98.32%
7	98.21%
8	99.53%
9	96.61%

Table 4.2: Confusion matrix of three group classification.

	Basic	Modified	Compound
Basic	95.980%	0.799%	3.219%
Modified	2.163%	97.836%	0.000%
Compound	9.340%	0.000%	90.659%

Chapter 5

CHARACTER RECOGNITION

5.1 Introduction

Character recognition is a process for converting character image into computer readable form. The input of the character recognition module is the text, segmented into lines, words and characters. The output is a coded file with an ASCII (or other character code) as well as special symbols for unrecognized patterns. Recognition phase is the most important part in the OCR where the system extracts information from the input character and detects the class in which the character belongs.

Our recognition scheme of modifiers is based on shape identification only. The recognition scheme of basic characters is more involved. A feature based tree classifier separates most of the characters. For some characters, additional heuristics are used to separate two or more characters identified near a leaf node of the tree. For the compound characters, feature-based tree classifier is initially used to separate them into small groups. Next, an efficient template matching approach is employed to recognize individual characters.

In this chapter, at first we discuss about the modified and basic character

recognition techniques. Next, the compound character recognition approaches are considered.

5.2 Modified and basic character recognition

In the previous chapter the tree classifier is introduced. Decision is made at each node of the tree classifier by certain feature(s) from the feature set. In particular, one or more of the following judgments are usually made.

(a) Presence/absence of particular features.

(b) If present, then test is made on

  (i) the position of the feature with respect to the boundary box of the character (left half, right half etc).

  (ii) relative position of a feature with respect to other feature (whose presence was confirmed at a higher node of the tree).

  (iii) zonal position of the feature (upper, middle, lower zone).

(c) The same feature may be present in more than one place of the character. For example, in the character JH(ঝ) feature numbered 2 of Fig.4.1 occurs twice. In this case, number of times the feature is present and their positions may also be used for decision making.

If a character belongs to the modifier class then the recognition is done as follows.

If the character satisfies the shape of stroke numbered 2 of Fig.4.1 then it is one of the vowel modifiers numbered 1,2,3 of Fig.2.3(a) or right part of the vowel modifiers numbered 9 or 10, or consonant modifier numbered 1 of Fig.2.3(c). Note that vowel modifier numbered 1 and right part of the vowel modifier numbered 9

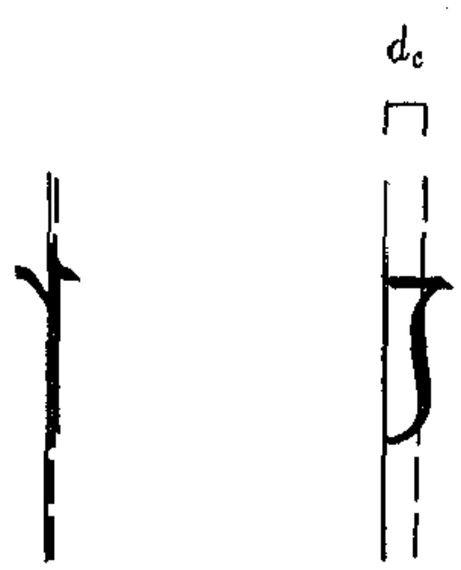


Figure 5.1: Two similar shaped modified character characters separation. Here dashed line indicates position where feature numbered 2 of Fig.4.1 exists in the character and the solid line is the vertical line through the lowermost pixel of the character.

of Fig.2.3(a) are the same. To discriminate them, the upper zone of the character is tested.

If the upper zone is empty then it is a vowel modifier numbered 1 (right part of the vowel modifier numbered 9) or consonant modifier numbered 1. To discriminate them, we find the column which contains the lowermost pixel of the modifier. See Fig.5.1 where this column is shown by solid line. Also, we note the column where the feature numbered 2 exists. This column is shown by dotted line in Fig.5.1. Let the distance between these two columns be d_c. If the distance d_c exceeds two times of the headline width (which is nearly 6% of the height of the text line) then the character is the consonant modifier numbered 1, Else, it is the vowel modifier numbered 1.

If the upper zone is non-empty, the character is one of the vowel modifier

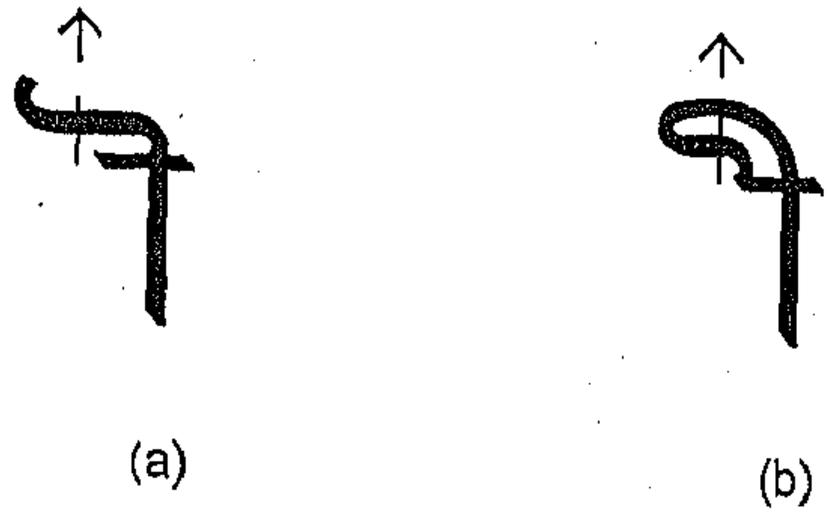


Figure 5.2: Two similar shaped modified characters separated by number of crossing. In (a) number of crossing is 1 while in (b) it is 2.

numbered 2, 3 or right part of the vowel modifier numbered 10. Noting the *crossing count* and the position of the upper part, these characters are separated as follows. The upper part of the modifier numbered 2 lies in the right side of the position where stroke numbered 2 occurs while it lies in the left side for the other two. Now, using crossing count from the headline towards upper zone, vowel modifier numbered 3 is separated from the right part of vowel modifier numbered 10. For vowel modifier numbered 3, the number of crossing is 2 while it is 1 for the right part of vowel modifier numbered 10. See Fig.5.2 for illustration.

Proceeding in this way other middle and upper zone modifiers can also be recognized.

Since modifiers for O(ে) and (AU)(ৌ) have two parts, one sitting to the left and another to the right of the basic or compound character, a delayed decision is made after the recognition of modifier for E(ে) which is the left part of the modifier for O(ো) or (AU)(ৌ). For example, if a modifier is recognized as

vowel modifier numbered 7 and its next to next character is recognized as vowel modifier numbered 1 then we conclude that they together are the modifier of the vowel $O(\text{ে})$

To recognize vowel modifier numbered 4, 5, 6 and consonant modifier numbered 2, lower zone of the character is tested. Note that these modifiers are not segmented as separate characters by our segmentation process. The presence of stroke numbered 5 of Fig.4.1 in smaller size and loop in the lower zone signifies the presence of these modifiers. To test consonant modifier numbered 2 of Fig.2.3(c), the presence of stroke numbered 1 of Fig.4.1 is tested in the lower part of the character.

We have noted that in Bangla all the punctuation marks except the last one fall under modified character group. To separation them, we have used their positional characteristics. For example, it can be noted from the Fig.2.2(d) and Fig.2.3(a) that punctuation mark numbered 1 and vowel modifier numbered 1 are similar shaped character but vowel modifier numbered 1 always occurs within a word with some other characters while punctuation mark numbered 1 always occurs in isolation.

The classification tree of basic characters is partially shown in Fig.5.3. Most leaf nodes of the tree point to a single character by the principal features. For some cases, a non- terminal node may correspond to at most three characters by using the principal features. In addition to the features of Fig.4.1 some other simple features are also used near the terminal nodes. Consider for example, R.(ঋ) and DH(ধ) which belong to the same node of the tree classifier (see Fig.5.3). To separate them we count the number of vertical lines. For R.(ঋ) we get 2 vertical lines while for DH(ধ) we get 1 vertical line. Similarly, to separate the characters KH(খ) and TH(থ) which belong to the same node (see Fig.5.3), we have traced some of their border pixels. For illustration see Fig.5.4. Starting from the leftmost pixel of the character, border pixel tracing is made in clockwise direction until it reaches the vertical stroke. During border tracing we calculate the distance of each traced pixel

to the top of the character's bounding box. Noting this distance sequence we have separated the characters. For example, in the character TH(থ) we get 1 transition point of this distance sequence while for KH(খ) we get 3 transition points. By transition we mean change of values from increasing mode to decreasing mode or decreasing mode to increasing mode. If for a character in this node the number of transition points are other than 1 or 3 then we reject the character. This technique has been used for the recognition of other characters also.

Some basic characters like consonants T,(ট), T,H(ঠ) and vowels U(উ), U,(ঊ), I(ই), I,(ঈ) etc. have some portions in the upper zone. Hence, the upper zone is inspected to detect them as we have done for the modifiers.

Crossing count is also used for the classification of basic characters. For example, using crossing count two similar shaped characters U(উ) and U,(ঊ) are separated as follows. Both the characters have the feature numbered 8. In the left part of the end point of vertical part of the stroke numbered 8, the number of crossing is noted. See Fig.5.5 for example. In U(উ) the number of crossing is 1 while in U,(ঊ) it is 2. If for a character in this node number of crossing is greater than 2 then we reject the character.

Note that the numerals do not have stroke numbered 1. Also, the numerals excepting ৭ (seven) and ৮ (eight) have no stroke numbered 2 and hence fall in the category of characters O(ও), AU(ঔ), N, (ঙ), T.(ৎ) etc. To recognize them we have traced their boundaries and used some principal features. Crossing counts and projection profiles are also used for their recognition.

Recognition techniques of the other basic characters can be described in a similar way. For example, Table-5.1 shows classification of some similar shaped basic characters.

The reject option is initiated at the lowest node of the recognition tree. If a test character does not show all of its features, it is rejected. As our recognition

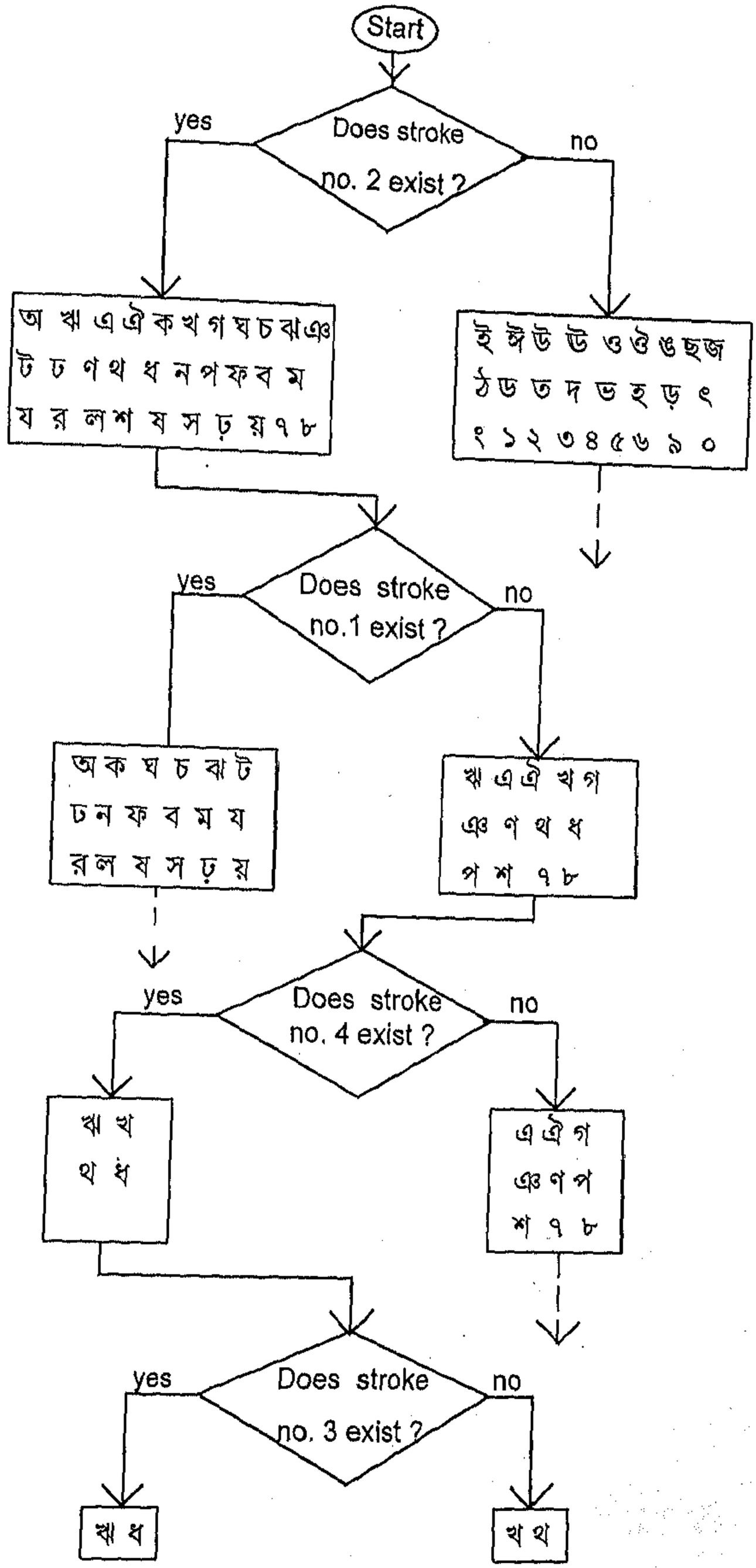


Figure 5.3: A flow chart representation of a portion of tree classifier for bi[illegible]racter recognition.
(Strokes are given in Figure 4.1)

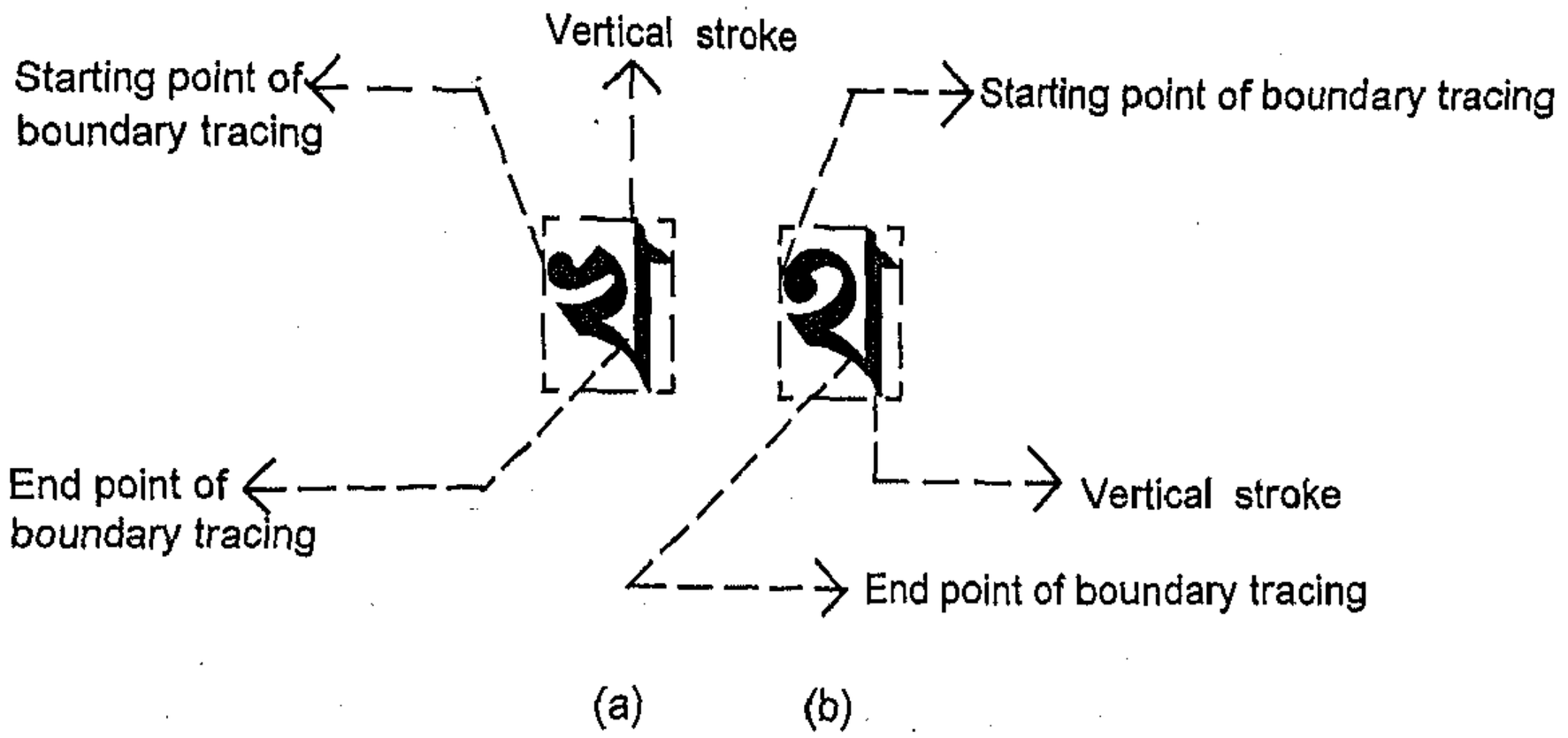


Figure 5.4: Two similar shaped basic characters separation by boundary tracing. In (a) number of transition point is 3 while in (b) it is 1.

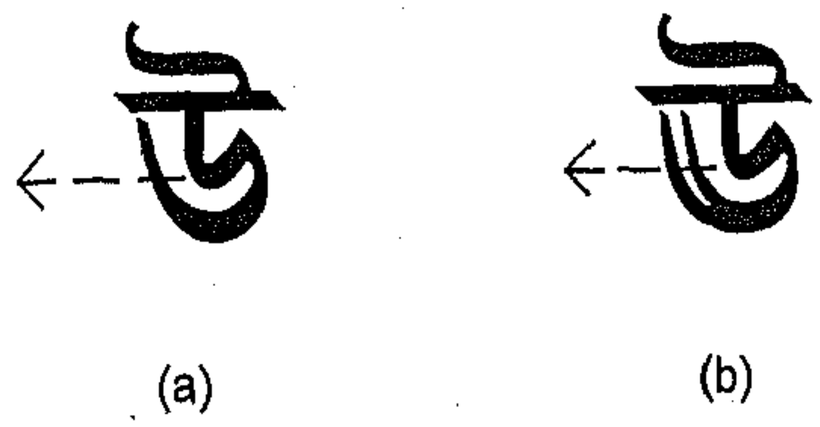


Figure 5.5: Separation of two similar shaped basic characters by number of crossing. In (a) the number of crossing is 1 while in (b) it is 2.

approach is a tree classifier, it is possible to memorize the subgroup to which the recognized text character belongs. This memorization of subgrouping can be helpful in error correction (see chapter 6).

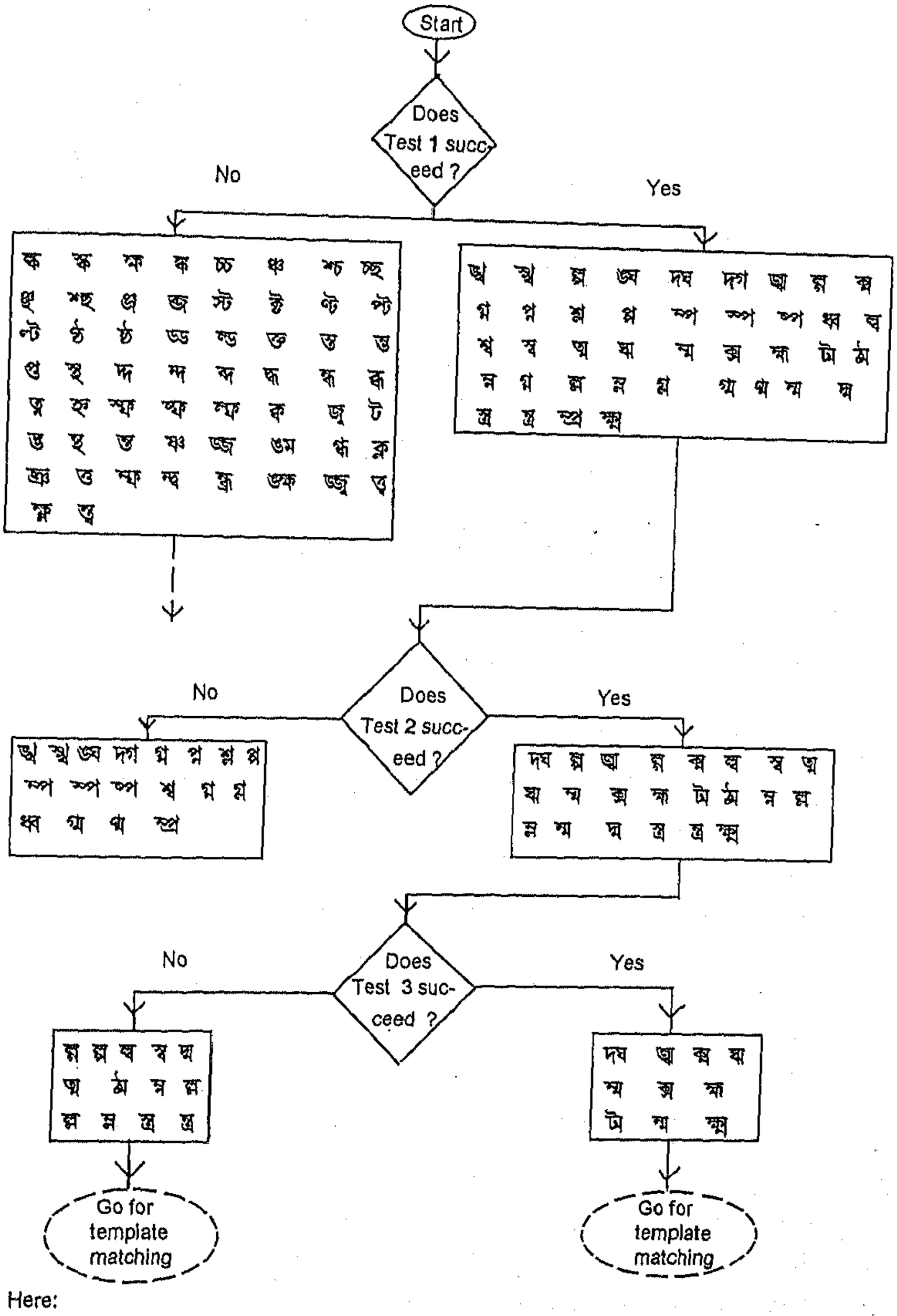
5.3 Compound character recognition

The compound character recognition is done in two stages. In the first stage, the characters are grouped into small subsets by a feature based tree classifier. In the second stage, characters in each group are recognized by a sophisticated run length based template matching approach. We have adopted this hybrid approach instead of only tree classifier approach because it is nearly impossible to find a set of stroke features which are simple to compute, robust and reliable to detect and are sufficient to classify a large number of complex shaped compound characters.

The features used in the tree classifier are headline, vertical line, left slant, right slant (i.e. feature numbered 1,2,3,4 of the Fig.4.1), boundary box width of the character middle zone, presence of some portions in upper zone etc. We noted that a terminal node of compound character tree classifier corresponds to a subset of at most 20 characters. These character templates are ranked in terms of their bounding box widths and stored during the training phase of the classifier at an address corresponding to this terminal node. A portion of tree classifier for subgrouping the compound characters given in Fig.2.5, is shown in Fig.5.6.

A candidate compound character of width W_c which has reached a terminal node of the tree is then matched against the stored templates corresponding to this node. Matching is done with the templates whose bounding box width is within $W_c \pm d$. The value of d is chosen as a fraction of W_c. Matching decision is speeded up by this process.

To superimpose the candidate on the template for computing the matching



Here:

Test 1: Checking of stroke number 2 of Figure 4.1 in the extreme right of the character bounding box.

Test 2: Checking of stroke number 1 of Figure 4.1 in the character.

Test 3: Checking for twice occurrence of stroke number 2 of Figure 4.1 in the character.

Figure 5.6: A portion of tree classifier for subgrouping compound characters.

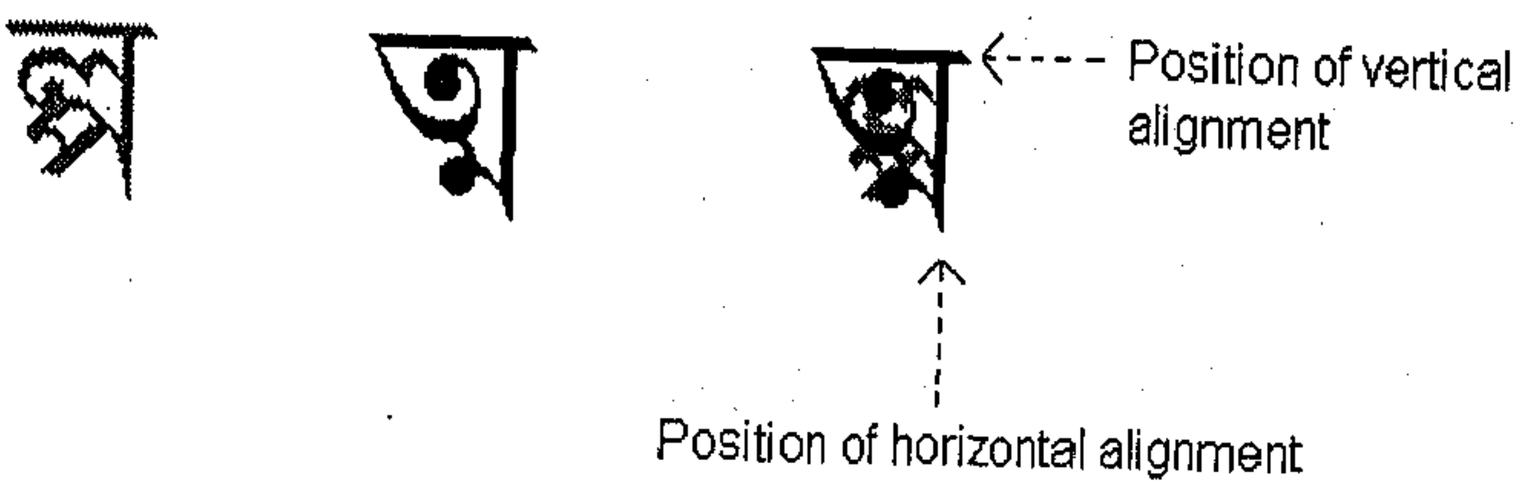


Figure 5.7: Example of template matching

score, the reference position is chosen by characteristic features of the candidate. For example, if stroke numbered 1 and 2 of Fig.4.1) are both present in the candidate (which is known from the first stage of the classifier), the group of templates must also have the same strokes, and the matching score is computed at first by aligning the candidate and template along these strokes. For illustration, see the Fig.5.7. Here, both the input character and template have stroke numbered 1 and 2. After the alignment matching score is computed. Scores for small shift about these strokes are also computed to find the best match. In this way the matching process is speeded up and accuracy of matching is also increased.

If the stroke numbered 1 is absent but stroke numbered 2 is present in the candidate then the above procedure is used by considering the headline as the line of vertical alignment.

For a character without these strokes, the headline is used as the reference position for vertical alignment. However, since there is no landmark for horizontal alignment the test character should be horizontally moved all over the template to

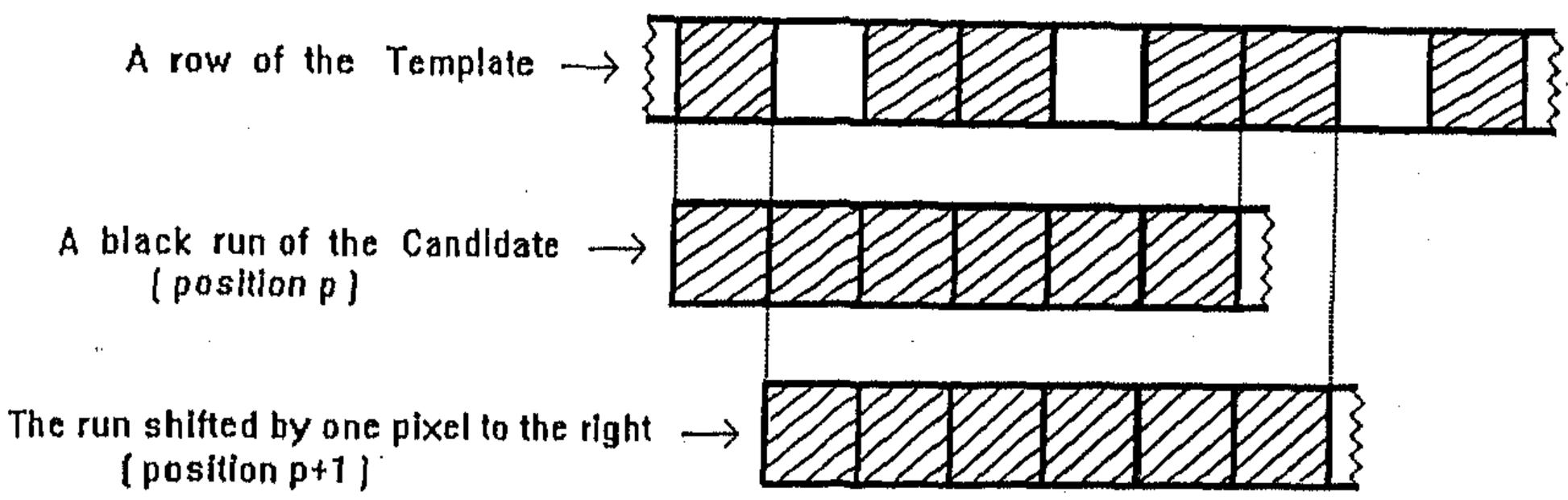


Figure 5.8: Run based template matching approach.

get the best score.

Further speed-up is done by using the concept of black and white runs, as illustrated in Fig.5.8. Let us consider a black run of the candidate character to be matched with a row of the template character. At position p, the black to black matching score is 4. At position p+1, the matching score is also 4. This result can be inferred by noting that pixels A and B in the template have the same colour. In general, we can compute the change in matching score for any run by noting the color of only two pixels in the template. Thus, the order of computation is of the order of number of runs, rather than the number of pixels.

A wide variety of correlation measures are available for template matching applications. Out of these the measure $J_C = \frac{n_{11}}{n_{11}+n_{10}+n_{01}}$ due to Jaccard and Needham [165], is found satisfactory for our purpose. Here n_{ij} = number of pixels with input image pixel value i and template pixel value j in the corresponding positions;

i,j = 0, 1. The J_C measure gives the ratio of total matched object pixels to the sum of the object pixels in template and candidate.

The best matching score for the candidate should exceed a threshold for acceptance as a recognized character. Else, the candidate is rejected as un-recognized.

One of the drawbacks of template matching approach is its susceptibility to character size variation. However, for Lino font we have noted that reasonable size variation can be accommodated by re-scaling the templates.

If templates for 12 point characters size are stored then it was found that characters of size ranging from 8 to 16 points can be matched by re-scaling the template mask without any appreciable error. Storing templates for different size is an alternative idea that needs more memory space. In any case the candidate middle-zone boundary box height is the guiding factor to choose the template of appropriate size which is either generated by re- sampling or called from the stored file of appropriate size. Recognition performance over different point size documents is given in the next chapter.

As mentioned before, some basic characters are found to be classified as compound characters during initial grouping. Their templates are also stored and they are matched with the candidate in the same way as above.

5.4 Results and discussion

Text containing about 20,000 words were digitized by flatbed scanner for making experiments with our systems. The text documents are pages from books, newspapers, juvenile literatures etc. printed in Lino font. The characters were of single style but with variable size ranging from 8 to 16 points. A limited set of bold characters were also experimented with.

During initial grouping of characters into three groups sometimes it is noted

that some basic characters like A(অ), J(জ) and N+(এঙ) fall into compound character category and compound characters like /NN/(ন্ন), /DD/(দ্দ) and /ND/(ন্দ) fall under basic character category. Similarly, basic characters M,(ং) occasionally fall into modifier group while the basic character H,(ঃ) and N.(ঁ) always fall into modifier group. On the otherhand, modified shape of E(এ) sometimes falls into basic character group. For correct recognition, we have considered these characters in both the groups in which they may be primarily classified.

We have obtained high recognition score on documents printed on clear paper. Based on the experiment it was noted that before error correction the system has an average accuracy of 96.48% in character level and 85.78% in word level, respectively. Note that error in word level is magnified n-fold where n is the average word length. Now, out of 14.22% word recognition errors 12.29% are due to single character error (i.e. misrecognized or rejected character). See Table-6.1 also. We have also noticed that 29% of single character error words are due to characters of compound character class and 71% of that for modifier and basic character class. We have seen that most of the rejected characters are due to compound characters. In the error calculation it is assumed that the original document contains no error.

The error rate depends on scanning resolution as well as paper quality. We observed that our system provides good result on documents of 12 points if scanned at 300 DPI. We noted that images captured from newspapers generate about 6% more error than other document images. We also noted that error rate increases by 9-12% in character level if the document image is captured from old documents.

We have computed misrecognition percentage of characters to their similar shaped characters and presented the results in Table-5.2. Here, the error percentage is computed as $\frac{\textit{number of character Y misrecognized as similar shaped character Z}}{\textit{total number of character Y in the text}} \times 100$

Average recognition time of the OCR system is 35-40 words per minute (which is equivalent to 155-180 characters per minute) on a SUN 3/60 machine (with

microprocessor MC68020, SUN OS version 3.0).

Some discussions on the performance of the system on size and style variations of the font are in order. Size and most style variations do not affect the modifier and basic character recognition scheme. Compound character recognition is somewhat affected by the size variation, but the rescaling of templates is effective if the range of size is between 8 and 16 points. For bold style, a set of bold templates yields better results. We have not made elaborate study on italics style since it is rare in Bangla documents.

Table 5.1
Separation techniques of some of the similar shaped basic characters at nodes near the leaf of the tree classifier.

Characters at a node near the leaf	Separation approach
ঘ ষ য় য	Dot below the character separates য় from others. Number of crossing is used for ষ separation. Boundary tracing technique is used for the separation of ঘ and য
ঝ ব র	Checking of two vertical lines separates ঝ from others. Dot below the character is used to separate র from ব
ত ভ	Boundary tracing technique distinguishes these characters.
ট ঢ ঢ়	Upper zone signature separates ট from others. The dot below the character separates ঢ় and ঢ
ড ড়	The dot below the character separates ড and ড়
ও ঔ	Upper zone signature distinguishes the characters.
ন ল	Boundary tracing technique is used for the separation of ন and ল

Table 5.2

Misclassification percentage of characters to their similar shaped characters.

Input char.	Misrecog-nized as	Percentage of misclassification	Input char.	Misrecog-nized as	Percentage of misclassification
প	গ	2.13	ক	ফ	2.91
গ	প	3.47	ফ	ক	5.22
ল	ন	2.66	ত	ভ	1.22
ন	ল	1.97	ভ	ত	3.76
ঘ	য	8.21	য়	য	3.16
য	ঘ	4.11	য	য়	2.94
ষ	য	5.22	খ	থ	7.39
য	ষ	7.13	থ	খ	2.79
ঊ	ঊ	1.33	র	ব	2.61
ঊ	ঊ	3.21	ব	র	3.17
ঋ	ধ	1.14	J	।	3.41
ধ	ঋ	0.97	।	J	2.83
ড	ড়	2.64	এ	ঐ	1.89
ড়	ড	3.79	ঐ	এ	2.32
৯	৯	1.72	শ	ণ	1.21
৯	৯	3.22	ণ	শ	2.17
ম	য়	1.49	ত্ত	ত্ত	11.13
য়	ম	2.11	ত্ত	ত্ত	9.14
গ্ম	গ্ম	7.49	স্থ	স্থ	6.17
গ্ম	গ্ম	5.33	স্থ	স্থ	8.37

POSTPROCESSING

DIVISION

Chapter 6

ERROR DETECTION AND CORRECTION

6.1 Introduction

The solution towards automatic error detection in words and automatically correcting them is a great research challenge. It has enormous application potentials in text and code editing, computer aided authoring, OCR, machine translation and natural language processing (NLP), database retrieval and information retrieval interface, speech recognition, text to speech and speech to text conversion, communication system for the disabled e.g. blind and deaf, computer aided tutoring and language learning, desktop publication, and pen-based computer interface. This chapter deals with automatic error detection and correction of our OCR system developed for Bangla.

The OCR word error can belong to one of the two distinct categories namely, *non-word error* and *real word error*. Let a string of characters separated by spaces or punctuation marks be called a *test string*. A test string is a *valid* word if it has a meaning. Else, it is a non-word. By real word error we mean a valid but

not the intended word in the sentence, thus making the sentence syntactically or semantically ill-formed or incorrect. In both cases the problem is to detect the word error and either suggest correct alternatives or automatically replace it by the appropriate word. The detection of real word error needs higher level knowledge compared to the detection of non-word error. In fact, detection of real word error needs NLP tools.

Here we are concerned with non-word error only. We have assumed that the document contains only valid words and the sentences are syntactically and semantically well formed. Of course, the OCR still can create real word error, but as discussed in Section 2.4, such errors are rare.

The organization of this chapter is as follows. A brief review on error detection and correction techniques is given in Section 6.2. Section 6.3 deals with OCR error patterns in Bangla. Error detection technique is elaborated in Section 6.4. Lexicon structure and error correction technique are described in Section 6.5 and 6.6, respectively. Lastly, results and discussion are presented in Section 6.7 followed by conclusion.

6.2 A review on error detection and correction

There are two major approaches of error detection namely dictionary look-up and n-gram matching in which either a lexicon file or a database of legal n-grams is used to locate one or more errors. In dictionary look-up method the test string is searched in the dictionary and if no match is found, an error is reported. In n-gram approach, the n-grams of the test string are found. If all n-grams are there in the system database then the string is considered as a valid word. Else, an error is reported. An erroneous string may be called misspelled string.

Several approaches are reported for the generation of candidates for word error

correction. Major among them are minimum edit distance techniques, similarity key techniques, rule based techniques, n-gram based techniques, probabilistic techniques and neural networks based techniques [88,151].

In the first technique the candidates are those valid words which have minimum edit distance from the misspelled string. The minimum edit distance is the minimum number of editing operations (i.e. insertions, deletions and substitutions) required to transform one string into another. The minimum edit distance spelling correction technique was initially developed by Damerau [37] and Levenshtein [93]. Modifications of the basic techniques are proposed to speed up the error correction effort. For example, Takahashi et al. [160] proposed a technique for English language that can correct the OCR error as well. Their method consists of two steps: selection of candidate words and approximate string matching between the input word and each candidate word. In their approach of candidate selection, multiple indexes are generated from combinations of a constant number of infrequent characters of the input word, and using these indexes candidate words are selected from the dictionary. They proposed a simple an efficient distance measure to select a few alternatives for the correct word. Wagner [170] introduced dynamic programming minimum edit distance techniques for error correction. A 'reverse' minimum edit distance method was used for the error correction by Church and Gale [34]. In this technique, a candidate set is produced by first generating every possible single-error permutation of the misspelled string and then checking the dictionary to see if any make up valid words.

In a similarity key based technique the idea is to map similarly spelled strings into identical or similar key. When a key is computed for a misspelled string, it provides a pointer to all similarly spelled words in the lexicon which may be accepted as the candidates [125]. This technique is efficient since it is not necessary to directly compare the input to all words of the lexicon.

To transform misspellings into valid words, a set of rules derived from the

knowledge of common spelling error pattern is used in a rule based technique [183].

Character n-grams such as bigram and trigram are used for error detection, candidate selection and ranking. Angell *et al.* [2] proposed an approach where the number of non-positional binary trigrams common to a misspelled string and a lexicon word are computed. A similarity measure is defined and computed for only that subset of lexicon words which have at least one trigram common to the misspelled string. These candidates can be ranked according to the similarity value. A clear explanation of the traditional use of n-grams in OCR correction has been provided by Riseman and Hanson [133]. They partitioned the dictionary into subdictionaries and constructed positional n-gram matrices for each subdictionary. Presently, hardware based techniques have been proposed that would exploit n-gram databases in parallel [168,65].

Probabilistic models have long been used for OCR error correction. Two types of probabilities namely transition probability and confusion probability have been exploited for the purpose. If n-grams are computed from a large corpus, they provide good estimates of the transition probabilities. Confusion probability represents how often a given character is mistakenly replaced by another character. The confusion probability depends on the source document. For example, an OCR device can generate different confusion probability distributions for different font types of the same alphabet. Using confusion matrix of character shapes Tanaka *et al.* [161] proposed two methods for high speed string correction of OCR.

Bayes' rule has been used by Bledsoe and Browning [10] in text error correction where the valid word maximizing aposteriori probability is considered as the correct word. The computational complexity of the approach is linear with the dictionary size and hence not practical with large dictionary. Kahan *et al.* [74] have combined confusion probabilities with dictionary look up to speed up the process. Also, transition and confusion probability are combined in a dynamic programming technique called *Viterbi algorithm* [144] for OCR error correction. However, a model

based on probabilistic information alone does not yield high degree of error correction. Rather, it is noted that combining dictionary information with probability can lead to high correction accuracy.

Other techniques such as neural network have been used for isolated word error correction. Kukich [87] as well as Cherkassky and Vassilas [31] used backpropagation algorithm for name correction applications. Techniques for reducing training time by partitioning the lexicon or by exploiting alternative network architectures are being explored [57].

Work done on English and other foreign languages is not well-suited to Indian languages for several reasons. Indian languages are mostly inflectional. Abundance of suffix, euphony and assimilation makes the dictionary compilation a non-trivial issue. Creative morphology is another important problem whereby new words are being added to the language.

Dictionary look-up is a standard way of checking non-word error [37]. The storage requirement and the look-up speed depend on the size of the lexicon. The lexicon should be tuned to the intended domain of applications. A small lexicon may lead to many false rejections of valid words. A large lexicon may increase the possibility of overlooking real word errors.

Our error detection and correction approach is based on dictionary search [26]. As mentioned before, there are about 300 character shapes in Bangla. Hence, assuming a character can sit at any position, the number of bigram entities will be about 300×300 and the number of trigram entities will be about $300 \times 300 \times 300$. Generation, storage and handling of such a huge number of entities is very difficult as well as time consuming. Also, as mentioned in Section 2 of Chapter 2, some modifiers have two parts, one sitting to the left and other to the right of the consonant (or compound) character. So in such cases, graphemically we get two bigrams though it was actually one.

Because of the problems stated above we have adopted a dictionary look-up approach for the error detection and correction of our Bangla OCR output. The dictionary based approach is partly justified by the statistical result (given in chapter 2) that confusing characters rarely make real word errors.

6.3 OCR error pattern

According to the structural shape of Bangla characters as well as the OCR technique used, the error can be classified into five categories: substitution, rejection (failure to recognize a character), run on (where two consecutive characters are misrecognized as a single character), split character (where one character is segmented into two parts, of which one or both parts are either misrecognized or unrecognized) and deletion error (where two consecutive characters are misrecognized as one of these characters). However, it is unlikely to have transposition and insertion error for Bangla text.

As stated above, it is assumed that the original document contains no error i.e. all words are valid surface words. The error distribution of our OCR system without error correction module is shown in Table-6.1. The system makes an average of 3.52% character recognition error and 14.22% word recognition error. Note that out of 14.22% word recognition errors 12.29% are due to single character error (i.e. misrecognized or rejected character). We have also noticed that 29% of the erroneous single characters belong to compound character class while the rest belong to modifier and basic character class. These results have been computed from 20,000 OCR recognized words.

We have noted that most of the rejected characters are compound characters. As mentioned earlier, the OCR classifier is a tree classifier and hence it is possible to memorize the subgroup to which a test character belongs before rejection at the template matching stage so that error correction could be limited to that subgroup

of characters. Also, we have noted that substitution error for a character can be one among at most 4 other characters. In case of rejection, a rejected compound character can belong to a group of at most 15 compound characters while a rejected basic character can be one among at most 4 basic characters.

6.4 Error detection approach

The OCR system output can be of two types. In the first type, the *test string* may contain one or more rejected characters. Such a string may be called *partially recognized string*. Any such string is an erroneous word and hence it need not be subjected to error detection procedure. In the second type, we may have a *fully recognized string*. Error detection is necessary for such a test string.

As mentioned before, our error detection approach is based on dictionary look-up. However, the dictionary does not contain all the surface words that may be found in the corpus. So, to check if a test string is a valid surface word or not, it is necessary to check if the string is a root word, or if it contains a valid root word followed by a suffix that grammatically agrees with the root word. The notion of grammatical agreement is explained later on in this section.

Let W be a string of n characters $x_1x_2.....x_n$. If W_1 is a string $x_1x_2.....x_n$ and if W_2 is another string $r_1r_2.....r_p$ then by $W_1 \oplus W_2$ we denote the concatenation of the two strings i.e. $W_1 \oplus W_2 = x_1x_2.....x_nr_1r_2.....r_p$. Let $\mid W \mid$ denote the number of characters in string W. Also, if a substring W_s of W contains x_1 (i,e first character of the string) then W_s is called *front substring*. Similarly, W_s is a *back substring* if W_s contains x_n (i,e last character of the string). Also, we can 'dissociate' a substring W_s of W from W. Dissociation is deletion of substring W_s from left or right of W. Thus, dissociation of W_s from W namely $W \ominus W_s = W_r$ is such that $W = W_s \oplus W_r$ if W_s is a front substring, while $W_r \oplus W_s$ if W_s is a back substring.

If W is a valid surface word then according to our simplified word morphology given in Section 2.2 of Chapter 2, it is possible that

either (i) W is a root word

or (ii) A substring $R_1 = x_1x_2.....x_r$ of W is a root word and another substring $S_1 = x_{r+1}x_{r+2}.....x_n$ of W is a suffix which grammatically agrees with R_1.

If an arbitrary string W satisfies one of the above two conditions then it may be declared as a valid surface word. The segmentation of W into root word and suffix (including null) satisfying these conditions may be called *valid parsing*.

Normally, a valid surface word can be parsed into *one* root word and *one* suffix (including null). Rarely, however, a string can have multiple valid parses. However, our aim is to get only one valid parse of W so that it can be passed as error-free word.

In order to parse an arbitrary string W, we use two separate lexicons namely, root word lexicon and suffix lexicon. To search for a valid root word substring, we can start from leftmost character (i,e beginning with x_1) and move rightwards. In general, we may have multiple substring matches $R_1,R_2,.....R_k$ so that $|R_k| > |R_{k-1}| > ..... > |R_1|$ in the root word lexicon. The distribution of such valid root word substrings is shown in Table-6.2. This table shows that most of the surface words have one valid root word substring. Similarly, to search for a suffix substring, we can start from the rightmost (i,e beginning with x_n) and move leftwards. Here too, we may have multiple substring matches $S_1,S_2,.....S_m$ in the suffix lexicon so that $|S_m| > |S_{m-1}| > ..... > |S_1|$. The distribution of valid suffix substrings is shown in Table-6.3. From the table it is seen that most of the surface words of length less than 5 have one valid substring and those of length greater than or equal to 5 have two valid substrings.

The situation for a typical string is shown in Fig.6.1. Note that all root-suffix pair are not *candidates* for parsing the string. For example, in Fig.6.1 the pair $\{R_2, S_m\}$ is not a candidate since $R_2 \oplus S_m \neq$ W (here $R_2 = x_1x_2x_3x_4$ and

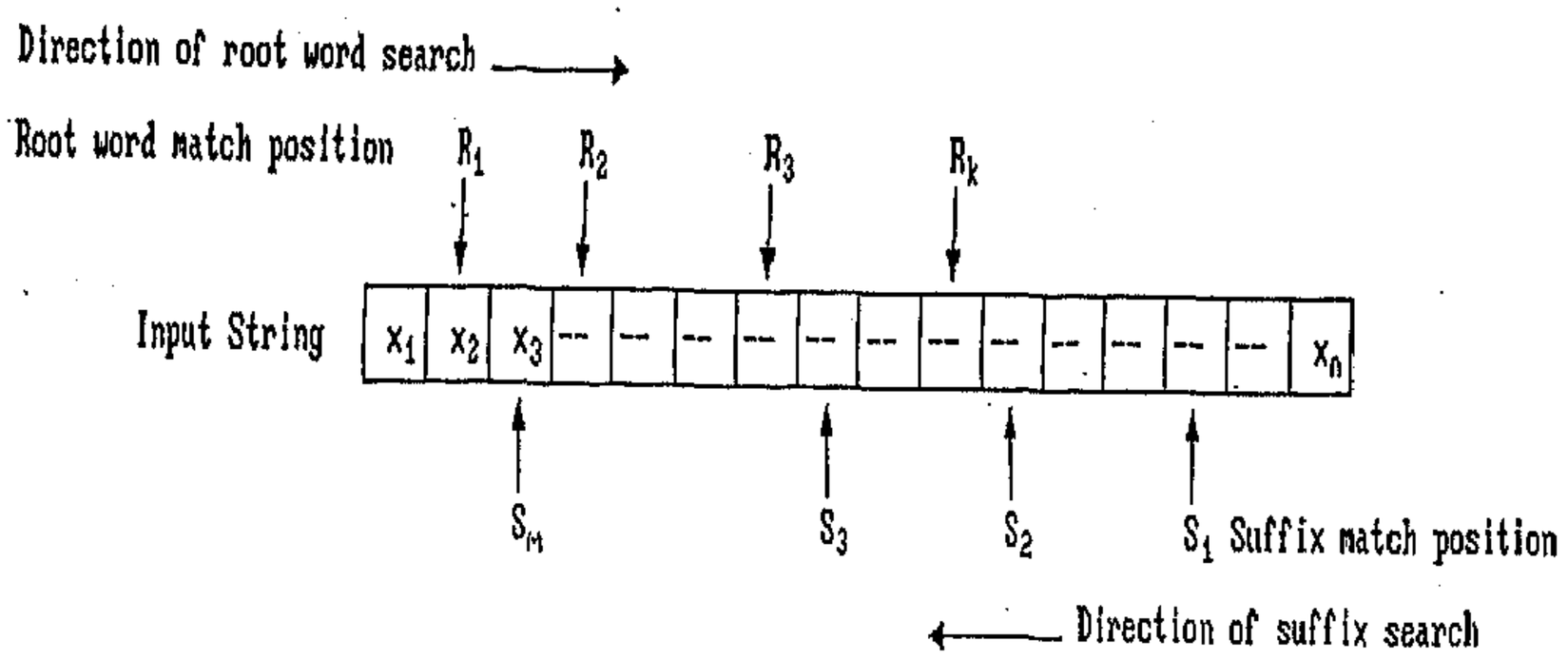


Figure 6.1: An example of character string parsing.

$S_m = x_3x_4x_5....x_n$). The pair $\{R_1, S_m\}$, $\{R_3, S_3\}$ and $\{R_k, S_2\}$ are only candidates for parsing. Now, if W is a valid surface word then at least one of these candidate pair has a valid parse. Distribution of candidate root-suffix pair is shown in Table-6.4. It shows that maximum number of candidate root-suffix pair substrings of a word is 3 and most of the suffixed surface words have only one root-suffix pair. These statistics have been computed from a corpus of 60000 surface words.

Now, we explain the notion of *grammatical agreement.* A root word cannot concatenate with arbitrary suffix to form a valid surface word. The valid concatenation is primarily governed by the *parts of speech* of the root word. For example, if the root word is a 'verb', then it can concatenate only with 'verb suffixes' to form a valid surface word. Thus, if $\{R_i, S_j\}$ is a candidate root-suffix pair and if R_i is a 'verb' then $\{R_i, S_j\}$ is not a valid parse if S_i is not a 'verb suffix'.

Suppose V denotes the set of all possible 'verb suffix' strings. A string R_i which is a verb root substring can concatenate with most but not all of the members of V to form valid surface words. The subset of unacceptable verb suffixes for R_i may be called U(R_i). A verb root string R_i *grammatically agrees* with a verb suffix string S_j if S_j does not belong to U(R_i).

We can check for grammatical agreement, if there are parts of speech markers with each root word and each suffix in the respective lexicons. In addition, information about unacceptable subset of suffixes should also be there for each word in the root word lexicon.

Thus, our algorithm for error detection is as follows:

Step 1: Consider a new string W. Start from left and find all root word sub-strings R_1,R_2 - - -R_k in W.

Step 2: If R_k = W, go to step 5. Else, start from right and find a valid suffix sub-string S.

Step 3: If $W \ominus S = R_i$, for any i, then call the grammatical agreement test. If the test is passed then go to step 5.

Step 4: Find the next valid suffix sub-string. If no such sub-string exists then go to step 6. Else, go to step 3.

Step 5: Report success (the string is a valid surface word). Go to step 1.

Step 6: Report failure. Call error correction algorithm and go to step 1.

6.5 Lexicon structure

As mentioned earlier, there are two lexicons in our systems : one for the root word and other for the suffix. The root word lexicon contains 60,000 valid root strings and the suffix lexicon contains only 800 suffix strings. The root word lexicon is organized as follows. It contains all root words as well as morphologically deformed variants of the root words (mostly verbs) so that simple concatenation with suffix make valid surface words. Words are kept in the main lexicon in alphabetical order along with the markers of the parts of speech and other grammatical informations.

Some of the morphologically deformed variants in the lexicon may not be used in isolation as a root word in the text. To detect such words another marker is also placed against the entries in the lexicon.

Root words of length upto 3 characters as well as the the distinct strings of first 3 characters of root words of length greater than 3 are represented in a trie structure. At each leaf node of the trie an address as well as a boolean flag is maintained. Address of a leaf node L_k of the trie points to the first word of the main lexicon, of which the first three characters are the same as those we encounter by traversing the trie, starting from the root node, upto that leaf node L_k. So, if a string is of length four or more (characters) then we can obtain the address from trie using the first three characters and then sequentially search the main lexicon to see if the string is a valid word. The lexical structure is shown in Fig.6.2. Our observation claims that the sequential search on an average requires five words to be checked for each input string where the lexicon size is 60,000 words.

The boolean flag of a leaf node L_k indicates whether the string of the characters obtained by traversing the trie, starting from the root node, upto L_k is a valid root word or not. Thus, by boolean flags, we can detect the validity of test strings whose length are less than or equal to 3, without any search in the root word lexicon.

The suffix lexicon is maintained in another trie structure. Each node in the trie either represents a valid suffix or an intermediate character of a valid suffix. A node representing a valid suffix is associated with a suffix marker and the parts of speech marker. If one traverses from the root of the trie to a node having valid suffix marker then the encountered character sequence represents a valid suffix string in reverse order (i.e. last character first). This is so because during suffix search we start from the last character of a string and move leftwards.

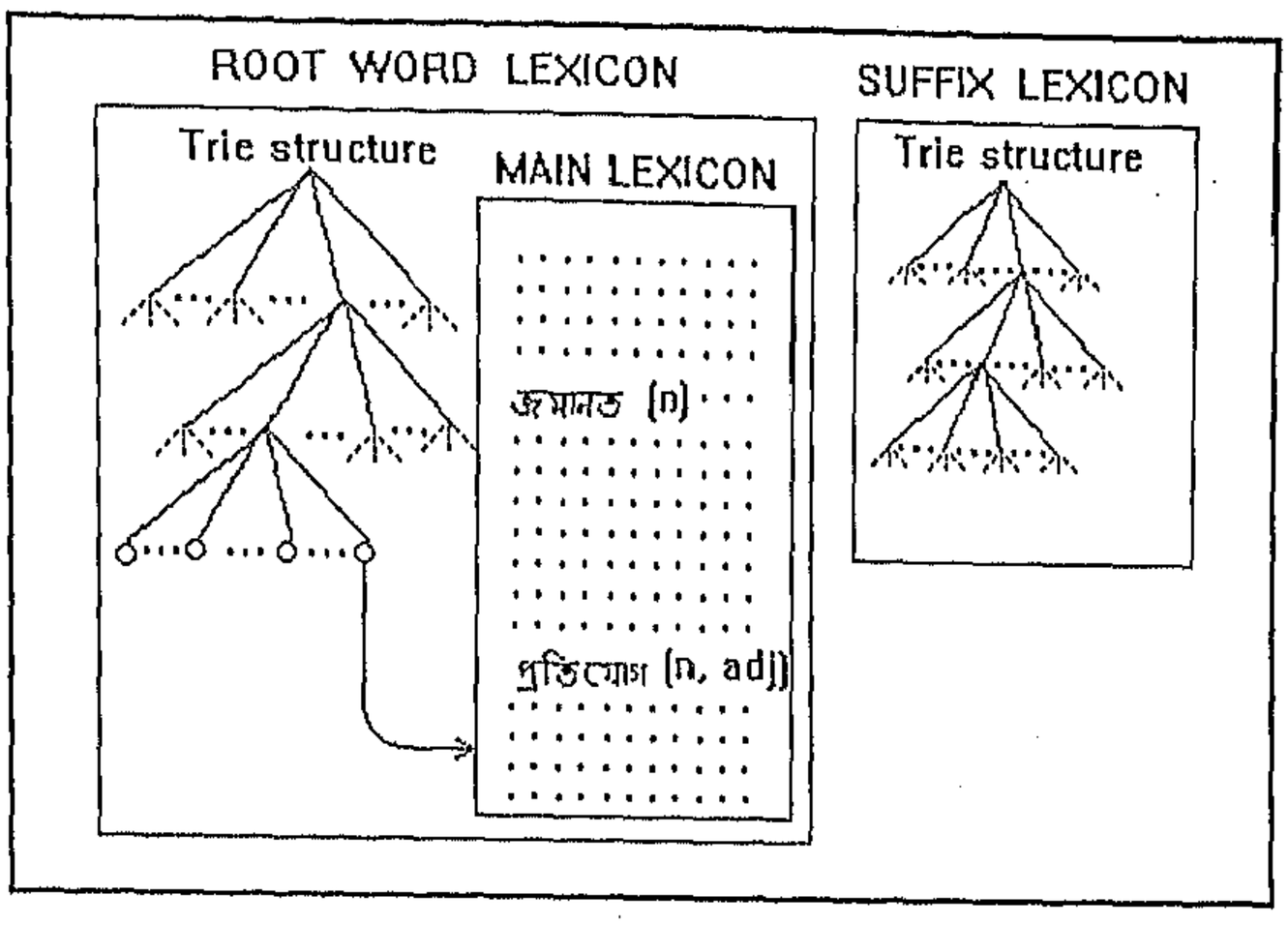


Figure 6.2: Error correction lexicon structure. (See examples in main lexicon where parts of speech markers are given in parenthesis).

6.6 Error correction approach

Because of the complexity of the surface words in Bangla language and because the error in our OCR output is mostly single character per word, our effort is limited to correcting single character in a word. Note that, single character includes compound character which is the graphemic combination of two or more basic characters.

Now, for error correction, we are left with *fully recognized string* which is an erroneous word according to the error detection strategy, or a *partially recognized string* with only one un-recognized character. At first, let us consider *fully recognized string*.

The information about all valid root substrings R_1, R_2, ... R_k and all valid suffix substrings S_1, S_2....S_m are recalled from the error detection program module. Now, the four situations that may arise and their corresponding error hypotheses are described below :

Case-1: $k = 0$, $m = 0$, i.e. no substring match is obtained either in root word lexicon or in suffix lexicon. Consider W as an incorrect root word and correct it using only root word lexicon.

Case-2: $k = 0$, $m > 0$, i.e. no substring match is obtained in root word lexicon but one or more suffix match are found in suffix lexicon (during error detection). It is assumed that one of these suffixes is correct and the error occurs in root word part only. For each suffix S_i find the substrings R'_i so that $R'_i \oplus S_i = W$ and correct R'_i using root word lexicon. Look for grammatical agreement with S_i and accept/reject the corrected string.

Case-2a: If agreement is not obtained for any corrected substring then correct the complete string W using root word lexicon.

Case-3: $k > 0$, $m = 0$, i.e. substring match is obtained only in the root word lexicon. It is assumed that one of the root word strings is correct and the error occurs

in suffix part only. Find the substrings S'_i so that $R_i \oplus S'_i = W$ and correct S'_i using the suffix lexicon. Look for grammatical agreement with R_i as above.

Case-3a: If agreement is not obtained for any of the corrected S_i then correct W using root word lexicon.

Case-4: $k > 0$, $m > 0$, i.e. matches are obtained in both lexicons. It is assumed that one of the root word strings is correct (i.e. suffix match is disregarded) and correction is done as in case (3) above.

Note that we have emphasized correction in suffix for case 3 and case 4 because it is quicker to search suffix lexicon. It is possible to reduce computation by neglecting some of the above possibilities (for example, we could not consider computation of Case 2a, 3a and reject W as incorrigible).

Let us now consider the *partially recognized string* which is not subject to error detection procedure. The string W in this case is subject to both root word and suffix substring detection. Here too, the four situations described above may occur. The error hypotheses are also identical. In each case, however, it is guaranteed that the substring to be corrected contains the unrecognized character.

Whatever might be the situation, either W or its substring should be corrected either for a root word or for a suffix before testing for the grammatical agreement. The correction is done as follows.

Let C be a character in the alphabet. Let D(C) denote the subset of characters which may be wrongly recognized as C. Members of D(C) may be called confusing characters for C. Now, if a recognized string W contains a character C and if it is a misrecognized character then the correct word for W is a string among the strings obtained when C is replaced by elements of D(C). Such strings can be called *candidate strings*. We can attach a probability to each candidate string depending on the apriori knowledge of occurrence frequency of a confusing character.

Since we do not know which character in W is misrecognized, we have to generate candidate strings by replacing each character in W by its confusing characters. In generating the candidates we should take care of run on, split and deletion errors also. A complete set of candidates is thus generated.

According to the performance of our OCR, the number of candidates per character rarely exceeds 4. With an average wordlength of 5 characters, in the worst case, the number of candidate strings per word can be 20. For a partially recognized string with a single unrecognized character, note that our OCR knows if it is a basic or compound character. For a basic character the number of candidates is 4 while for a compound character it is about 10 on an average.

The candidate set thus formed, is matched in decreasing order of probability against either root-word lexicon or suffix lexicon. As soon as a match is found, it is accepted. If no member of the set matches with an entry of the lexicon then the substring is rejected.

6.7 Results and discussion

We have noted that our error correction module can correct 74.22% of single character error generated by the OCR system. As a result, the overall accuracy of the OCR system is 94.87% in word level while 98.76% in character level. Out of 25.78% single character error in correction effort 14.73% are due to rejection and the 11.05% are due to un-intended word. We noted that the overall degradation in the error correction performance is mainly due to compound characters.

We have also noted that out of 14.22% word error (shown in Table-6.1) 13.80% are non-word and 0.42% are real word errors (which cannot be corrected). Moreover, note from the Table-6.1 that 1.93% words have more than one error which cannot be corrected by the approach.

The effect of different sizes of single font characters on the error correction system is observed. Final word recognition rates on different point size documents (from 8 to 16) are given in Table-6.5.

Our error correction system puts an identification mark for incorrigible words. The advantage of such marking is that the words can be easily identified for manual correction.

6.7.1 Conclusion

Error detection and correction of inflectional Indian languages is very difficult problem. To the best of our knowledge, there is no published work dealing with this problem. Here, we have developed an OCR single error detection and correction approach for Bangla language. Work on multiple errors detection and correction is in progress. Our error detection and correction technique can be applied to other inflectional Indian languages as well.

Table 6.1: Character and word recognition error of the proposed OCR system.

Character recognition error	3.52%
Word error due to single misrecognized character	9.17%
Word error due to single rejected character	3.12%
Word error of other types (Double character error, Run on error, Splitting error, Deletion error etc.)	1.93%
Total word recognition error	14.22%

Table 6.2: Probability estimates of number of valid root word substrings.

WORD LENGTH	Number of Candidate Root Word substrings				
	1	2	3	4	5
1	1.0000	0.0000	0.0000	0.0000	0.0000
2	0.9086	0.0914	0.0000	0.0000	0.0000
3	0.7281	0.2562	0.0156	0.0000	0.0000
4	0.6296	0.2921	0.0742	0.0041	0.0000
5	0.6101	0.3026	0.0760	0.0109	0.0004
6	0.6443	0.3379	0.0959	0.0215	0.0004
7	0.5435	0.3242	0.1067	0.0244	0.0012
8	0.5340	0.3376	0.1059	0.0224	0.0000
9	0.5463	0.3149	0.1121	0.0267	0.0000
10	0.5090	0.3177	0.1191	0.0542	0.0000
11	0.5470	0.3248	0.0855	0.0427	0.0000
12	0.6400	0.2600	0.0800	0.0200	0.0000
13	0.4400	0.4000	0.0800	0.0800	0.0000
14	0.4000	0.6000	0.0000	0.0000	0.0000

Table 6.3: Probability estimates of valid suffix substrings of words.

WORD LENGTH	Number of candidate suffix substrings						
	1	2	3	4	5	6	7
1	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.9477	0.0523	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.6712	0.3102	0.0187	0.0000	0.0000	0.0000	0.0000
4	0.5404	0.3829	0.0739	0.0028	0.0000	0.0000	0.0000
5	0.3390	0.5321	0.1146	0.0143	0.0000	0.0000	0.0000
6	0.3211	0.4563	0.1903	0.0310	0.0013	0.0000	0.0000
7	0.3029	0.4695	0.1545	0.0648	0.0074	0.0009	0.0000
8	0.2902	0.4797	0.1660	0.0583	0.0051	0.0006	0.0000
9	0.2349	0.4908	0.1710	0.0800	0.0160	0.0062	0.0012
10	0.2276	0.4639	0.2276	0.0766	0.0044	0.0000	0.0000
11	0.2265	0.4530	0.2320	0.0829	0.0055	0.0000	0.0000
12	0.1204	0.4630	0.3055	0.1111	0.0000	0.0000	0.0000
13	0.1667	0.4048	0.2619	0.1667	0.0000	0.0000	0.0000
14	0.0000	0.3077	0.6154	0.0769	0.0000	0.0000	0.0000
15	0.2500	0.5000	0.2500	0.0000	0.0000	0.0000	0.0000
16	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000

Table 6.4: Probability estimates of candidate root-suffix pair substrings.

WORD LENGTH	Number of valid root-suffix pair		
	1	2	3
2	1.0000	0.0000	0.0000
3	0.9820	0.0180	0.0000
4	0.8892	0.1108	0.0000
5	0.8703	0.1250	0.0047
6	0.9199	0.0736	0.0065
7	0.9335	0.0666	0.0000
8	0.9800	0.0200	0.0000
9	0.9825	0.0175	0.0000
10	1.0000	0.0000	0.0000
11	1.0000	0.0000	0.0000
12	1.0000	0.0000	0.0000
13	1.0000	0.0000	0.0000

Table 6.5: Word recognition rate on different point size documents.

Point size	8	10	12	14	16
Recognition rate	93.90%	95.65%	95.70%	95.19%	94.21%

Chapter 7

CONCLUSION

7.1 Contributions of the present thesis

This thesis has two major contributions: (1) A study of graphemic character occurrences in words of Bangla language and (2) an OCR system development to read printed Bangla script.

At first, this thesis deals with the study of graphemic character occurrences in words of Bangla language. These occurrence statistics of characters are presented for words collected from Bangla books, newspapers, magazines, novels and juvenile literatures. Some of the computed statistics are the occurrence percentage of vowels/consonant (unigram). Positionwise occurrences of the vowel/consonant, percentage of compound character, percentage of vowel modifiers, confusing character statistics, bigram, trigram etc. The application potentials of these computed statistics are also described in this thesis.

Later, it deals with a complete OCR system to read printed Bangla script for single font. This is the first OCR work among all script forms used in Indian sub-continent. The main studies that have done for the OCR are as follows.

A fast and robust skew angle detection technique based on the characteristics of the language alphabet has been proposed. The method has additional advantage that the headline, which is important in future stage, is already detected during skew correction.

Since most of the characters of a word are connected through headline, it is necessary to segment the characters for recognition. We have proposed a robust algorithm for segmentation of characters from words. Next, basic characters, modifiers and compound character separation approach has been presented using zonal information and shape characteristics. For basic characters and modifiers recognition a feature based tree classifier has been used. Because of complex shape of compound characters and because of their large number, a hybrid approach has been described for their recognition. In the first stage of this hybrid approach the characters are grouped into small subsets by a feature based tree classifier and in the second stage characters in each group are recognized by a sophisticated run based template matching approach.

Since Bangla is a highly inflectional language, where 70% of the words are inflected and since Bangla is not purely alphabetical script, most of the existing error detection techniques are not suitable for Bangla. We used a dictionary based approach where root words and suffixes are maintained in separate dictionaries. The approach is quite effective and efficient to improve the OCR output.

7.2 Scope of future work

The work reported in this thesis can be extended in several directions. Some of them are listed below:

1 **Improvement of error correction approach:**

Our OCR error correction module can correct single character error only. Hence, the full potential of error correction in improving the OCR accuracy has not been exploited. Word error correction is a difficult and challenging task in an inflectional language like Bangla. However, if the task is accomplished, it can not only improve the OCR performance but also can lead to a good spell checker in Bangla, which is not available anywhere.

2 **OCR for poor quality documents:**

We have developed a system which is satisfactory for good quality documents. Elaborate study on poor quality document was not undertaken. Experiments should be made to see the effect of poor quality paper, printing as well as noise of various types and take corrective measures [131,132].

3 **Development of multi-font OCR:**

We have developed a single-font Bangla OCR system. Although font variations in Bangla documents are small in number, it is useful to develop a multi-font system. To achieve this goal the compound character recognition technique should be less dependent on template matching. Work has been started in our laboratory to find such a technique.

4 **OCR of other Indian language:**

Many Indian scripts including Bangla and Devnagari (Hindi) are derived from the same ancient script, Brahmi. Bangla and Devnagari scripts have a lot of common graphemical features including the presence of headline and vertical strokes. The characters in the alphabet have one to one correspondence and the corresponding characters have the same phonetic name in Bangla and Devnagari. Devnagari also has a similar set of compound characters. Moreover Hindi, like Bangla, is an inflectional language.

These features suggest that our approaches for Bangla OCR can be used for Devnagari OCR with little modifications. We have made a preliminary study of applying our method for the OCR of Devnagari script [28]. We have noted that skew correction and zone separation as well as line, word and character segmentation modules of our Bangla OCR can be directly used for Devnagari. The basic and compound character separation can also be done in somewhat similar way. Also, the decision tree classifier scheme suits well for Devnagari, although some of the stroke features are not identical with those used for Bangla.

We believe that our work will stimulate OCR activities of other major Indian language scripts. Some work on Telugu script OCR has been initiated in the Institute in collaboration with our laboratory.

5 Bi-lingual/multi-lingual script document OCR:

Since India is a multi-lingual country, it is instructive to develop multi-lingual OCR system. To start with, one can have a three language script OCR with Bangla, Devnagari and English in the same document. These three scripts are most popular in India. Here, the first task is to separate the three classes of script before feeding them to individual OCR system of each script. A preliminary work has been done in our Laboratory for separation of these scripts [120]. It was noted that English can be easily distinguished but Bangla and Devnagari may get confused with each other.

6 Hand-printed OCR system development:

In future, there may be more demand for OCR on hand-printed documents. One particular case may be table-form document where hand-printing should be done within specified bounding box. Some work has been started in our laboratory and it was found that recognition of such a constrained hand-printing is a difficult task for Bangla script because of the vowel modifiers and compound characters.

7 OCR for the visually handicapped:

One of the primary motivations of early development of OCR system was a reading aid for the visually handicapped, as noted in the work of Tyrin [97]. This line of activities is continued by Kurzweil [90], among others. In India too there is a great need for reading aid for the blind. One possible way of achieving this goal is to convert the OCR output into speech format. Some work on speech synthesis has been done in our lab. The speech synthesis is done by diphone concatenation approach where only 750 diphones are sufficient for the purpose. Prior probabilities of the diphones are used to make the synthesis efficient in time. Some utterance rules are used for grapheme to phoneme conversion in the OCR recognized word. A word having rejected character(s) is uttered character by character. The speech output is quite intelligible but somewhat flat. We are adding the stress and intonation rules to make the speech more lively.

Finally, we beleive that our work will encourage activities of automatic document processing in general and OCR of Indian language scripts, in particular.

Bibliography

[1] T. Agui and N. Nagahashi, 'A description method of hand-printed Chinese characters', *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 1, pp. 20-24, 1979.

[2] R. C. Angell, G. E. Freund, and P. Willett, 'Automatic spelling correction using a trigram similarity measure', *Inf. Process. Manage*, vol. 19, pp. 255-261, 1983.

[3] T. Akiyama and N. Hagita, 'Automatic entry system for printed documents', *Pattern Recognition*, vol. 23, pp. 1141-1154, 1990.

[4] H. I. Avi-Itzhak, T. A. Diep and T. A. Garland, 'High accuracy optical character recognition', *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 17, pp. 218-225, 1995.

[5] H. S. Baird, 'The skew angle of printed documents', *In Proc. of Society of Photographic Scien. Engg.*, Rochester, NY, vol. 40, pp. 21-24, 1987.

[6] H. S. Baird, 'Anatomy of a versatile page reader', *Proceedings of the IEEE*, vol. 80, pp. 1059-1165, 1992.

[7] M. Berthod, *On line analysis of cursive writing, computer analysis and perception*, vol. 1, Visual signals, C. Y. Suen and R. D. Mori Eds. CRC Press, Cleveland, Ohio, U.S.A. 1982.

[8] N. Bhattacharia, 'Some statistical studies of languages', *Ph.D thesis, Indian Statistical Institute*, Calcutta, 1965.

[9] S. Biswas, S. Dasgupta and D. Bhattacharia, 'Samsad Bangla Abhidhan', *Sahitya Samsad*, Calcutta, 1992.

[10] W. W. Bledsoe and I. Browning, 'Pattern recognition and reading by machine', *In Proc. of the Eastern Joint Computer Conference*, vol. 16, pp. 225-232, 1959.

[11] B. Blesser, R. Shillman, T. Kuklinski, C. Cox, M. Eden and J. Ventura, 'A theoretical approach for character recognition based on phenomenological attributes', *In Proc. of 1st Int. Conf. on Pattern Recognition*, Washington, pp. 33-40, 1973.

[12] M. Bokser, 'Omni-document technologies', *Proceedings of the IEEE*, vol. 80, pp. 1066-1078, 1992.

[13] R. M. Bozinovic and S. N. Srihari, 'Off line cursive script word recognition', *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 11, pp. 68-83, 1989.

[14] D. J. Burr, 'Experiments with a connectionist text reader', *Proc. IEEE Int. Conference on Neural Networks*, San Diego, Calif., New York, vol.4, pp. 717-724, 1987.

[15] T. W. Calvert, 'Non-orthogonal projections for features extraction for pattern recognition', *IEEE Trans. on Computer*, vol. 19, pp. 447-452, 1970.

[16] R. Casey and E. Lecolinet, 'A survey of methods and strategies in character segmentation', *IEEE Trans. on Pattern Analysis and Machine Intell.*, vol. 18, pp. 690-706, 1996.

[17] R. Casey and G. Nagy, 'Recognition of printed characters', *IEEE Trans. on Electron, Computer*, vol. 15, pp. 91-101, 1966.

[18] R. Casey and G. Nagy, 'Automatic reading machine', *IEEE Trans. on Computer*, vol. 17, pp. 492-503, 1968.

[19] R. Casey and G. Nagy, 'Decision tree design using a probabilistics model', *IEEE Trans. on Information Theory*, vol. IT-30, pp. 93-99, 1982.

[20] G. L. Cash and M. Hatamian, 'Optical character recognition by the method of moments', *Computer Vision Graphics and Image Processing*, vol. 39, pp. 291-310, 1987.

[21] M. Chandrasekaran, R. Chandrasekaran and G. Siromony, 'Context dependent recognition of hand-printed Tamil characters', *In Proc. Int. Conf. System. Man Cyb.,(India)*, vol. 2, pp. 786-790, 1984.

[22] R. Chandrasekaran, M. Chandrasekaran and G. Siromony, 'Computer recognition of Tamil, Malayalam and Devanagari characters', *J. Inst. Electron. Telecom. Engg.(India)*, vol. 30, pp. 150-154, 1984.

[23] S. K. Chatterjee, *The origin and development of Bengali language*, University of Calcutta, 1920.

[24] B. B. Chaudhuri and D. Dutta Majumder, *Two-tone image processing and recognition*, Wiley Eastern Limited, New-Delhi, 1993.

[25] B. B. Chaudhuri and U. Pal, 'Relational studies between phoneme and grapheme statistics in modern Bangla language', *Journal of the Acoustical Society of India*, vol. 23, pp. 67-77, 1995.

[26] B. B. Chaudhuri and U. Pal, 'OCR error detection and correction of an inflectional Indian language script', *In Proc. 13th Int. Conf. on Pattern Recognition*, vol. 3, pp. 245-249, 1996.

[27] B. B. Chaudhuri and U. Pal , 'Skew angle detection of digitized Indian script documents', *IEEE Trans. on Pattern Analysis and Machine Intell.*, vol. 19, pp. 182-186, 1997.

[28] B. B. Chaudhuri and U. Pal, 'An OCR system to read two Indian language scripts: Bangla and Devnagari', *4th Int. Conf. on Document Analysis and Recognition*, Ulm, Germany, 18-20 August, 1997 (accepted).

[29] B. B. Chaudhuri and U. Pal, 'A complete printed Bangla OCR system' *Pattern Recognition* (in Press).

[30] Y. S. Cheng and C. H. Leung, 'Chain code transform for Chinese character recognition', *In Proc. Int. Conf. Cyb. Soc.*, Tucson, AZ, U.S.A., pp. 42-45, 1985.

[31] V. Cherkassky and N. Vassilas, 'Back- propagation networks for spelling correction', *Neural Networks*, vol. 1, pp. 166 - 173, 1989.

[32] Z. Chi, J. Wu and H. Yan, 'Hand-written numeral recognition using self-organizing maps and fuzzy rules', *Pattern Recognition*, vol. 28, pp. 59-66, 1995.

[33] P. Chinnuswamy and S. G. Krishnamoorty, 'Recognition of hand-printed Tamil characters', *Pattern Recognition*, vol. 12, pp. 141-152, 1980.

[34] K. W. Church and W. A. Gale, 'Enhanced Good-Turing and cat-cal: Two new methods for estimating probabilities of English bigrams', *Computer Speech Language*, 1991.

[35] R. W. Cornew, 'A statistical method of spelling correction', *Inform. Control*, vol. 12, pp. 79-93, 1968.

[36] C. Cox, B. Blesser and M. Eden, 'The application of type font analysis to automatic character recognition', *In Proc. 2nd Int. Conf. on Pattern Recognition*, Copenhagen, pp. 226-232, 1974.

[37] F. J. Damerau, 'A technique for computer detection and correction of spelling errors', *Commun. of the ACM*, vol. 7, pp. 171-176, 1964.

[38] G. Das, S. Bhattacharya and S. Mitra, 'Representing Ohamia, Bengali and Manipuri text in the daisy-wheel printer',*J. Inst. Electrical and Telecom-Engg.*, vol. 30, pp. 251-256, 1984.

[39] R. W. Donaldson and G. T. Toussaint, 'Use of contextual constraints in recognition of contour-traced hand-printed characters', *IEEE Trans.*

on Computer, vol. c19, pp. 1096-1099, 1970.

[40] A. K. Dutta, 'A generalized formal approach for description and analysis for major Indian scripts', *J. Inst. Elec. telecom. Engineering (India)*, vol. 30, pp. 155-161, 1984.

[41] A. K. Dutta and S. Chaudhuri, 'Bengali alpha- numeric character recognition using curvature features', *Pattern Recognition*, vol. 26, pp. 1757-1770, 1993.

[42] A. K. Dutta 'Rule based approach to labeling of speech wave in multi-syllabic words', *In Proc. Int. Workshop on Speech Processing*, TIFR, Bombay, 1988.

[43] M. Eden, *Hand-written generalization and recognition, recognizing patterns*, Kolers and M. Eden, Eds, pp. 138-154, M. I. T. Press, Cambridge, MA, 1968.

[44] A. W. Edwards and Robert L. Chambers, 'Can a priori probabilities help in character recognition ?', *Journal of the Association for Computing Machinery*, vol. 2, pp. 465-470, Oct. 1964.

[45] R. W. Ehrich, 'A contextual post-processor for cursive script recognition- A summary', *In Proc. 1st Int. Joint Conf. Pattern Recognition*, pp. 169-171, 1973.

[46] R. W. Ehrich and K. J. Koehler, 'Experiments in the contextual recognition of cursive script', *IEEE Trans. on Computer*, vol. c24, pp. 182-194, 1975.

[47] S. S. El-Dabi, R. Ramsis and A. Kamel, 'Arabic character recognition system: A statistical approach for recognizing cursive type-written text', *Pattern Recognition*, vol. 23, pp. 485-495, 1990.

[48] F. El-Khaly and M. A. Sid-Ahmed, 'Machine recognition of optically captured machine printed Arabic text', *Pattern Recognition*, vol. 23, pp. 1207-1214, 1990.

[49] T. S. EL-Sheikh and R. M. Guindi, 'Computer recognition of Arabic cursive scripts', *Pattern Recognition*, vol. 21, pp. 293-302, 1988.

[50] H. F. Feng and T. Pavlidis, 'Decomposition of polygons into simpler components: Feature generation for syntactic pattern recognition', *IEEE Trans. on Computer*, vol. 24, pp. 636-650, 1975.

[51] L. A. Fletcher and R. Kasturi, 'A robust algorithm for text string separation from mixed text/graphics images', *IEEE Trans. on Pattern Analysis and Machine Intell.*, vol. 10, pp. 910-918, 1988.

[52] H. Fujisawa, Y. Nakano and K. Kurino, 'Segmentation methods for character recognition: From segmentation to document structure analysis', *Proceedings of the IEEE*, vol. 80, pp. 1079-1092, 1992.

[53] M. Kushnir, K. Abe and K. Matsumoto, 'An application of the Hough transform to the recognition of printed Herbrew characters', *Pattern Recognition*, vol. 16, pp. 183-191, 1983.

[54] , H. F. Gaines, *Cryptanalysis*, Dover, New York, pp. 218-221, 1956.

[55] E. J. Galli and H Yamada, 'An automatic dictionary and the verification of machine-readable text', *IBM Systems Journal*, vol. 6, pp. 192-207, 1967.

[56] H. Genchi, K. Mori, S. Watanabe and S. Katsuragi, 'Recognition of hand-printed numerical characters for automatic letter sorting', *Proceedings of the IEEE*, vol. 56, pp. 1292-1301, 1968.

[57] M. Gersho and R. Reiter, 'Information retrieval using self-organizing and hetero-associative supervised neural networks. *In Proc. of IJCNN*, San Diego, Calif., 1990.

[58] V. K. Govindan and A. P. Shivaprasad, 'Character recognition - a survey', *Pattern Recognition*, vol. 23, pp. 671-683, 1990.

[59] R. M. Goodman and P. Smith, 'Decision tree design from a communication theory standpoint', *IEEE Trans. on Information Theory*, vol.

IT-34, pp. 979-994, 1988.

[60] G. H. Granlund, 'Fourier pre-processing for hand-printed character recognition', *IEEE Trans. on Computer*, vol. C21, pp. 195-201, 1972.

[61] R. L. Grimsdale, F. H. Summer, C. J. Tunis and T. Kilburn, 'A system for the automatic recognition of patterns', *Proceedings of the IEE*, vol. 106, pp. 210-221, 1959.

[62] P. M. Hahn and N. C. Randall, 'A best-match file search procedure for postal direction', *In Proc. Int Conf. Cybern. Soc.*, pp. 512-518, 1972.

[63] A. R. Hanson , E. M. Riseman and E. Fisher, 'Context in word recognition', *Pattern Recognition*, vol. 8, pp. 35-45, 1976.

[64] A. Hashizume, P. S. Yeh and A. Rosenfeld, 'A method of detecting the orientation of aligned components', *Pattern Recognition Letters*, vol. 4, pp. 125-132, 1986.

[65] J. Henseler, J. C. Scholtes and C. R. Verdoest, 'Design of a paralel knowledge-based optical character recognition system', *Master of Science Thesis*, Dept. of Mathematics and Informatics, Delft Univ. of Technology.

[66] S. C. Hinds and J. L. Fisher and D. P. D'Amato, 'A document skew detection method using run-length encoding and the Hough transform', *In Proc. 10th Int. Conf. on Pattern Recognition*, vol. 1, pp. 464-468, 1990.

[67] J. N. Holmes and N. C. Sedgwick, 'Noise compensation for speech recognition using probabilistic model', *In Proc. ICASSP-86*, Tokyo, pp. 741-744, 1986.

[68] H. S. Hou, *Digital Document Processing*, John Wiley, Canada, 1983.

[69] J. J. Hull and S. N. Srihari, 'Experiments in text recognition with binary n-gram and Viterbi algorithms', *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 4, pp. 520-530, 1982.

[70] T. S. Huang and M. Lung, 'Separating similar complex Chinese characters by Walsh transform', *Pattern Recognition*, vol. 20, pp. 425-428, 1987.

[71] T. Iijima, Y. Okumura and K. Kuwabara, 'New process of character recognition using sieving method', *Information and Control Research*, vol. 1, pp. 30-35, 1963.

[72] A. K. Jain and S. Bhattacharjee, 'Text segmentation using Gabor filters for automatic document processing', *Machine Vision and Applications*, vol. 5, pp 169-184, 1992.

[73] F. Jelinek, 'Continuous speech processing by statistical method', *Proceeding of the IEEE*, vol. 64, pp. 532-556, 1976.

[74] S. Kahan, T. Pavlidis and H. S. Baird, 'On the recognition of printed character of any font and size', *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 9, pp. 274-288, 1987.

[75] J. Kanai, 'Text line extraction using character prototypes', *In Proc. of the IAPR Workshop on SSPR*, Murray Hill, N.J. pp. 182-191, 1990.

[76] J. Kanai, T. A. Nartker, S. V. Rice and G. Nagy, 'Performance metrics for document understanding systems', *In Proc. Int. Conf. Document Analysis and Recognition*, pp. 424-427, 1993.

[77] R. Kasturi, S. T. Bow, W. El-Masri, J, Gattiker and U. B. Mokate, 'A system for interpretation of line drawings', *IEEE Pattern Anal. and Machine Intell.*, vol. 12, pp. 978-992, 1990.

[78] R. Kasturi, S. Siva and L. O'Gorman, 'Techniques for line drawings interpretation, An overview', *In Proc. Int. Workshop on Machine Machine Vision and Applications*, Tokyo, pp. 151-160, 1990.

[79] R. Kasturi and L. O'Gorman, 'Document image analysis techniques: An introduction', *Machine Vision and Applications*, vol. 5, pp. 141-142, 1992.

[80] R. Kasturi and M. M. Trivedi, *Image analysis applications, Marcel Dekker*, New York, 1990.

[81] J. N. Kapur, P. K. Sahoo and A. K. C. Wong, 'A new method for gray-level picture thresholding using the entropy of the histogram', *Computer Vision Graphics and Image Process.*, vol. 29, pp. 273-285, 1985.

[82] J. Keller and D. E. Rumelhart, 'A self- organizing integrated segmentation and recognition neural net', *Advances in neural information processing systems*, J. E. Moody, S. J. Hanson and R. P. Lippmann Eds. Morgan Kaufmann, San Mateo, Calif, vol.4, pp. 496-503, 1992.

[83] M. D. Kernighan, K. W Church and W. A. Gale, 'Spelling correction program based on a noisy channel model', *In Proc. of COLING-90, The 13th International Conference on Computational Linguistics*, vol. 2, (Helsinki, Finland), Hans Karlgren, Ed., pp. 205-210, 1990.

[84] I. Khan, S. K. Gupta, and S. H. S Rizvi, 'Statistics of printed Hindi text characters: preliminary result', *J. Inst. Electronics and Telecom. Engg.*, vol. 37, pp. 268-275, 1991.

[85] C. Kimpan, A. Atoh and K. Kawanishi, 'Fine classification of printed Thai character recognition using the Karhunen-Loeve expansion', *Proceedings of the IEEE*, vol. 134, pp. 257-264, 1987.

[86] A. Kornai, K. Mohiuddin and S. Connell, 'An HMM-based legal-amount-field OCR system for checks', *In Proc. Int. Conf. on Systems, Man and Cybernetics*, Vancouver, 1995.

[87] K. Kukich, 'Variations on a back-propagation name recognition net', *In Proc. of the Advanced Technology Conference*, vol. 2, U. S. Postal Service, Washington D.C., pp. 722-735, 1988.

[88] K. Kukich, 'Techniques for automatically correcting words in text', *ACM Computing Surveys*, vol. 24, pp. 377-439, 1992.

[89] A. Kundu, Y. He and P. Bhal, 'Recognition of hand-written word: First and second order hidden Markov model based approach', *Pattern Recognition*, vol. 22, pp. 283- 297, 1989.

[90] R. C. Kurzweil, 'Kurzweil reading machine for the blind', (users manual), *Kurzweil computers products*, Cambridge, MA, 1990.

[91] L. Lam and C. Y. Suen, 'Structural classification and relaxation matching of totally unconstrained hand-written ZIP-code numbers', *Pattern Recognition*, vol. 21, pp. 19-31, 1988.

[92] D. S. Le, G. R. Thoma and H. Wechsler, 'Automatic page orientation and skew angle detection for binary document images', *Pattern Recognition*, vol. 27, pp. 1325-1344, 1994.

[93] V. I. Levenshtein, 'Binary codes capable of correcting deletions, insertions and reversals', *Sov. Phys. Dokl.*, vol. 10, pp. 707-710, 1966.

[94] Y. K. Lin and K. S. Fu, 'Automatic classification of cervical cells using a binary tree classifier', *Pattern Recognition*, vol. 16, pp. 69-80, 1983.

[95] Y. Lu, 'Machine printed character segmentation-an overview', *Pattern Recognition*, vol. 28, pp. 67-18, 1995.

[96] S. A. Mahmoud, 'Arabic character recognition using Fourier descriptors and character contour encoding', *Pattern Recognition*, vol. 27, pp. 815-824, 1994.

[97] J. Mantas, 'An overview of character recognition methodologies', *Pattern Recognition*, vol. 19, pp. 425-430, 1986.

[98] J. Mao, K. Mohiuddin and T. Fujisaki, 'A two- stage multi-network OCR system with a soft pre-classifier and a network selector', *In Proc. Int. Conf. on Document Analysis and Recognition*, Montreal, 1995.

[99] M. Mohamed and P. Gader, 'Hand-written word recognition using segmentation free hidden Markov modeling and segmentation based dy-

namic programming technique', *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 18, pp. 548- 554, 1996.

[100] K. Mohiuddin and J. Mao, A comparative study of different classification for hand-printed recognition, *Pattern Recognition Practice IV* (E. Gelsema and L. Kanal Eds.), *Elsevier Science*, The Netherlands, 1994.

[101] S. Mori, K. Yamamoto and M. Yasuda, 'Research on machine recognition of hand-printed characters', *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 6, pp. 386-405, 1984.

[102] S. Mori, C. Y. Suen and K. Yamamoto, 'Historical review of OCR research and development', *Proceedings of the IEEE*, vol. 80, pp. 1029-1058, 1992.

[103] J. K. Mui and K. S. Fu, 'Automatic classification of nucleated blood cells using a binary tree classifier', *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 2, pp. 429-439, 1980.

[104] G. Nagy, 'Preliminary investigation of techniques for automatic reading of unformated text', *Commun. of the ACM*, vol. 11, pp. 480-487, 1968.

[105] G. Nagy, 'Optical character recognition: theory and practice', *Handbook of Statistics*, P. R. Krishnaiah and L. N. Kanal eds. vol.2, Amsterdam, North Holland, pp. 621-649, 1982.

[106] G. Nagy, 'Optical Scanning Digitizers', *IEEE Trans. on Computer*, vol. 16, pp. 13-14, 1983.

[107] G. Nagy, 'Chinese Character recognition - A twenty five years retrospective', *In Proc. 9th Int. Conf. on Pattern Recognition*, pp. 109-114, 1988.

[108] G. Nagy, 'Teaching a computer to read', *In Proc. 11th Int. Conf. on Pattern Recognition*, pp. 225-229, 1992.

[109] G. Nagy, 'At the frontiers of OCR', *Proceedings of the IEEE*, vol. 80, pp. 1093-1100, 1992.

[110] G. Nagy, 'Validation of simulated OCR data sets', *Symp. Document Analysis and Information Retrieval*, pp. 127-135, 1994.

[111] G. Nagy and S. Seth, 'Modern optical character recognition', *The Froehlich/Kent Encyclopedia of Telecommunications*, (F. E. Froehlich and A. Kent Eds.), *Marcel Dekker*, Inc, pp. 473-531, 1996.

[112] G. Nagy, S. Seth and K. Einspahr, 'Decoding substitutions ciphers by means of word matching with applications to OCR', *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 9, pp. 710-715, 1987.

[113] H. Nishida and S. Mori, 'An approach to automatic construction of structural model for character recognition', *In Proc. First Int. Conf. on Document Analysis and Recognition*, France, pp. 231-241, 1991.

[114] L. O'Gorman and G. A. Story, 'Subsampling text images', *In Proc. Int. Conf. on Document Analysis and Recognition*, France, pp. 219-227, 1991.

[115] L. O'Gorman, 'The document spectrum for page layout analysis', *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 15, pp. 1162-1173, 1993.

[116] L. O'Gorman and R. Kasturi, *Document image analysis*, *IEEE Computer Society Press*, Los Alamitos, CA, 1995.

[117] N. Ostu, 'A threshold selection method from gray level histograms', *IEEE Trans. on Systems Man and Cybernetics*, vol. 9, pp. 62-66, 1979.

[118] U. Pal and B. B. Chaudhuri, 'An improved document skew angle estimation technique', *Pattern Recognition Letters*, vol. 17, pp. 899-904, 1996.

[119] U. Pal and B. B. Chaudhuri, 'Printed Devnagari script OCR system', *Vivek*, vol.10, pp. 12-24, 1997.

[120] U. Pal and B. B. Chaudhuri, 'Automatic separation of words in Indian multi-lingual multi-script documents', *4th Int. Conf. on Docu-*

ment Analysis and Recognition, Ulm, Germany, 18-20 August, 1997 (accepted).

[121] T. Pavlidis and F. Ali, 'Computer recognition of hand-printed numerals by polygonal approximation', *IEEE Trans. on Syst. Man Cyb.*, vol. 5, pp. 610-614, 1975.

[122] T. Pavlidis and J. Zhou, 'Page segmentation and classification', *Computer Vision Graphics and Image Processing*, vol. 54, pp. 484-496, 1992.

[123] J. L. Peterson, 'Computer program for detecting and correcting spelling errors', *Commun. of the ACM*, vol. 23, pp. 676-685, Dec, 1980.

[124] J. L. Peterson, A note on undetected typing errors, *Commun. of the ACM*, vol. 29, pp. 633-637, 1986.

[125] J. J. Pollock and A. Zamora, 'Automatic spelling correction in scientific and scholarly text', *Commun. of the ACM* , vol. 27, pp. 358-368, 1984.

[126] W. Postl, 'Detection of linear oblique structures and skew in in digitized documents', *In Proc. 8th Int. Conf. on Pattern Recognition*, pp. 464-468, 1986.

[127] S. N. S. Rajasekaran and B. L. Deekshatulu, 'Recognition of printed Telegu characters', *Computer Graphics and Image Processing*, vol. 6, pp. 335-360, 1977.

[128] A. K. Ray and B. Chatterjee, 'Design of a nearest neighbor classifier system for Bengali character recognition', *J. Inst. Elec. Telecom. Engineering (India)*, vol. 30, pp. 226-229, 1984.

[129] S. V. Rice, J. Kanai and T. A. Nartker, 'A report on the accuracy of OCR devices', *Technical Report, Information Science Research Institute*, University of Nevada at Las Vegas, 1992.

[130] S. V. Rice, J. Kanai and T. A. Nartker, 'An Evaluation of OCR accuracy', *In UNLV ISRI Annual Report, Information Science Research Institute*, University of Nevada at Las Vegas, 1992.

[131] S. V. Rice, J. Kanai and T. A. Nartker, 'An evaluation of OCR accuracy', *In UNLV ISRI Annual Report, Information Science Research Institute*, University of Nevada at Las Vegas, pp. 9-35, 1993.

[132] S. V. Rice, J. Kanai and T. A. Nartker, 'The third annual test of OCR accuracy', *In UNLV ISRI Annual Report, Information Science Research Institute*, University of Nevada at Las Vegas, pp. 11-38, 1994.

[133] E. M. Riseman and A. R. Hanson, 'A contextual post-processing system for error correction using binary n-grams', *IEEE Trans. on Computer*, vol. c23, pp. 483-493, May, 1974.

[134] J. Rocha and T. Pavlidis, 'Character recognition without segmentation', *IEEE Trans. on Pattern Anal. Machine Intell.*, vol. 17, pp. 903-909, 1995.

[135] A. Rosenfeld, 'Digital straight line segments', *IEEE Trans. on Computer*, vol. C23, pp. 1264-1269, 1974.

[136] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, vol. 2, *Academic Press*, Orlando, 1982.

[137] D. E. Rumelhart, G. E. Hington and R. J. Williams, 'Learning internal representations by error propagation', *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E Rumelhart and J. L. McClelland Ed. Bradford book, *MIT press*, 1986.

[138] P. K. Sahoo, S. Soltani and A. K. C. Wong, 'A survey of thresholding techniques', *Computer Vision Graphics and Image Processing*, vol. 41, pp. 233-260, 1988.

[139] I. Sekita, K. Toraichi, R. Mori, K. Yamamoto and H. Yamada, 'Feature extraction of hand-printed Japanese characters by spline function for relaxation matching', *Pattern Recognition*, vol. 21, pp. 9-17, 1988.

[140] I. K. Sethi and B. Chatterjee, 'Machine recognition of constrained hand-printed Devnagari numerals', *J. Inst. Elec. Telecom. Engineering*, vol.

22, pp. 532-535, 1976.

[141] I. K. Sethi and B. Chatterjee, 'Machine recognition of constrained hand-printed Devnagari', *Pattern Recognition*, vol. 9, pp. 69-76, 1977.

[142] R. J. Shillman, T. T. Kuklinski and B. A. Blesser, 'Experimental methodologies for character recognition based on phenomenological attributes', *In Proc. 2nd Int. Conf. on Pattern Recognition*, Copenhagen, pp. 195-201, 1974.

[143] M. Shridhar and A. Badrelgin, 'High accuracy character recognition algorithm using Fourier and topological descriptors', *Pattern Recognition*, vol. 17, pp. 515-523, 1984.

[144] R. Singhal and G. T. Toussaint, 'Experiments in text recognition with the modified Viterbi algorithm', *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 1, pp. 184-193, 1979.

[145] R. M. K. Sinha, 'Rule based contextual post- processing for Devnagari text recognition', *Pattern Recognition*, vol. 20, pp. 475-485, 1985.

[146] R. M. K. Sinha, 'Role of context in Devnagari script recognition', *J. Inst. Elec. Telecom Engineering (India)*, vol. 33, pp. 87-91, 1987.

[147] R. M. K. Sinha, 'A Syntactic pattern analysis system and its application to Devnagari script recognition', *Ph.D Thesis*, Electrical Engineering Dept., Indian Institute of Technology, Indian, 1973

[148] R. M. K. Sinha and H. Mahabala, 'Machine recognition of Devanagari script', *IEEE Trans. on Systems, Man and Cybernetics*, vol. 9, pp.435-441, 1979.

[149] R. M. K. Sinha, B. Prasada, F. Houlr and M. Sabourin, 'Hybrid contextual text recognition with string matching', *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 15, pp. 915-924, 1993.

[150] G. Siromony, R. Chandrasekaran and M. Chandrasekaran, 'Computer recognition of printed Tamil characters', *Pattern Recognition*, vol. 10,

pp. 243-247, 1978.

[151] S. N. Srihari, 'Computer text recognition and error correction', *IEEE Computer Society Press*, Los Angeles, 1985.

[152] S. N. Srihari, 'High-performance reading machines', *Proceedings of the IEEE*, vol. 80, pp. 1120- 1132, 1992.

[153] S. N. Srihari and J. J. Hull, 'Character Recognition', *Encyclopedia of Artificial Intelligence*, (In S. C. Shariro Eds.) *Wiely*, New York, pp. 138-150, 1992.

[154] S. N. Srihari and V. Govindaraju, 'Analysis of textual images using the Hough transform', *Machine Vision and Applications*, vol. 2, pp. 141-153, 1989.

[155] W. Stallings, 'Recognition of printed characters by automatic pattern analysis', *Computer Graphics and Image Process.*, vol. 1, pp. 47-65, 1966.

[156] W. Stallings, 'Approaches to Chinese character recognition', *Pattern Recognition*, vol. 8, pp. 87-98, 1976.

[157] C. Y. Suen, 'N-gram statistics for natural language understanding and text processing', *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 1, pp. 164-172, 1979.

[158] C. Y. Suen, M. Berthod and S. Mori, 'Automatic recognition of hand-printed character - the state of art', *Proceedings of the IEEE*, vol. 68, pp. 469-487, 1980.

[159] A. Sur, *Two hundred years of Bangla printing*, Jignasha, Calcutta, 1977.

[160] H. Takahashi, N. Itoh, T. Amano and A. Yamashita, 'A spelling correction method and its application to an OCR system', *Pattern Recognition*, vol. 23, pp. 363-377, 1990.

[161] E. Tanaka, T. Kohashiguchi and K. Shimamura, 'High speed string correction for OCR', *In Proc. 6th Int. Conf. on Pattern Recognition*,

pp. 340-343, 1986.

[162] G. Tauschek, 'Reading machine', *U. S. Patent, 2026329*, 1935.

[163] H. Tenant, *Natural Language Processing*, Princeton, N. J. Petrocelli, 1980.

[164] J. N. Tripathi, 'Statistical studies of printed Devnagari (Hindi) text', *J. Inst. Electronics and Telecom. Engg.*, vol. 17, pp. 25-27, 1971.

[165] J. D. Tubbs, 'A notes on binary template matching', *Pattern Recognition*, vol. 22, pp. 359-365, 1989.

[166] S. Tsujimoto and H. Asada, 'Major components of a complete text reading system', *Proceedings of the IEEE*, vol. 80, pp. 1133-1149, 1992.

[167] J. R. Ullmann, 'Experiments with the n- tuple method of pattern recognition', *IEEE Trans. on Computer*, vol. 18, pp. 1135-1137, 1969.

[168] J. R. Ullmann, 'A binary n-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words', *Computer Journal*, vol. 20, pp. 141-147, 1977.

[169] Mrs. Uma and S. Kalyanaraman, 'Script processing and computer based lexicons of south Asian languages: A perspective on Kalyan Saraswati multi-media operating system', *In Proc. of Conf. on Information Technology, Application in Language Script and Speech*, pp. 7-38, New-Delhi, 1994.

[170] R. A. Wagner, 'Order-n correction for regular languages', *Commun. of the ACM*, vol. 17, pp. 265-268, 1974.

[171] J. Wang and J. Jean, 'Resolving multi-font character confusion with neural networks', *Pattern Recognition*, vol. 26, pp. 175-187, 1993.

[172] J. Wang and J. Jean, 'Segmentation of merged characters by neural networks and shortest path', *Pattern Recognition*, vol. 27, pp. 649-658, 1994.

[173] K. Y. Wang, R. G. Casey and F. M. Wahl, 'Document analysis system', *IBM J. Res. Development*, vol. 26, pp. 647-656, 1982.

[174] P. S. P Wang, *Character and hand-written recognition*, World Scientific, Singapore, 1991.

[175] Q. R. Wang and C. Y. Suen, 'Analysis and design of a decision tree based on entropy reduction and its application to large character set recognition', *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 6, pp. 406-417, 1984.

[176] Q. R. Wang, C. Y. Suen, 'Large tree classification with heuristic search and global training', *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 9, pp. 91-102, 1987.

[177] S. Wendling, C. Gagneux and G. Stamon, 'Use of Haar transform and some of its properties in character recognition', *In Proc. 3rd Int. Conf. on Pattern Recognition*, CA, U.S.A., pp. 844-848, 1976.

[178] S. Wendling and G. Stamon, 'Hadamard and Haar transforms and their power spectrum in character recognition', *In Proc. Workshop on Pattern Recognition and Artificial Intelligence*, Hyannis, Mass, U.S.A., pp. 103-112, 1976.

[179] J. S. Weszka, 'A survey of threshold selection techniques', *Computer Graphics and Image Processing*, vol. 7, pp. 259-265, 1978.

[180] X. L. Xie and M. Suk, 'On machine recognition of hand-printed Chinese characters, by feature matching', *Pattern Recognition*, vol. 21, pp. 1-7, 1988.

[181] H. Yan, 'Skew correction of document images using interline cross-correlation', *CVGIP: Graphical models and Image Processing*, vol. 55, pp. 538-543, 1993.

[182] E. J. Yannakoudakis and D. Angelidakise, 'An insight into the entropy and redundancy of the English dictionary', *IEEE Trans. on Pattern*

Anal. and Machine Intell., vol. 10, pp. 960-970, 1988.

[183] E. J. Yannakoudakis and D. Fawthrop, 'An intelligent spelling corrector', *Inf. Process Manage.*, vol. 19, pp. 101-108, 1983.

[184] M. Yoshida and M. Eden, 'Hand-written Chinese character recognition by an analysis by synthesis method', *In Proc. 1st Int. Conf. on Pattern Recognition*, Washington, pp. 197-204, 1973.

[185] G. K. Zipf, *Human behaviors and the principle of least effort : An introduction to human ecology*, Addison - Wesley press, INC, Cambridge 42, Massachusetts, 1949.

Publications of the Author related to this thesis

1 B. B. Chaudhuri and U. Pal, 'Character occurrence statistics in Bangla language', *Technical Report*, TR 94/2, Indian Statistical Institute, 1994.

2 U. Pal and B. B. Chaudhuri, 'Computer recognition of printed Bangla script', *Int. Journal of Systems Science*, vol. 26, pp. 2107-2123, 1995.

3 B. B. Chaudhuri and U. Pal, 'Relational studies between phoneme and grapheme statistics in current Bangla', *Journal of the Acoustical Society of India*, vol. 23, pp. 67- 77, 1995.

4 U. Pal and B. B. Chaudhuri, 'An improved document skew angle estimation technique', *Pattern Recognition Letters*, vol. 17, pp. 899-904, 1996.

5 B. B. Chaudhuri and U. Pal, 'OCR error detection and correction of an inflectional Indian language script', *Proc. 13th Int. Conf. On Pattern Recognition*, vol. 3, pp. 245- 249. Vienna, 1996.

6 U. Pal and B. B. Chaudhuri, 'Printed Devnagari script OCR system', *Vivek*, vol. 10, pp. 12-24, 1997.

7 B. B. Chaudhuri and U. Pal, 'Skew angle detection of digitized Indian script documents', *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 182-186, 1997.

8 B. B. Chaudhuri and U. Pal, 'An OCR system to read two Indian language scripts: Bangla and Devnagari', *4th Int. Conf. on Document Analysis and Recognition*, Ulm, Germany (Accepted).

9 B. B. Chaudhuri and U. Pal, 'A complete printed Bangla OCR system', *Pattern Recognition* (In Press).