A NOVEL APPROACH TOWARDS DEVANAGARI TRANSLITERATON USING STATISTICAL AND STRUCTURAL FEATURE EXTRACTION

A Dissertation submitted in partial fulfillment of the requirements for the award of the Degree of

MASTER OF ENGINEERING
IN
WIRELESS COMMUNICATION

Submitted by

JASMINE KAUR
Roll No. 801463011

Under the Guidance of

Dr. VINAY KUMAR
Assistant Professor, ECED
Thapar University, Patiala

Department of Electronics and Communication Engineering
THAPAR UNIVERSITY, PATIALA
(Established under the section 3 of UGC Act, 1956)
PATIALA – 147004 (PUNJAB)
July 2016

DECLARATION

I, **Jasmine Kaur**, hereby declare that the dissertation entitled "**A novel approach towards Devanagari transliteration using statistical and structural feature extraction**" is an authentic record of my own work carried out towards the partial fulfillment of requirements for the award of degree of Master of Engineering in Wireless Communication from Thapar University, Patiala, under the supervision of **Dr. Vinay Kumar**, Assistant Professor, Electronics and Communication Engineering Department.

The matter presented in this dissertation has not been submitted in any other University/Institute for the award of any other degree.

Jasmine

Date: 13-7-2016

Jasmine Kaur

Roll No. 801463011

This is to certify that the above statement made by the student is correct to the best of my knowledge and belief.

Date: 13-7-2016

Dr. Vinay Kumar

Assistant Professor

ECED, TU, Patiala

Countersigned by:

Dr. Sanjay Sharma

Professor and Head ECED

Thapar University, Patiala

Dr. S.S. Bhatia

Dean of Academic Affairs

Thapar University, Patiala

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to **Dr. Vinay Kumar**, Assistant Professor, Department of Electronics and Communication Engineering, Thapar University, Patiala for his patient guidance, invaluable advice and immense support throughout this project. He helped me in clarifying abstruse concepts and understanding objective of my work. I am truly very fortunate to have the opportunity to work with him and without his cooperation this project would not have seen light of day.

I am also thankful to our **Head of the Department, Dr. Sanjay Sharma** for providing me adequate environment in carrying the work. I would like to extend my gratitude to entire faculty and staff of Electronics and Communication Engineering Department who directly or indirectly helped me in the process and contributed towards this work.

I am sincerely thankful to **Dr. Amit Kumar Kohli,** Associate Professor and P.G. Coordinator, and **Dr. Hem Dutt Joshi,** Assistant Professor and Program Coordinator, Electronics and Communication Engineering Department, for their support and providing the facilities for successful completion of this dissertation.

Last, but not the least, I am greatly indebted to my family and friends for their years of unyielding love and encouragement throughout this dissertation.

Place: TU, Patiala **Jasmine Kaur**

Date: 13-7-2016 Roll No. 801463011

ABSTRACT

Majority of the ancient Indian literature such as Bhagavad Gita, Vedas, Mahabharata, and Ramayana is written in Devanagari script. Devanagari script is popular in India and is known by just a small fraction of population whereas Roman script is widely adopted all over the world. To make the rich voluminous Indian literature readily available to the people who are unfamiliar with Devanagari script, transliteration of the Devanagari documents into a much familiar Roman script is the way to go.

This dissertation attempts in Romanization of Devanagari document using character recognition with the help of underlying statistical and structural properties of the characters. The character recognition process interprets the document images and converts the text into editable format. Moreover automation of this process will greatly reduce the human interference while converting the Devanagari text documents to much familiar and editable roman script. However it is a challenging task because of the complex structure and enormity of Devanagari character set as compared to limited size of roman alphabets.

One of the first tasks performed to isolate the constituent characters is segmentation. Line segmentation methodology in this dissertation discusses the case of overlapping and skewed lines. Overlapping line segmentation is based on number of connected components which is made equivalent to number of individual lines in the image. Mathematical morphological operation, closing and dilation to be exact are used to limit skew angle variation range thereby expediting the projection profile method of skew correction. The presented skew correction method works for full range of angles. The proposed character segmentation algorithm is designed to segment conjuncts and separate shadow characters. Presented shadow character segmentation scheme employs connected component method to isolate the character, keeping the constituent characters intact. Statistical features namely different order moments like area, variance, skewness and kurtosis along with structural features of characters are employed in two phase recognition process. After recognition, constituent Devanagari characters are mapped to corresponding roman alphabets in a way that resulting roman alphabets have similar pronunciation as the source characters. The algorithm is evaluated comprehensively on various Devanagari documents with positive results.

CONTENTS

DECLARATION I

ACKNOWLEDGEMENT II

ABSTRACT III

CONTENTS IV

LIST OF FIGURES VI

LIST OF TABLES VIII

LIST OF ABBREVIATIONS IX

1. Chapter 1 Introduction 1-7

1.1. Motivation 2

1.2. Features of Devanagari script 3

1.3. General model of Devanagari character recognition system 4

1.3.1. Binarization 5

1.3.2. Segmentation 5

1.3.3. Feature extraction 6

1.3.4. Character recognition 6

1.4. Organisation of Dissertation 7

2. Chapter 2 Literature Review 8-17

2.1. Binarization 9

2.2. Pre-Processing 9

2.3. Line segmentation 11

2.4. Word segmentation 12

2.5. Character segmentation 13

2.6. Feature extraction 14

3. Chapter 3 Proposed Methodology 18-55

3.1. Binarization 20

3.2. Line segmentation 21

3.2.1. Segmentation of unskewed lines 21

3.2.2. Segmentation of overlapping lines 22

3.2.3. Segmentation of skewed lines 26

3.3. Word segmentation 35

3.4. Character segmentation 35
3.4.1. Segmentation of lower modifiers/descenders 37
3.4.2. Segmentation of conjuncts/touching characters and shadow characters 40
3.5. Feature extraction and classification of modifiers 46
3.6. Feature extraction for character recognition 48
3.7. Recognition 52
3.7.1. Inadequate classification due to similarity in character pairs after removing header line 52
3.7.2. Inadequate character classification due to rakar modifier 54
3.8. Transliteration 55
4. **Chapter 4 Results and performance analysis** **56-72**
4.1. Experimental results 57
4.2. Performance analysis 70
5. **Chapter 5 Conclusion and future scope** **73-75**

REFERENCES 76-79
LIST OF PUBLICATIONS 80
ORIGINALITY REPORT 81

LIST OF FIGURES

Figure 1.1 Character zones in Devanagari script 03
Figure 1.2 Stages of Devanagari recognition process 04
Figure 1.3 Example showing various components in a word..... 06
Figure 3.1 Stages of Devanagari recognition process and errors in different stages .. 19
Figure 3.2 Example of Binarization 20
Figure 3.3 Line Segmentation 22
Figure 3.4 Example of overlapping lines 22
Figure 3.5 Example of types of pixel connectivity 23
Figure 3.6 Flowchart representing method to isolate overlapping lines 24
Figure 3.7 Overlapping line segmentation 25
Figure 3.8 Example of dilation with line SE of varying angles 27
Figure 3.9 Example of closing with SEs of different sizes 27
Figure 3.10 Flowchart depicting skew correction methodology 29
Figure 3.11 Skew correction of anticlockwise aligned image..... 33
Figure 3.12 Flowchart to correctly align flipped image 34
Figure 3.13 Segmented line with its vertical projection showing the columns used for word segmentation 35
Figure 3.14 Segmentation of a word 36
Figure 3.15 Example of Devanagari characters with varying height 37
Figure 3.16 Segmentation of character with isolated lower modifier 38
Figure 3.17 Segmentation of characters with joined lower modifier 39
Figure 3.18 Example of character segmentation with obstructing lower modifier 40
Figure 3.19 Conjuncts and shadow characters 40
Figure 3.20 Analysing conjuncts on the basis of their width 42
Figure 3.21 Flowchart for segmenting conjuncts 43
Figure 3.22 Conjunct segmentation..... 44

Figure 3.23 Characters in shadow and their segmentation 44
Figure 3.24 Flowchart representing method to isolate shadow characters 45
Figure 3.25 Shadow characters 45
Figure 3.26 Shadow character segmentation 46
Figure 3.27 Example of 8-connected neighbours with origin at *M(r,c)* 47
Figure 3.28 Modifier image showing extreme points 47
Figure 3.29 Top strip modifier components 48
Figure 3.30 Example illustrating neighbouring features of two different modifiers 48
Figure 3.31 Grid showing 13 different non-uniform zones 49
Figure 3.32 Zoning and corresponding mask formation 49
Figure 3.33 Example of similar character pair after header line removal 53
Figure 3.34 Flowchart distinguishing between similar looking characters after header line removal 53
Figure 3.35 Few examples of consonant-rakar combination 54
Figure 3.36 Flowchart for accurate classification of a consonant-rakar combination 54
Figure 3.37 Phonetically similar Roman alphabet for each Devanagari character (IAST scheme) 55
Figure 4.1 Results for Image 1 59
Figure 4.2 Results for Image 2 61
Figure 4.3 Results for Image 3 62
Figure 4.4 Results for Image 4 64
Figure 4.5 Results for Image 5 65
Figure 4.6 Results for Image 6 66
Figure 4.7 Results for Image 7 67
Figure 4.8 Results for Image 8 68
Figure 4.9 Results for Image 9 69
Figure 4.10 Special cases of conjunct combinations 71

LIST OF TABLES

Table 1.1 Types of characters and their examples……...…6

Table 3.1 Formulas for calculating different transition and projection moments…..51

Table 4.1 Performance of composite character segmentation……71

Table 4.2 Performance of Devanagari transliteration process……....72

LIST OF ABBREVIATIONS

IAST	International Alphabet of Sanskrit Transliteration
ITRANS	Indian languages Transliteration
RGB	Red-Green-Blue
LH	Horizontal Lows/vertical Highs
PSL	Piece wise Separating Line
OCR	Optical Character Recognition
CR	Character Recognition
k-NN	k-Nearest Neighbour
CC	Connected Component
SE	Structuring Element
BB	Bounding Box

CHAPTER -1

INTRODUCTION

1.1 Motivation

Devanagari script has its roots dated back thousands of years ago. Most of the Indian literature such as Bhagavad Gita, Vedas, Mahabharata, and Ramayana is written in Devanagari. Such voluminous literature necessitates transliteration into roman script to make it more accessible to people who are unfamiliar with Devanagari.

The human reading process is one of the most complex operation evidently showcasing human intelligence over any artificial recognition system. There is general consensus among researchers in this field that reading incorporates two major subprocess: recognition and comprehension. Now it will be hard for an individual to comprehend any words if the corresponding language script is unfamiliar to him. This is where transliteration comes into play. Transliteration involves transformation of one script to another based on phonetic similarities between the characters of two distinctive scripts. In general, mapping between Devanagari and Roman character in a transliteration scheme should be as close as possible to the pronunciation of the source character in the target script. Romanization of Devanagari characters is useful, sometimes necessary and there are number of ways to do it. But still there are few points that have to be taken into consideration:

i) **Reversibility:** The aim of transliteration is to be unique and hence reversible. There should be one to one mapping between source and target script, making it possible to restore original script from converted text.

ii) **Simplicity:** Since basic roman script has a smaller character set as compared to phonetically distinct Devanagari characters, digraphs and diacritics are used to represent Devanagari characters in Roman script.

iii) **Universal:** Universal usage can only be ensured by using an international standard of transliteration. Several methods are available for transliteration of Devanagari script such as International Alphabet of Sanskrit Transliteration (IAST) which is a widely used standard, Hunterian system which is officially adopted by Government of India, Indian languages Transliteration (ITRANS) etc.

This thesis work attempts in Romanization of Devanagari document using character recognition with the help of underlying statistical and structural properties of characters. The character recognition process interprets the document images and converts the text into editable format. Recognition systems for languages using roman script are quite extensively developed but same is not the case with Indic scripts like Devanagari. The

motivation behind this research is developing a recognition system that improves Devanagari recognition process by discussing several problems faced at each phase and transliterating the recognized characters.

1.2 Features of Devanagari script

Devanagari is one of the most popular scripts in India. One of the prime aspects of Devanagari script is diverse set of sounds it can support. Because there is typically a character for each of the phonemes in Devanagari, the character set tends to be quite large as compared to roman script. Along with 11 vowels and 33 consonants which form basic characters, Devanagari script comprises of half consonants, modifiers and conjuncts. Vowels can be written as individual characters or as diacritics above, below, before or after the consonants to modify their sound. This representation of vowel is called ascender when vowel is placed above the core strip and descender when it is placed below. Two or more consonants when combined together form a composite character called conjunct. Characters in Devanagari script are connected via horizontal header line. No roman character has such characteristics and therefore segmentation of Devanagari characters becomes quite a complicated task. Devanagari text can be divided into three zones: A *core* zone that contains basic characters and conjuncts, an *upper* zone which contains ascenders or upper modifiers and a *lower* zone which contains descenders or lower modifiers.

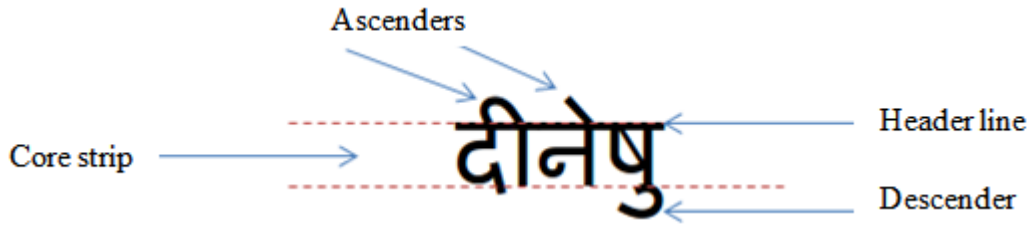


Figure 1.1: Character zones in Devanagari script [1]

1.3 General model of Devanagari character recognition system

For Devanagari transliteration process we have used character recognition model with transliteration process as concluding stage. The basic character recognition model can be divided into following stages:

i) Binarization
ii) Segmentation
iii) Feature extraction
iv) Character recognition

A brief explanation of all the stages is given in next subsections. Figure 1.2 shows a basic character recognition model. This dissertation discusses each of these stages while developing an automatic transliteration system.

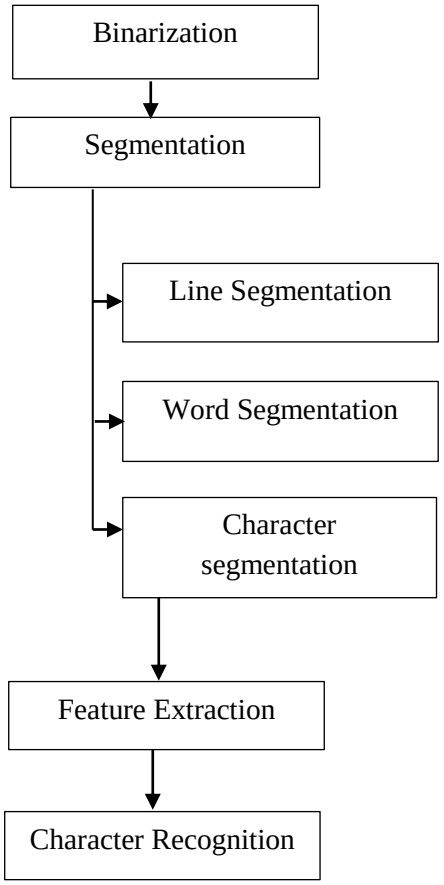


Figure 1.2: Stages of Devanagari recognition process

1.3.1 Binarization

Document binarization is the first crucial step in processing of Devanagari documents.The objective of binarization is to convert the given input grayscale or RGB document image into a bi-level representation zero and one. The character pixels with value '0' represent colour black and pixels with value '1' are represented as white. Binarizing the document decreases the computations. Therefore, most of the document processing systems have been developed to work on binary images. Binarization techniques [2] can be briefly classified into two categories:

i) Global thresholding scheme where a single threshold value is used to binarize whole document.

ii) Local thresholding where threshold value varies over the document image and is calculated on the basis of pixel intensities in a small neighbourhood. In this technique various sized windows are used and a single threshold value is computed for the pixels lying in this window.

1.3.2 Segmentation

In Devanagari optical character recognition, segmentation is one of the most critical steps. Segmentation refers to extracting lines from a document and thereafter extracting words and isolating individual characters from these segmented words for further processing. Figure 1.3 shows individual character components after segmentation. Due to complex structure of Devanagari alphabets and presence of conjuncts and modifiers as shown in Table 1, the segmentation procedure is much more complex than segmentation in roman script. The segmentation strategies [3] can be classified into three categories as follows:

i) The classical approach that segments the character subimages based on character-like features.

ii) Recognition-based approach, in which a search is made for image elements that are similar to each segment of the word.

iii) A holistic approach that attempts to recognize the complete word.

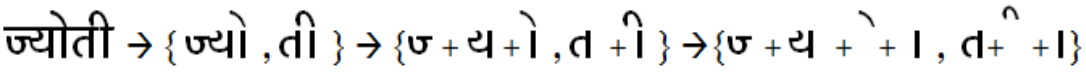


Figure 1.3: Example showing various components in a word.

Table 1.1: Types of characters and their examples

Character type	Examples
Consonant/ Vowel	ह र अ
Consonant-ascenders	हे बि री
Consonant-descender	हु सू
Conjunct	थ्ह क्क
Conjunct-ascender	क्हे क्री
Conjunct-descender	क्हु थ्रू

1.3.3 Feature extraction

Feature extraction is a process of computing a set of parameters that characterize the shape of the segmented character as precisely and uniquely as possible. The features are selected in such a way that they help in discerning one character from another. Selection of feature extraction methods is probably the single most important factor in achieving high performance in recognition. Feature extraction method [4] can be divided into two categories:

i) Structural method, which are based on geometrical and topological properties of characters such as number of endpoints, cross points, strokes and their direction etc.

ii) Statistical methods, which are derived from statistical distribution of pixels such as moments, zoning, projection profiles etc.

For this dissertation we have chosen combination of structural and statistical features.

1.3.4 Character recognition

Recognition is used to identify the underlying character and assign a label to character images based on extracted features and relationship between them. Accuracy of recognition highly depends on results of all the former steps.

1.4 Organisation of Dissertation

This dissertation comprises of 5 chapters which are organized as below:
Chapter 1: Introduction, this chapter explains motivation behind this research. Features of Devanagari script and general model Devanagari character recognition system and its brief explanation have also been discussed in this chapter.

Chapter 2: Literature survey, in this chapter we review the literature of Devanagari character recognition that includes algorithms for line segmentation, character segmentation, feature extraction etc.

Chapter 3: Proposed methodology, in this chapter we have described proposed method of automatic transliteration of Devanagari document. After binarizing the input document, line segmentation method along with special case of overlapping and skewed lines has been explained. In next stage, character segmentation of composite characters such as conjuncts and shadow characters has been discussed. Feature extraction using statistical and structural features of characters and recognition of modifiers and characters is described next. Final stage includes transliteration of recognized Devanagari characters.

Chapter 4: Results and performance analysis, in this chapter we display the transliteration results of several Devanagari documents. Performance analysis of the proposed technique is also discussed in this chapter.

Chapter 5: Conclusion and future scope, this chapter concludes the entire dissertation and mentions the areas where improvement can be made.

CHAPTER -2

LITERATURE REVIEW

This chapter presents a survey of previously done work in the field of Devanagari character recognition. Various methods used for each corresponding stage of Devanagari character recognition are overviewed in this section. On the basis of literature survey we have discussed various gaps in different phases and objective of dissertation later in this dissertation.

2.1 Binarization

Till date several improvised binarization methods have been proposed based on **Niblack's method** [5] that binarizes the image by altering mean and standard deviation in small neighbourhoods of a pixel. Based on this method **J. Sauvola** *et al*. [6] attempted to threshold a gray-level image by fluctuating the mean threshold value with a factor that contains a constant value and dynamic standard range. **C. Wolf** *et al*. [7] further improved the thresholding technique [5] and [6] by using contrast values rather than gray values. He normalized contrast and mean gray values of the image to compute the threshold.

Feng *et al*. [2] proposed a binarization technique employing two windows, one within another. For the inner contained window three features mean, minimum gray intensity and standard deviation are calculated whereas for larger window dynamic standard deviation range is calculated. This method utilizes the notion that windows with text present larger standard deviation contrary to the others.

Y. Solihin *et al*. [8] proposed a histogram based global thresholding technique wherein each pixel in an image is classified into one of the three classes: foreground, background, and a fuzzy area between them where it is hard to determine whether a pixel belongs to the foreground or the background.

2.2 Pre-Processing

T. Y. Zhang *et al*. [9] proposed a thinning algorithm that removes all the contour points that are not part of skeleton, all the while preserving end points and pixel connectivity. The method utilizes 8-pixel connectivity to obtain character image with unitary thickness.

A. Papandreou *et al*. [10] proposed a hybrid technique that uses both vertical projection and bounding box technique to determine the skew in a document with Latin characters. For a non-skewed document great peaks are observed in the certain columns with

minimum bounding box area. The ratio of summation of black pixels in a column to the area of bounding box is maximised, by rotating the image, to determine the skew angle.

Shutao Li *et al*. [11] presented a method for skew detection using wavelet decomposition and projections. This scheme works better if we know the layout of the document beforehand. For this method, LH subband is selected as it preserves horizontal structure of image. For the matrix of LH subband coefficients horizontal projection profile is calculated by rotating it through a range. The angle for which criterion function is maximum is regarded as final skew angle. Criterion function is selected in such a way that it not only reduces computational time but also improves accuracy.

B.V. Dhandra *et al*. [12] presented a method which uses morphological technique dilation to fill the space between the characters using a fixed appropriate structuring element. The dilated text lines are then region labelled to find orientation angle for different region. Average value of orientation angles of different regions is considered as final skew of the document. For obtaining orientation angle, an ellipse is fitted over these regions. The angle of major axis of ellipse with respect to horizontal direction is orientation angle. This method was experimented for skew angles in range of 0° to 15°.

M. Hanmandlu *et al.* [13] proposed a pre-processing technique to remove slant in character image, obtain skeletonized image and later smoothing this pre-processed image to remove any unwanted pixels. Slant removal is done by dividing character image in two halves and using centre of gravity of two halves to remove the slant. Recognition of character is carried out using vector distances as a feature and modified exponential membership function.

A.K. Das *et al.* [14] presented a skew correction method using morphological operations. The document image is closed to form black bands corresponding to the text lines. The black bands have bumps due to presence of modifiers that are removed by opening the image using square structural element. The uniformly smeared text line image is then vertically scanned to find 1 to 0 transitions that are mapped as a new image. This gives outline of bottom portion that has single pixel width. Lines with length less than the threshold value are removed. For each of the remaining lines skew angle is calculated and resultant skew is median of skew values computed for selected lines.

2.3 Line segmentation

U. Pal *et al*. [15] presented a line segmentation technique using projection profile method. Projection profile is first used to determine the orientation of text blocks by using hill valley distance as a parameter. In a horizontal projection profile, columns with white run length less than the threshold value are turned to black to obtain a smoothed version of profile. From this smoothed profile, top-most hill and bottom-most valley points are determined to calculate hill valley distance. This distance is then used to decide the mode of text block. After determining the orientation of text block, authors used valley points to segment the lines. A valley with least height denotes a boundary line and a text line is found between two consecutive boundary lines.

U. Pal *et al*. [16] presented another method for both line and word segmentation. This algorithm uses piece wise projection wherein document page is divided into vertical stripes. For each of these stripes a row called piece wise separating line (PSL) is found with sum of black pixels equal to zero. If multiple consecutive rows have their black pixel sum equal to zero then foremost row is selected. From these piece wise separating lines, few potential PSLs are selected and joined to get individual text lines. In some cases no potential PSL is found and are taken care of during joining. After line segmentation, vertical histogram of line is computed and peak and valley points are used to segment the words.

N.K. Garg *et al.* [17] presented a line segmentation technique that uses several assumptions based on height of text lines and is based on header line and base line detection. First rough estimate of header line is made by detecting a row with maximum pixels. Next two consecutive rough header lines are taken and divided into four stripes and for each of these stripes a row with minimum pixels is consider as base line. A text line lies between a header line and corresponding base line. This algorithm makes too many assumptions to calculate base and header line, as a result it is not so effective.

G. Louloudis *et al.* [18] presented text line detection technique for handwritten documents that uses connected component along with hough transform to extract individual lines. The document image is first scanned to find connected component that are divided into three separate domains based on height. For the first subdomain with least height, corresponding to the height of a single text line without accents, hough

transform is utilized to select potential lines. Last step separates connected characters in remaining sub-domains.

A. Zahour *et al.* [19] presented text line extraction technique that employs contour following to extract the text lines in a document image. For each text line, partial contour is followed in forward and then in opposite direction to separate the lines.

S. Pal *et al.* [20] presented a line and word separation technique for printed documents based on projection profile and printed documents. This method uses a four step technique to find the candidate line that separates the two consecutive middle lines of horizontal projection. First, run length smearing of text lines is performed and the white runs with length less than five times the stroke width are changed to black. Second, recursive procedure is applied to find middle lines for the segmentation. Third, candidate line is found by vertically scanning the histogram and is used to separate the text lines. At last the problem of overlapping and touching components is resolved. To separate the overlapping lines, contour points of overlapping components are traced.

The text lines in a document can be oriented in different directions or can be curved. A water-reservoir-based technique is proposed by **U. Pal** *et al.* [21] to extract such lines. In this method, initially connected components are labelled and are discerned as either isolated or touching. Later, each touching component is identified as straight or curved based on reservoir base area and envelope points. Two candidate points of each touching component are computed along with their candidate regions. On the basis of candidate regions, components are separated as different text lines.

2.4 Word segmentation

R. Manmatha *et al.* [22] presented a method for automatically segmenting the words from document using scale space approach. In this technique, line images are filtered at different scales to form blobs which correspond to characters at small scales and to words at larger scale. The goal is to find an optimum scale such that formed blobs correspond to words rather than to characters. To separate the words from each other the blobs are confined to their corresponding bounding box.

2.5 Character segmentation

S. Kompalli *et al*. [23] explained several challenges faced in Devanagari OCR like fused ascender segmentation, descender segmentation due to varying character height and corresponding modifier placement and problems faced during core component separation.

S. Kompalli *et al*. [24] presented a recognition based segmentation technique for Devanagari optical character recognition. In this proposed method character image is split into horizontal runs and each of these runs is then further identified as merging, splitting and continuing runs considering number of horizontal runs above and below the current run. Adjacent continuing runs are then combined together to form a single block. Centroids of these blocks are then calculated and graph is constructed by joining centroid of neighbouring blocks. Segmentation is performed by selecting the subgraphs from block adjacency graph representation of character or word images.

R.M.K Sinha *et al*. [1] proposed a segmentation method for touching and fused Devanagari characters. This method uses a two pass algorithm to segment the composite characters. First, the statistical information about width and height of a character is used to determine whether the character box consists of composite characters or not. In second pass these composite characters are further segmented using profiles. Instead of horizontal projection, collapsed horizontal projection is employed for further segmentation. Collapsed horizontal projection is obtained by removing vertical bar in a character image along with image on the right of the vertical bar. To separate the composite characters, segmenting column is found by computing horizontal projection and checking its continuity.

U. Garain *et al*. [25] developed an algorithm for effectively selecting a column for segmentation of touching characters. Before performing segmentation several key observation have been made by researchers based on statistical analysis of composite characters. This method used fuzzy multifactorial analysis which employs calculation of five factors for each character column based on the observed features of touching Devanagari characters and then isolates the characters which require further segmentation. For these selected characters a 5-D vector is calculated that reflects five different aspects of each column in the character image. The 5-D vector is minimised to 1-D matrix to find appropriate segmentation cut. Each fuzzy factor used reflects some statistical behaviour of touching character.

S. Kompalli *et al*. [7] outlined two different techniques for recognition of printed multi-font Devanagari text. In first phase, words are segmented along linear boundaries using vertical and horizontal profiles and structural features of character. Subsequently, classification is performed assuming that the Devanagari components have been accurately segmented. The second approach uses classifiers for preliminary hypothesis to obtain frequently occurring characters for each segment of the word before attempting conjuncts or consonant-descenders segmentation. These results are utilized in segmentation of certain composite characters and descenders to obtain isolated core characters and corresponding modifiers. Former technique is segmentation driven while the latter is recognition driven segmentation. Unlike first approach, recognition driven segmentation uses non-linear boundaries to segment Devanagari components.

2.6 Feature extraction

R.M.K. Sinha *et al*. [27] developed an OCR system for Devanagari script that utilized structural features of the characters for classification. For this purpose they devised a system using four robust filters based on structural features. These features include division of characters based on coverage region of core strip and vertical bar, horizontal crossing and vertex points. Number of horizontal crossings is found by scanning each row of the character image and dividing this sequence into three sub-sequences. The number of end points and their position in terms of 9(3x3) zones are also utilized as feature for character recognition.

V. Bansal *et al*. [28] designed a front end system to integrate several statistical knowledge sources at various stages. The researchers explained the importance of all of the knowledge sources that are based on structural properties of characters. Some of these knowledge sources include character confidence matrix, character confusion matrix, height and width statistical information of characters, word envelope information, vertex point and pixel density information. Some of these features are used as filters whereas others are used to rank the candidate character. Few of the knowledge sources are acquired a priori while others are obtained from the text as it is processed.

U. Pal *et al*. [29] presented a feature extraction technique based on directional information obtained from direction of gradient. The image is mean filtered and segmented into 49x49 blocks to find directions of gradient. The obtained direction of gradients is quantized into 32 directions with $\pi/16$ intervals. Histogram of these 32

directional gradients is computed and smoothing filter is applied to reduce the resulting directions to a value 8. To further reduce the size of directional feature vector, the image grid of 49x49 blocks is down sampled to 7x7 block configuration giving a 392 dimension feature vector.

The feature extraction technique proposed by **U. Pal** *et al*. fails for similarly shaped characters as directional features for such characters are approximately the same. **T. Wakabayashi** *et al.* [30] devised a novel technique to reduce such errors by weighting the direction feature vector by a factor called F-ratio. F-ratio is computed by ratio of between-class variance to within-class variance. It alters the directional feature vector of two similarly shaped characters by weighting the feature elements to accentuate distinguishable portion of the similarly shaped characters and diminish common portion of the characters, so that identically shaped characters can be identified easily.

A novel scheme of feature extraction based on chain code histogram of character contour, shadow features and view based feature was proposed by **S. Arora** *et al*. [31]. Shadow represents length of a projection on a side. For computing shadow features, character bounding box is divided into eight octants. For character portion in each of these octants shadow is computed on its sides. Using shadow features researchers obtained 24 features. To determine chain code features, contour points of character image are determined and next a different code is assigned to each point in the contour in such a way that it indicates direction of next pixel of contour. The contour image is divided in 5x5 blocks. For each block, frequency of direction code is found along with histogram of corresponding chain code. The view of a character is one of its four projections and consists of pixels belonging to the contour with extreme value for one of the coordinate.

Govindaraju *et al*. [32] considered gradient features for feature extraction scheme. The gradient feature is computed using sobel operator which measures the magnitude and direction of intensity changes in a small neighbourhood of each pixel. For gradient magnitudes above a particular threshold value gradient map is determined which when concatenated form feature vector.

For their recognition driven segmented characters, **S. Kompalli** *et al*. [7] employed gradient, structural and concavity features for CR of Devanagari text. These features are scale invariant in nature and represent local edge, curvature and strokes information of a character.

N. Sharma *et al*. [33] devised a recognition scheme based on features extracted from contour pixels of the characters by computing corresponding directional chain codes. In this case, the contour points of the character image are found and then the bounding box of character image is divided into 7 x 7 blocks. In each block, the direction chain code for all contour pixels is computed in four different directions and later frequency of chain codes is computed. Histogram of the values of these direction codes in each block is then constructed.

B. V. Dhandra *et al*. [34] a recognition system based on spatial features. Directional spatial features like stroke density, eccentricity, extent stroke length and the number of strokes are employed as potential features to characterize the characters. k-NN classifier is further employed to classify the characters based on these features.

R. Jayadevan *et al*. [35] presented detailed survey regarding feature extraction and classification techniques used for recognition. The most important results and beneficial direction of various methods used till date has been highlighted.

O.D. Tier *et al*. [4] presented an overview of different feature extraction method for character recognition. They discussed several feature extraction method like template matching, zoning, Zernike moments, geometric moments, contour profiles, projection histogram etc. that can be used for different representations of individual characters. The feature extraction methods have been discussed in terms of expected distortion and variability of characters.

M. Hanmandlu *et al.* [36] proposed a feature extraction method utilizing skeletonized character images. To find the feature set character image is first pre-processed using a thinning algorithm to obtain a skeleton that preserves shape information of original character. The obtained thinned character image is then divided 24 boxes and vector distance for each foreground pixel from lower left corner is calculated. The feature set consists of distance vector of each box obtained by normalizing sum of all the distances in a box by number of total foreground pixels.

N. Arica *et al.* [37] overviewed entire character recognition process explaining several utilized pre-processing, segmentation and recognition methods for characters of different representations. Present status of various CR techniques along with their gaps and future scope has been discussed quite extensively.

R.J. Ramteke *et al.* [38] discussed efficacy of feature extraction method based on moment invariants and presented an improved moments invariant technique that calculates moments of different zones. Second feature vector of proposed technique utilized different features calculated by dividing image into two halves horizontally and vertically separately. Third feature set employed left and right diagonal zones.

CHAPTER -3

PROPOSED METHODOLOGY

The present dissertation focuses on automatic transliteration of Devanagari script. This chapter explain various stages of proposed Devanagari transliteration scheme. During document processing several possible challenges and errors are perceived. The possible errors in different phase have been indicated in Figure 3.1. The present chapter discusses each of these challenges and possible ways to overcome them while developing automatic transliteration system.

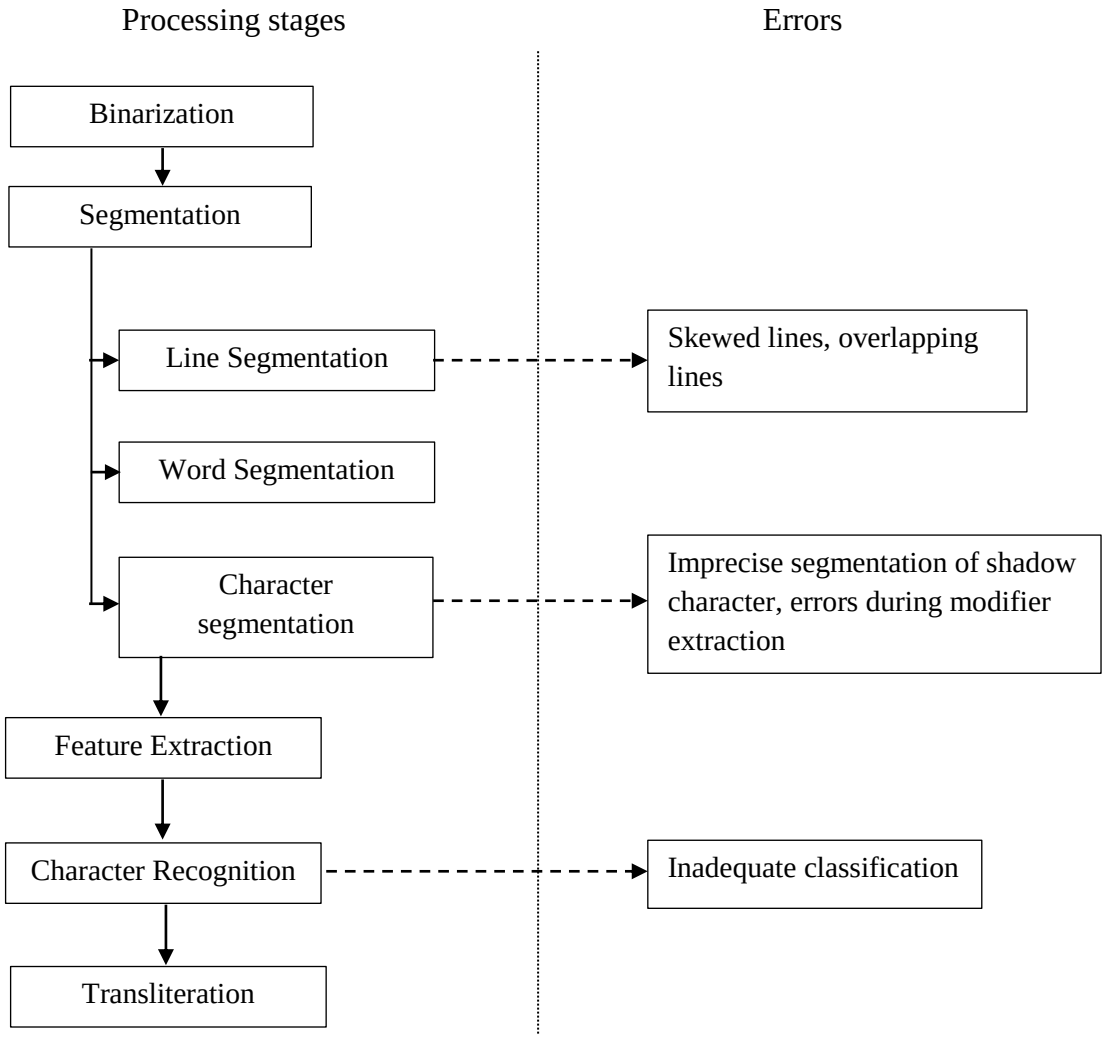


Figure 3.1: Stages of Devanagari recognition process and errors in different stages

3.1 Binarization

A image of Devanagari script I_G is converted into an inverted binary image I'_{HxW} by using basic global thresholding. Binary image is inverted so that relevant character information is represented by value '1' whereas redundant background information is represented by '0'. The histogram of grayscale image, with intensity values $I_G(m,n)$ ϵ [0,1…255], for intensity value ***i*** is found using equation (1).

$$H(i) = \sum \delta(I_G(m,n) - i) \qquad (1)$$

Since image consists of text over plain black background, we do not consider background with patterns, resulting histogram will be concentrated largely at the extreme ends of grayscale image and difference Δ (given by equation (2)) is quite large.

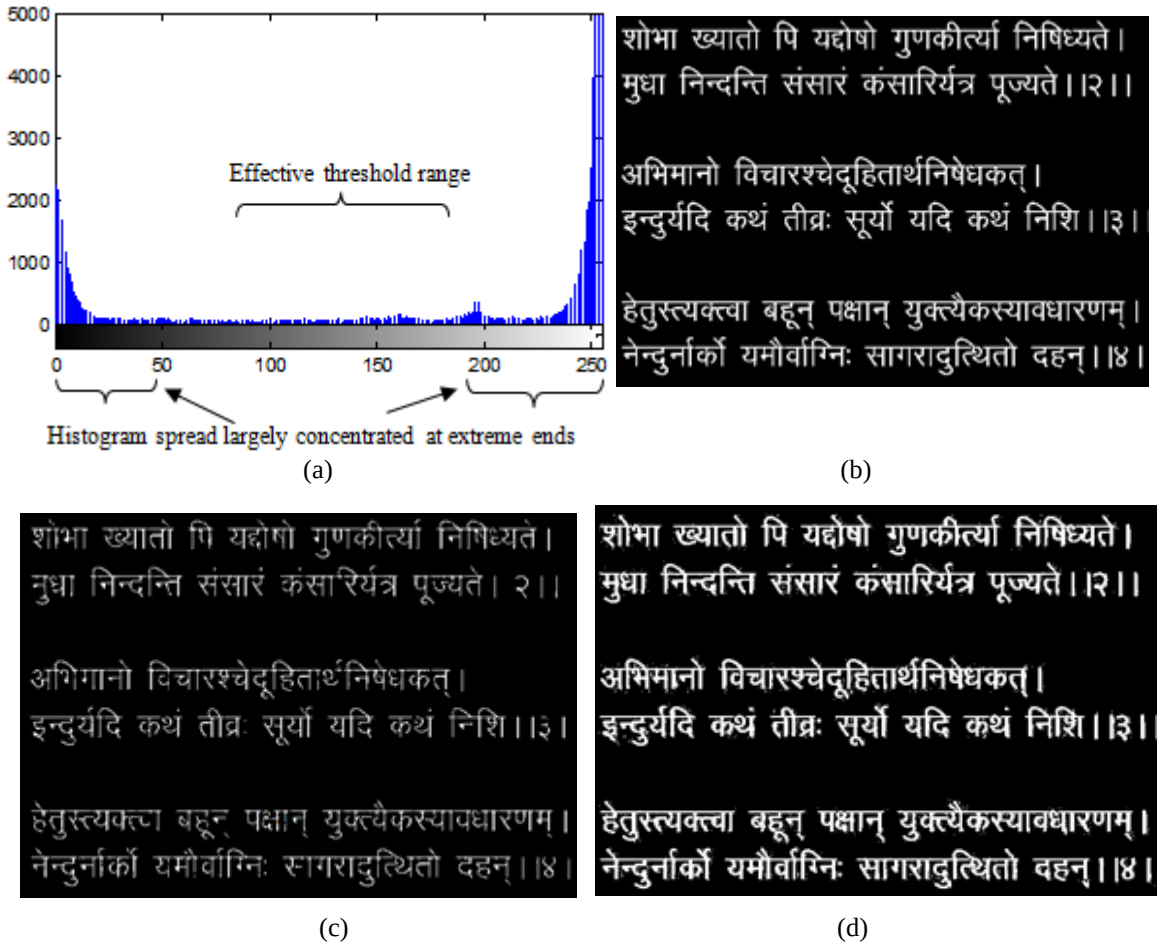


Figure 3.2: Example of Binarization (a) Histogram of input image (b) output binary image with threshold *T*=170 (c) output binary image with threshold *T*=30 (d) output binary image with threshold *T*=220

Based on experimentation we consider values greater than 150, so the threshold value separating foreground and background can have many possible values.

$$\Delta = \max_{i\epsilon[0,127]} H(i) - \max_{i\epsilon[128,255]} H(i) \quad (2)$$

Through experimentation it is found that suitable value of threshold *T* lies in the vicinity of $\Delta/2$ and different threshold values in this effective threshold range have little to no effect on the resulting image I'_{HxW} obtained using equation (3)

$$I'_{HxW} = \sim I_{HxW}(m,n) = \begin{cases} 1 & if\ I_G(m,n) \leq T \\ 0 & if\ I_G(m,n) > T \end{cases} \quad (3)$$

Image I'_{HxW} , white text on black background, is used in all of the following subsequent steps. Figure 3.2 shows result of binarization on I_G. When using a threshold value lying in a threshold region, precise binarized image is obtained with no broken or smeared characters, Figure 3.2(b). However, choosing a threshold value lying beyond threshold range gives an image where background is not distinctively separated from foreground, Figure 3.2(c), whereas a threshold value lying to the left of threshold range gives fragmented characters, Figure 3.2(d).

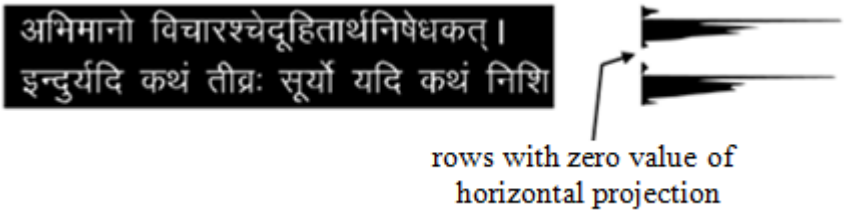
3.2 Line segmentation

Line segmentation process extracts individual lines from the image document. The following sections explain different cases of line segmentation.

3.2.1 Segmentation of unskewed lines

Decomposition of image I'_{HxW} into individual lines is carried out by computing horizontal projection *HP(m)* using equation (4). If the document has skew we first de-skew (refer section 3.2.3) it to remove the tilt. Horizontal projection is a histogram calculating number of white pixels in each row of an image. Rows corresponding to zero *HP(m)* value are used to isolate the adjacent lines shown in Figure 3.3.

$$HP(m) = \sum_{n=1}^{W} I'(m,n) \ \forall\, m \in [1,H] \quad (4)$$



(a) Input image I'_{HxW} with its horizontal projection

(b) Constituent segmented lines

Figure 3.3: Line segmentation (a) Input image I'_{HxW} with its horizontal projection

3.2.2 Segmentation of overlapping lines

In Devanagari when descenders of upper line overlap with ascenders of lower line, two lines partially overlap. It is not possible to separate them merely by drawing a horizontal line due to the absence of segmenting row in horizontal projection as shown in Figure 3.4(a).

a) Overlapping lines with corresponding horizontal projection

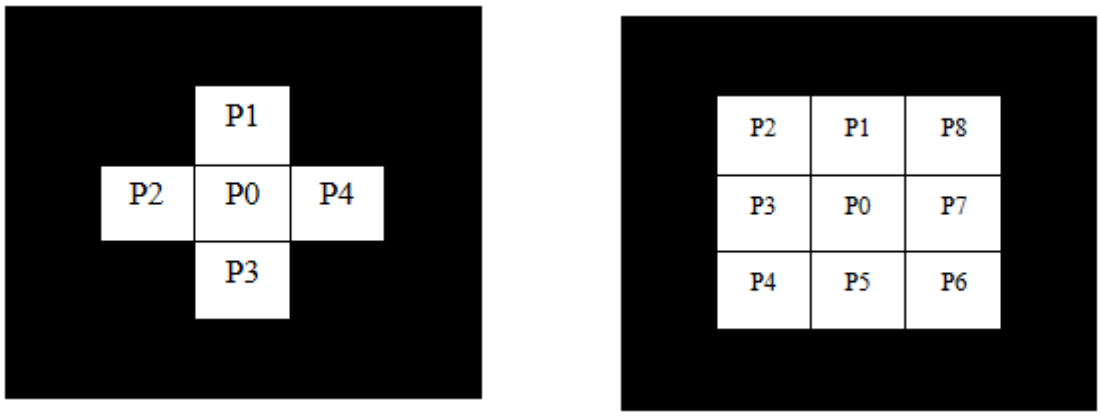
b) segmented lines by drawing a horizontal parallel cut

Figure 3.4: Example of overlapping lines

Instead of segmenting lines by drawing a parallel cut (which leaves behind fragmented modifiers of another line as shown in Figure 3.4(b)) or by making a contour [20], we use the concept of connected component separate these lines

Connected component labelling

To isolate the overlapping lines, we need to determine connected components (CCs) in such a way that number of CCs is *analogous to number of lines* in a document image. CC labelling sweeps the entire document image and groups the foreground (white) pixels into components based on pixel connectivity, *i.e.* all pixels in a CC share same pixel intensity value and are linked with each other. When computing numbers of connected elements in an image generally two types of pixel connectivity [41] are supported, 4-pixel connectivity as shown in Figure 3.5(a) and 8-pixel connectivity shown in Figure 3.5(b).



(a) Example of 4-pixel connectivity where pixels P0, P1, P2, P3 and P4 form a 4-pixel connected component

(b) One connected component based on 8-pixel connectivity

Figure 3.5: Example of types of pixel connectivity

Overlapping lines segmentation methodology

Using 8-pixel connectivity overlapping lines in the image I'_{ncc} can be isolated using following steps:

Step 1: Find the number of 8-connected elements N_{cc} [41] and corresponding pixel set I^i_{cc} associated with each component n^i_{cc}, where i=1, 2…….N_{cc}.

Step 2: 8-pixel connected elements with area less than 10 are removed by setting their constituent pixels equal to zero. Resulting image I_{ncc}, shown in Figure 3.7(b), contains overlapping lines with detached modifiers removed and can be found using equation (5).

$$I_{ncc} = (I_{cc}^{i} = 0 \Leftrightarrow card(I_{cc}^{i}) < 10) \quad (5)$$

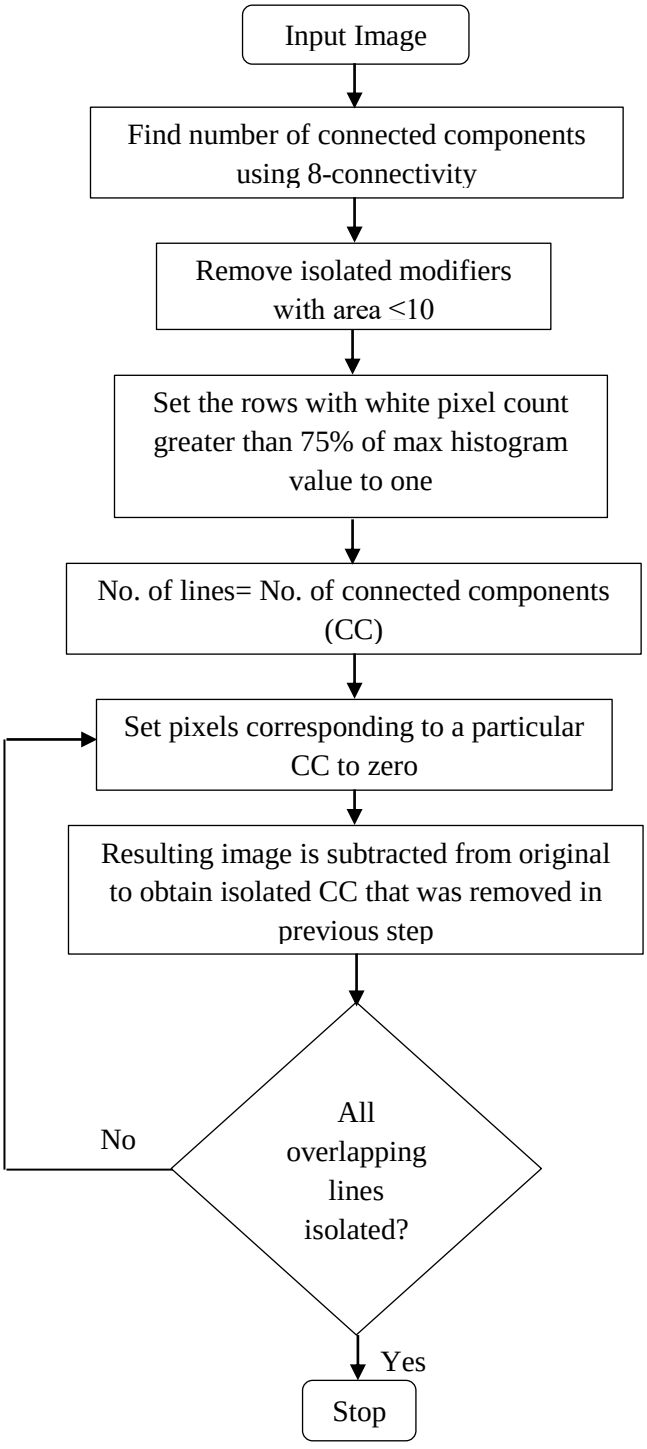


Figure 3.6: Flowchart representing method to isolate overlapping lines

Step 3: For image I_{ncc}, horizontal projection histogram is found using equation (6) and all the rows, equation (7), with pixel count greater than 0.75*max value are set to value 1 using equation (8). It results in grouping of all the words, in a particular line, into one wholesome 8-connected component. After this operation, number of CCs corresponds to number of individual lines in an image as shown in Figure 3.7(c).

$$HP(m) = \sum_{n=1}^{W} I_{ncc}(m,n) \ \forall \ m \ \in [1, H] \tag{6}$$

$$Rows = (m | HP(m) \geq 0.75 * HP_{max} \ \forall \ m) \tag{7}$$

where HP_{max} denotes the maximum value of horizontal projection $HP(m)$.

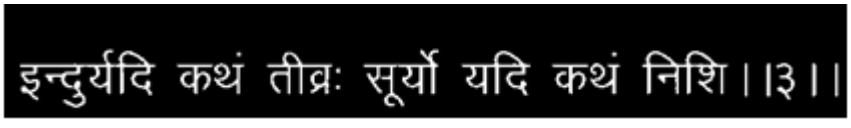
$$I_{ncc}(m,n) = 1 \ \forall \ n, m \in Rows \tag{8}$$

(a) Overlapping lines to be segmented

(b) Isolated modifiers removed

(c) Corresponding words of a line grouped into one CC

(d) Image after removing pixels corresponding a particular CC



(e) segmented line

Figure 3.7: Overlapping line segmentation

Step 4: In the image I_{ncc}, obtained from Step 3, pixels corresponding to a specific connected component, which needs to be isolated, are set to zero. The resulting image is obtained using equation (9) and is shown in Figure 3.7(d).

$$I_{ncc}(I_{cc}^{i}) = 0;\ where\ i \in [1, N_{cc}] \tag{9}$$

Step 5: Subtracting image I_{ncc}, obtained from Step 4, from original image I'_{ncc} results in an image containing isolated line (n_{cc}^{i}) which was removed in Step 4 as shown in Figure 3.7 (e).

$$n_{cc}^{i} = I'_{ncc} - I_{ncc} \tag{10}$$

Step 6: Step 4 and 5 are repeated until all the overlapping lines are segmented.

3.2.3 Segmentation of skewed lines

For segmentation of skewed lines, alignment of text lines with respect to horizontal axis is first corrected and then horizontal projection profile method mentioned in section 3.2.1 is used to isolate the lines.

Traditional projection profile methods are extremely slow and tend to fail for larger skew angles as the input document has to be rotated iteratively through fairly large range of angles. Projection profile methods are well suited to estimate skew angles within $\pm\ 10^0$. In the proposed improved approach of skew correction, time to compute skew angle is substantially reduced by using morphologically operations [42] to limit the effective range of resulting skew angle variation of a document to $\leq 15^0$. So the document needs to be rotated by an angle in this particular range, thereby increasing the computational speed. Along with increasing the speed and accuracy, operation limit for our proposed method is improved to full range of skew angle detection, i.e. -180^0 to 180^0.

The effect of morphological operations used in proposed algorithm is explained below:

Dilation: Linear SE (structuring element) symmetric with respect to centre and half the width of input image is chosen in the proposed scheme. The initial angle of the line is 10^0 and direction of orientation of SE depends on whether the image is skewed in clockwise or anti clockwise direction. The alignment of the line SE is increased iteratively by a constant value during subsequent steps of proposed algorithm. Observing Figure 3.8 we conclude that changing the orientation of line SE with respect to the text lines changes the

number of bounding box in an image. Through experimentation it is seen that when angle between a text line and line SE is less than 10 degrees then the number of bounding box is approximately equal to number of lines in an image. As we keep on increasing this angle difference, number of BB decrease in number until finally becoming equal to 1 above a certain constant angle difference.

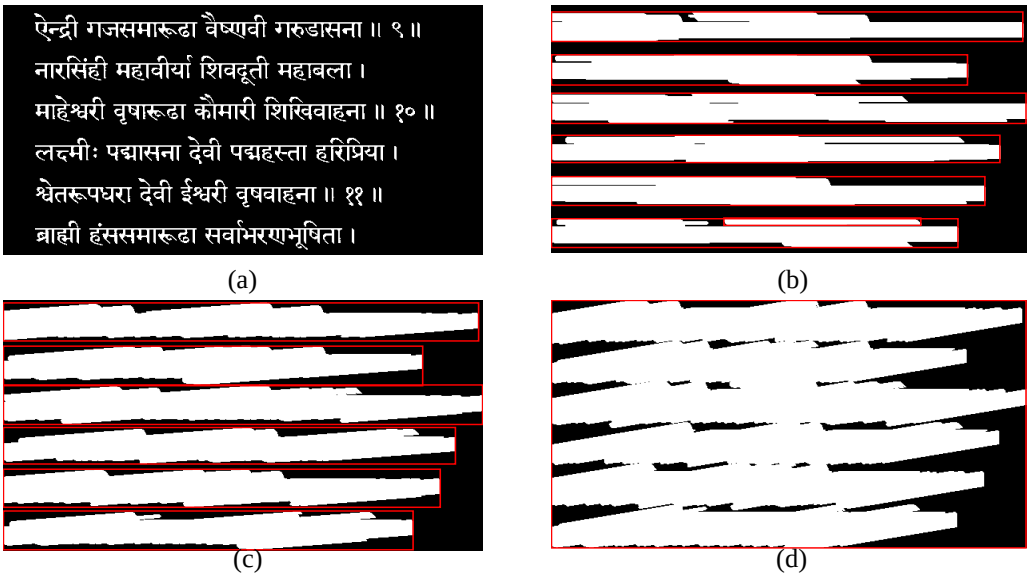


Figure 3.8: Example of dilation with line SE of varying angles (a) original image (b) dilation with SE at angle of 0^0 (c) dilation with SE at angle of 5^0 (d) dilation with SE at angle of 10^0

Closing: Performing closing operation on an image fills the background regions between the characters causing them to smear into solid white blobs. Structuring element chosen for proposed algorithm is a square element with size 15x15. As seen in Figure 3.9(b) SE of too small size can be mostly traced around the foreground region without any part of the element entering inside the foreground region. So, the size of SE is selected in such a way that it is large enough to smear the words into perfect solid blobs.

Figure 3.9: Example of closing with SEs of different sizes (a) original image (b) closing with SE of size 5x5 (c) closing with SE of size 10x10 (d) closing with SE of size 15x15

Skew correction methodology

The proposed algorithm for de-skewing image I_{skew} using morphological operations, depicted in flowchart of Figure 3.10, is given in steps below:

Step 1: Close the image I_{skew}, using equation (11), with square structuring element S_{sq} of size 15x15 to form solid text blobs corresponding to each text word. This particular action causes the peaks of horizontal projection profile to become more prominent.

$$I_{close} = I_{skew} \bullet S_{sq} = \left(I_{skew} \oplus S_{sq}\right) \ominus S_{sq} \tag{11}$$

where $\oplus$ and $\ominus$ denote dilation and erosion and can be found using equation (12) and (13).

$$I_{dilate} = I_{skew} \oplus S_{sq} = \left\{ i + s \middle| i \in I_{skew} \wedge s \in S_{sq} \right\} \tag{12}$$

$$I_{erode} = I_{skew} \ominus S_{sq} = \left\{ i \middle| \forall\, s \in S_{sq}, i + s \in I_{skew} \right\} \tag{13}$$

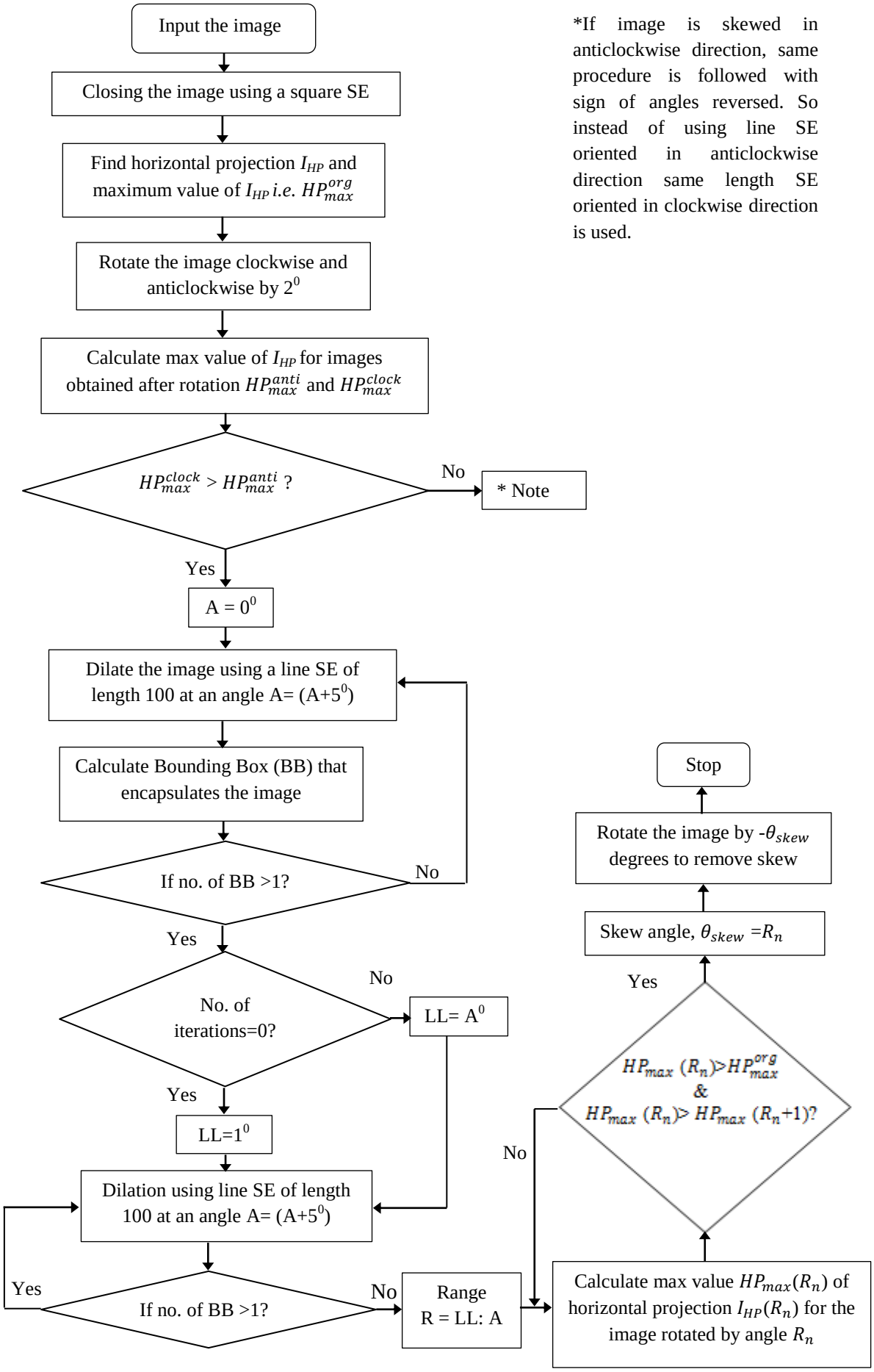
Compute horizontal projection profile of closed image and its peak value HP_{max}^{org}, using equation (14).

$$HP_{max}^{org} = \max\left(\sum_{n=1}^{W} I_{close}(m,n) \;\forall\, m \in [1,H] \right) \tag{14}$$

Closed image and its corresponding projection profile are shown in Figure 3.11(c) and Figure 3.11(d), respectively. Comparing horizontal projection of original image in Figure 3.11(b) with closed image projection in Figure 3.11(d) we notice that peaks are more prominent in case of closed image projection.

Step 2: Rotate the image I_{close} in clockwise and anticlockwise direction by 2^0. Respective rotated images I_{close}^{C} and I_{close}^{A} are shown in Figure 3.11(e) and Figure 3.11(g).

Step 3: Find horizontal projection profile of rotated images and their corresponding maxima HP_{max}^{anti} and HP_{max}^{clock} using equation (15) and (16). If peak of anticlockwise rotated image is greater than clockwise rotated image, i.e., $HP_{max}^{anti} > HP_{max}^{clock}$, image is skewed in clockwise direction and needs to be rotated in anticlockwise direction by a certain angle to remove the tilt and vice versa. As peak value HP_{max}^{anti} of projection in Figure 3.11(h) is less than HP_{max}^{clock} value of Figure 3.11(f), the image is skewed in counter clockwise direction.



*If image is skewed in anticlockwise direction, same procedure is followed with sign of angles reversed. So instead of using line SE oriented in anticlockwise direction same length SE oriented in clockwise direction is used.

Figure 3.10: Flowchart depicting skew correction methodology

$$HP_{max}^{clock} = \max\left(\sum_{n=1}^{W} I_{close}^{C}(m,n) \ \forall\, m \in [1,H]\right) \quad (15)$$

$$HP_{max}^{anti} = \max\left(\sum_{n=1}^{W} I_{close}^{A}(m,n) \ \forall\, m \in [1,H]\right) \quad (16)$$

$$\theta_{skew} < 0 \Leftrightarrow HP_{max}^{anti} > HP_{max}^{clock} \quad (17)$$

Step 4: For anticlockwise skewed image dilation, using equation (18), is performed using a line structuring element S_{line} of length 100 pixels angled at 10^0 as shown in Figure 3.11(i). For a clockwise skewed image dilation is performed using a line SE of length 100 pixels angled at -10^0.

$$I_{dilate} = I_{skew} \oplus S_{line} = \{i + s | i \in I_{skew} \Lambda\, s \in S_{line}\} \quad (18)$$

Number of enclosing BB is determined for resulting dilated image. For an image shown in Figure 3.11(i) BB count is equal to 1.

Step 5: If number of BB in *Step 6.* is greater than one, it implies that line SE and text lines of an image are oriented approximately at same angle and lower limit of skew angle variation range is assigned a value of 1^0. Upper limit in this case is found by further dilating the skewed image using equation (19). The line SE element we use is of same length as in previous step and its orientation angle is iteratively increased by 5^0 until encompassing BB number equals 1. SE orientation angle for which BB enclosing the dilated image is equivalent to 1 forms upper limit of skew angle variation range.

Step 6: When BB count in *Step 6.* is equivalent to one, orientation angle difference between line SE and text lines is greater than 10^0. To find lower limit angle skewed image is dilated, using equation (19), with SE of same length as in *Step 6*, and its angle is incremented by 5^0 for each dilation until number of BB is greater than one. Alignment angle of SE for which dilated image is enclosed by a single BB is taken as lower limit and image is continued to be dilated iteratively until enclosing BB number is equivalent to 1 again. This final SE orientation angle is considered as upper limit of skew angle variation range. Limiting the skew angle variation to a much smaller range expedites the whole process of finding document tilt. Observing images in Figure 3.11(j) through Figure 3.11(l), range R is found to be $[40^0{:}55^0]$.

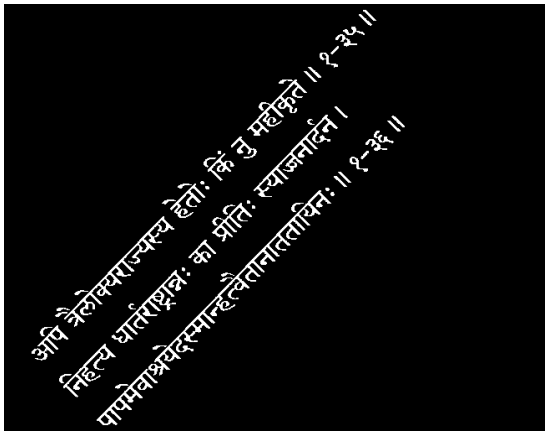
Step 7: Once the final range of skew angles (R) is computed, closed image I_{close}, obtained in step 2, is rotated iteratively through this range and horizontal projection profile peak $HP_{max}(R_n)$ is calculated for angle R_n until $HP_{max}(R_n)$ is both greater than HP_{max}^{org} found in Step 2, and peak value $HP_{max}(R_{n+1})$ found for next angle in the range. R_n represents n^{th} angle value in range R. Angle R_n satisfying equation (20) is resulting anticlockwise tilt of document image with respect to horizontal direction and this angle can be corrected by rotating the image by an angle of $-\theta_{skew}$. Final unskewed image, I_{us} is given in Figure 3.11(m).

$$HP_{max}(R_n) \ = \max\left(\sum_{n=1}^{W} I_{close}(m,n) \ \forall \ m \ \in [1,H]\right) \tag{19}$$

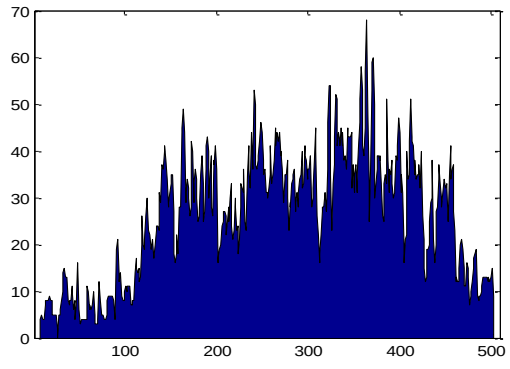
$$\theta_{skew} = R_n \Leftrightarrow \left(HP_{max}(R_n) > HP_{max}^{org} \ \Lambda \ HP_{max}(R_n) > \ HP_{max}(R_n + 1)\right) \tag{20}$$

Step 10: For image skewed in clockwise direction Steps 5 through 9 are followed in order, the only difference is that line SE orientation angles by which image is dilated have opposite sign i.e., image is dilated using SE oriented in clockwise direction.

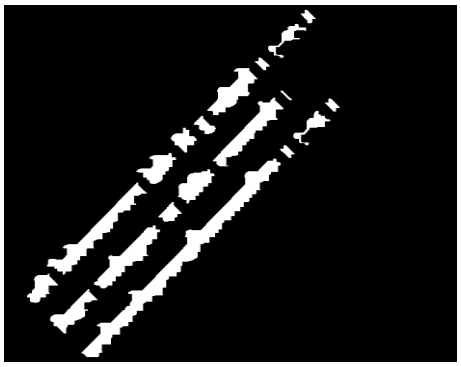
Step 11: Once the image skew is corrected, next part involves determining whether image I_{us} has been flipped upside down or not.



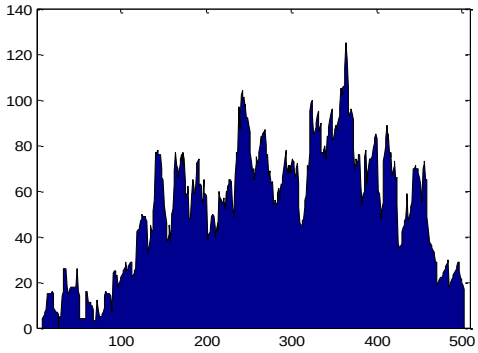
(a) Original image



(b) Horizontal projection of original image

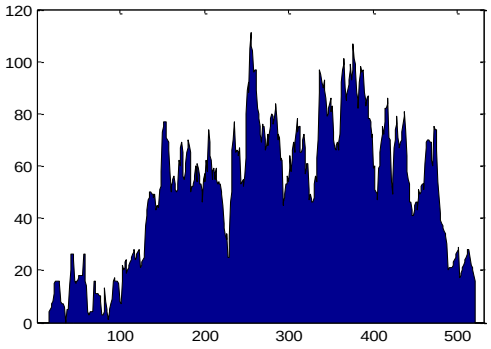


(c) Closed image using square SE of size 15x15

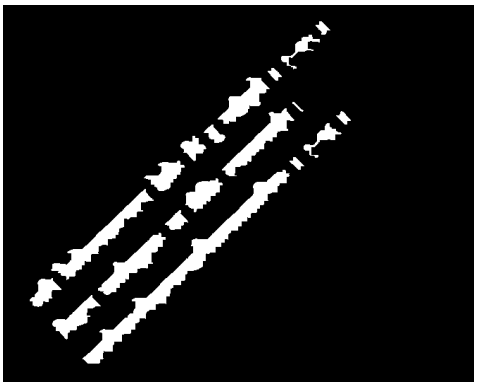


(d) Horizontal projection of skewed image

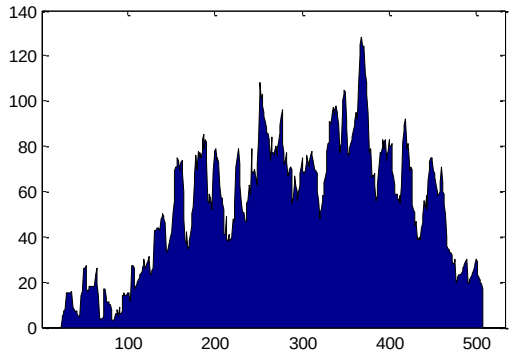
(e) Image rotated clockwise by 2^0



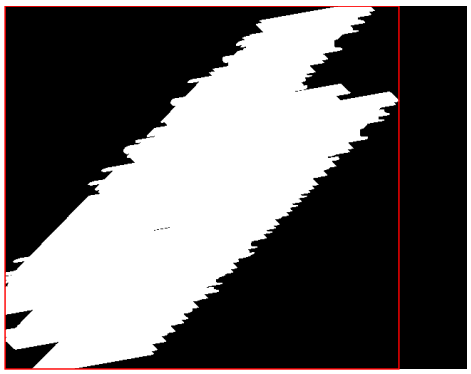
(f) Horizontal projection of clockwise rotated image with peak value 128



(g) Image rotated anticlockwise by 2^0

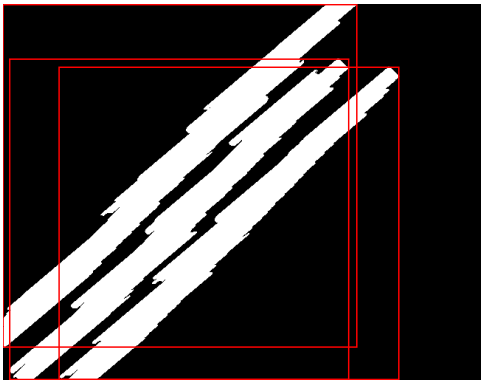


(h) Horizontal projection of anticlockwise rotated image with peak value 111

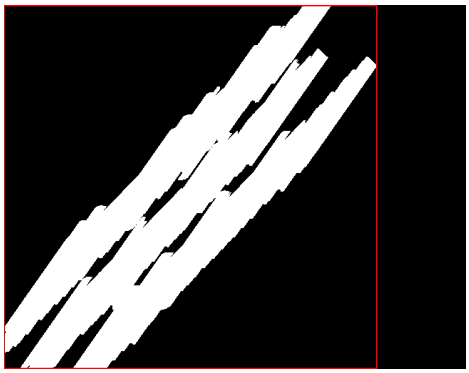


(i) Image dilated using line SE of size 100 at angle 10^0

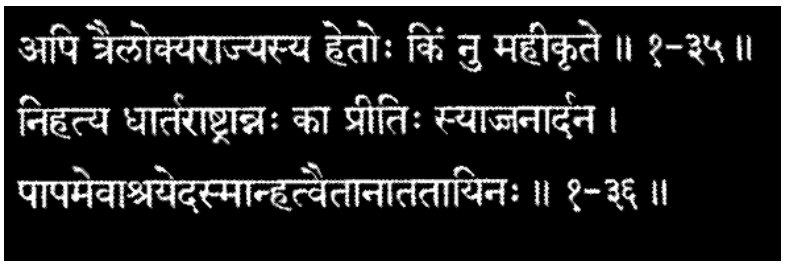
(j) Image dilated using line SE of size 100 at angle 20^0



k) Image dilated using line SE of size 100 at angle 40^0



(l) Image dilated using line SE of size 100 at angle 55^0



(m) Image corrected by rotating original image by -46^0

Figure 3.11: Skew correction of anticlockwise aligned image

Steps to figure out whether an image I_{us} is flipped upside down or not:

Step 1: Segment a single text line say L_{us}, from unskewed document image using method mentioned in Section 3.2.1 and crop it to its minimum bounding box dimensions.

Step 2: Find horizontal projection histogram say $HP_L(m)$, for image L_{us} using equation (4).

Step 3: Find the point HP_{max}^{index} where maxima occurs.

$$HP_{max}^{index} = \max_{m \in [1,H]} HP_L(m) \tag{21}$$

Step 4: If this maxima lies in first half of projection histogram, image is correctly oriented. In other words, if HP_{max}^{index} is less than the midpoint index (say HP_{mid}^{index}) of horizontal projection, image is correctly aligned and no change is made. Otherwise the obtained image is flipped upside down and is rotated by θ_{skew} =180 degrees to obtain final image.

$$HP_{mid}^{index} = \left\lfloor \frac{H}{2} \right\rfloor \tag{22}$$

Where *H* gives length of horizontal projection and $\lfloor \bullet \rfloor$ denotes greatest integer function.

$$\theta_{skew} = 180° \Leftrightarrow HP_{max}^{index} > HP_{mid}^{index} \tag{23}$$

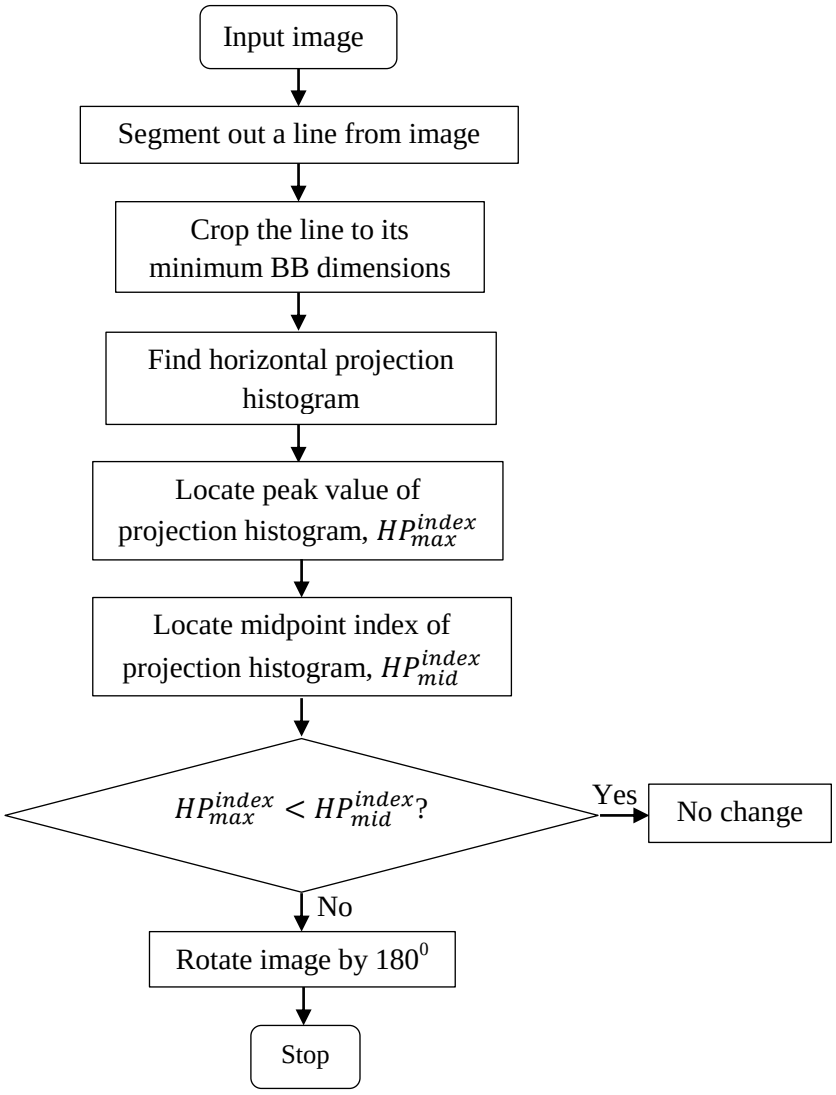


Figure 3.12: Flowchart to correctly align flipped image

3.3 Word segmentation

To separate the words, vertical projection of a segmented line say L_{KxP} is calculated using equation (24). The columns for which $I_{VP}(n)$ is zero act as partition between adjacent words as demonstrated in Figure 3.13.

$$I_{VP}(n) = \sum_{m=1}^{K} \mathrm{L}(m,n) \;\; \forall \, n \;\; \in [1, P] \tag{24}$$

A word subimage W_{UxV} can be simply represented as combination of different constituent subimages:

<Devanagari word: $\boldsymbol{W_{UxV}}$> = <Header line: $\bar{\boldsymbol{H}}$> + <Character set with modifiers: $\boldsymbol{W'}$ >

<Character set with modifiers: $\boldsymbol{W'}$ > = <Ascenders: $\boldsymbol{W_A}$ > + < Descenders: $\boldsymbol{W_D}$ > +

< Individual characters: $\dot{\boldsymbol{W}}$ >

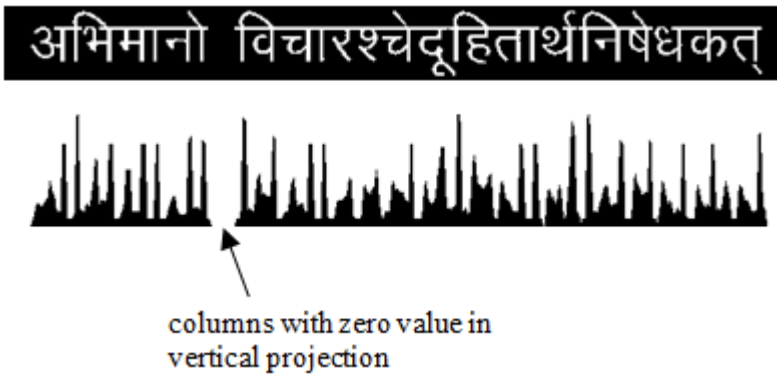


Figure 3.13 Segmented line with its vertical projection showing the columns used for word segmentation

3.4 Character segmentation

Character segmentation process extracts constituent characters from Devanagari word subimage and utilizes following operations for this purpose.

Vertical Projection: For a word subimage W_{UxV}, vertical projection $VP_W(n)$ is given by

$$VP_W(n) = \sum_{m=1}^{U} \mathrm{W}(m,n) \;\; \forall \, n \;\; \in [1, V] \tag{25}$$

Horizontal Projection: Horizontal projection $HP_W(m)$ can be found by traversing entire range of rows from m=1 to m=U and calculating number of white pixels for each row using equation (26).

$$HP_W(n) = \sum_{m=1}^{V} \mathrm{W}(m,n) \;\forall\, m \;\in [1, H] \tag{26}$$

The above operations are employed quite a few times when performing preliminary character segmentation. Preliminary character segmentation comprises of following steps:

Step 1: Removing the header line: The characters of Devanagari word are connected together via header line. To isolate the characters of a word header line needs to be eliminated [1]. To locate the header line ($\bar{H}$) we compute horizontal projection $HP_W(m)$ of corresponding word and the rows corresponding to highest $HP_W(m)$ value contain the header line as shown in Figure 3.14(a). Header line can be computed using following equation (27):

$$\bar{H} = \max_{m\epsilon[1,U]} HP_W(m) \tag{27}$$

Setting the value of these rows to zero essentially removes the header line. Word subimage without header line, W' can be computed by using equation (28):

$$W'_{\mathrm{UxV}} = [W_{\mathrm{UxV}} \,|W_{\mathrm{UxV}}\,(m,n) = 0\; \forall m\epsilon\, \bar{H}, n\epsilon\, [1, V]] \tag{28}$$

Step 2: Separating the characters: After removing header line, residual image W'_{UxV} encompasses core strip fused together with descenders and isolated ascenders. The columns [1] with zero value in vertical projection, given by equation (25), are used as delimiters for extracting character subimages from W'_{UxV} as shown in Figure 3.14(b).

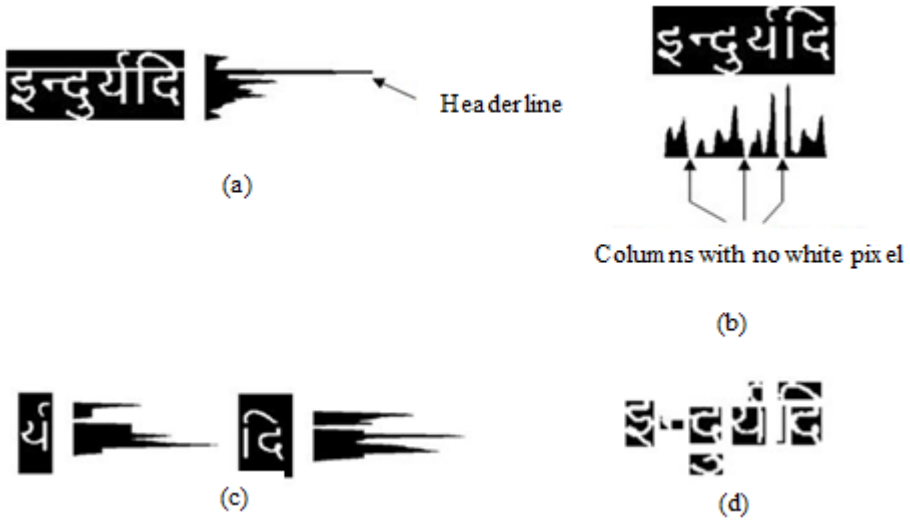


Figure 3.14: Segmentation of a word (a) word image with horizontal projection of word taken to locate header line (b) word image after removing header line with vertical projection to locate segmenting columns (c) segmented core characters with horizontal projection (d) final word after segmentation

Step 3: Separating the top modifiers/ascenders: Removing header line delineates upper strip from rest of the image due to the presence of extending black strip [1] as shown in Figure 3.14(c) and are easily separated using rows with zero value in horizontal projection using equation (26). Result of preliminary segmented word is shown in Figure 3.14(d).

3.4.1 Segmentation of lower modifiers/ descenders

Following preliminary character segmentation in previous step, lower modifiers present below the core strip are removed in this section.

Devanagari script has characters of varying height. During preliminary segmentation the end bar from the following characters is removed due to absence of ON (white) pixels in intermediate columns, hence giving us half characters with their height quite less than that of the average height of characters (denoted by *Avg_Ht*) as shown in Figure 3.15(a). Then there are characters with their height approximately equal to *Avg _Ht* shown in Figure 3.15(b) and characters with lower modifiers with their height greater than *Avg_Ht* as shown in Figure 3.15(c). So image boxes with height either greater than 0.85**Maximum height* or *Avg_Ht* are marked for further segmentation. We use former threshold value because when using *Avg_Ht* as a deciding factor, long character like ह with height slightly greater than *Avg_Ht* is also cropped. This lower cropped portion resembles the lower modifier '*halant*' (्) very closely. To avoid such undesired segmentation we choose 0.85**Maximum height* (denoted as *Thresh_Ht*) as threshold value to distinguish the characters for further segmentation.

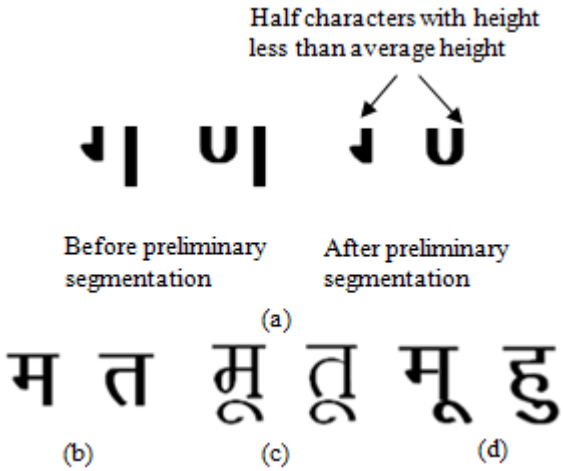


Figure 3.15: Example of Devanagari characters with varying height (a) Half characters with height less than *Avg_Ht* (b) Character with height equal to *Avg_Ht* (c) Height greater than *Avg_Ht* with gap between core character and modifier (d)) Characters with height greater than *Avg_Ht* and joined core character and modifier

When a descender is placed below a core character, one of two possible cases occur:

i) Gap between character and modifier: In some cases modifier do not touch the core character as depicted in Figure 3.15(c). In such cases segmentation of lower modifier is an easy task and can be accomplished by taking horizontal projection of the character and using rows with zero value to separate the modifier as shown in Figure 3.16.

ii) Joined character and modifier: In other cases modifiers are attached to the character below the core strip shown in Figure 3.15(d). In such cases *Thresh_Ht* is used for segmentation. Detailed discussion is given below.

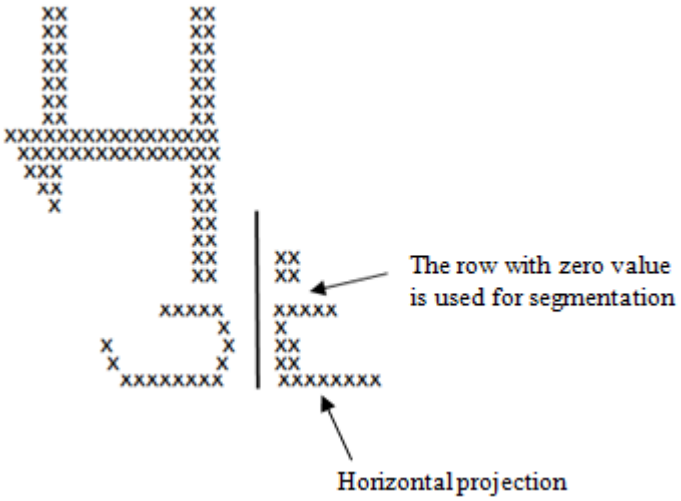


Figure 3.16: Segmentation of character with isolated lower modifier

To locate an appropriate row that separates the core character from descender in image I_D, we utilise the maximum height of characters. On the basis of this threshold (*Thresh_Ht* =0.85* *Maximum height*) segmentation region R_s is evaluated using equation (30) and is shown in Figure 3.17(b). Segmentation region usually incorporates few rows adjacent to the *Thresh_Ht*. Traversing through all the rows in this region we note the column indices that contain first (leftmost) and last (rightmost) pixel respectively shown in Figure 3.17(c). The row say *Seg_row*, with minimum difference between selected *Col_right* and *Col_left* is finalised as segmentation row for separating core character and lower modifier shown in Figure 3.17(c).

$$Thresh_{Ht} = 0.85 * Max_{ht} \quad (29)$$

$$R_s = Thresh_{Ht} \pm 2 \quad (30)$$

$$C^x = \{y | I_D(x,y) = 1 \; ; \; \forall y, x \in R_s\}_n \tag{31}$$

$$Col\_left^x = C^x(1) \; \forall \; x \in R_s \tag{32}$$

$$Col\_right^x = C^x(n) \; \forall \; x \in R_s \tag{33}$$

$$Seg\_row = \min_{x \in R_s} Col\_right^x - Col\_left^x$$

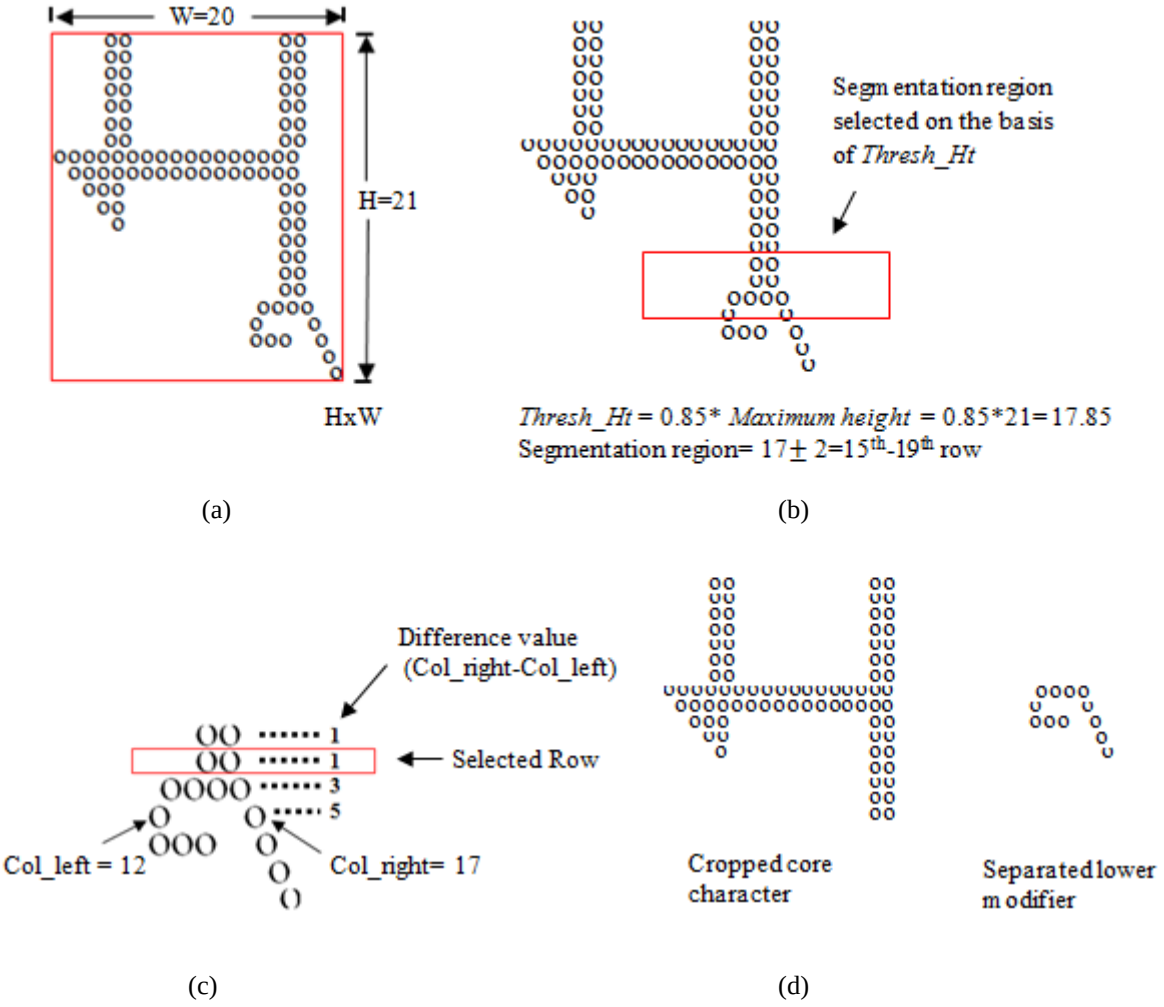
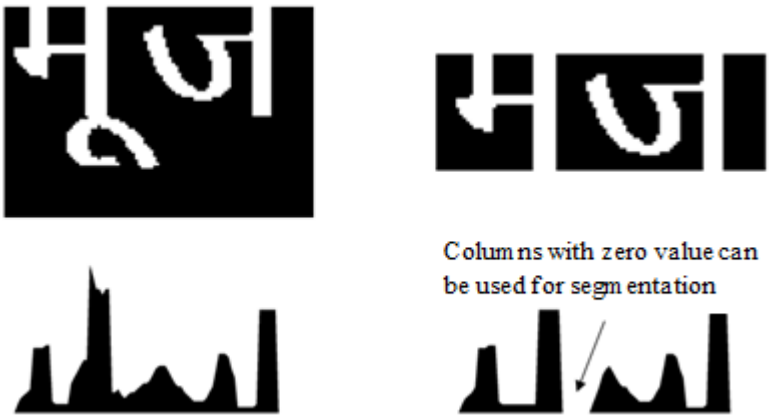


Figure 3.17: Segmentation of characters with joined lower modifier (a) Image I_D of character consisting of lower modifier with dimensions HxW (b) character image with segmentation region within the rectangle (c) Difference value calculated for each row in segmentation region (d) Separated subimages of core character and lower modifier

After removing lower modifiers from core characters, next step involves segmenting those core characters that although are not touching each other but still they cannot be segmented due to absence of valid column with no character pixel. The presence of descender interferes with separation of such non-touching core characters shown in

Figure 3.18(a). These character subimages can be easily segmented using vertical projection once the lower modifier has been removed as shown in Figure 3.18(b).

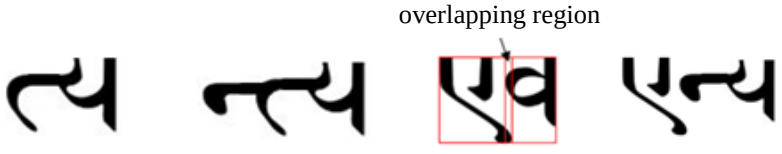


(a) (b)

Figure 3.18: Example of character segmentation with obstructing lower modifier (a) Vertical projection of character image before removing lower modifier (b) Vertical projection of character image after removing lower modifier

3.4.2 Segmentation of conjuncts/touching characters and shadow characters

Conjuncts in Devanagari script are formed when two or more consonants are joined together usually by removing the right portion of former consonant and affixing it next to an intact consonant shown in Figure 3.19(a) and (b). Shadow characters, however, are the those characters which do not touch but overlap one another in such a way that they cannot be segmented without clipping off a portion of either character shown in Figure 3.19(c) and (d). After preliminary character segmentation and removal of lower modifiers, next step involves separation of touching and shadow characters into their constituent character subimages. The character images marked for further segmentation, due to their width, may either contain conjuncts, shadow characters or both.



(a) (b) (c) (d)

Figure 3.19: Conjuncts and shadow characters: (a) Conjunct with combination of two characters (b) Conjunct with combination of three characters (c) Example of shadow characters (d) Image with characters in shadow and touching each other

Primary feature used to separate conjuncts or shadow character is width of the character. Decisive threshold to select characters for further segmentation is based on average width of characters and is chosen based on following observations regarding conjuncts and shadow characters:

i) The width of composite characters is comparable to or greater than twice the average width (*Avg_Wd*) of characters.

ii) The width of a conjunct composed of two constituent Devanagari characters is within close range to twice the average width. Figure 3.20(b) shows example of such conjuncts. Threshold range for such conjuncts is chosen using equation (35):

$$Thresh\_Wd_2 = 1.85 * Avg_{Wd} : 2.55 * Avg_{Wd} \quad (35)$$

iii) The width of a conjunct comprising three constituent Devanagari characters is usually three times the average width. Threshold width chosen for such conjuncts is given by equation (36). Figure 3.20(c) shows example of conjuncts consisting 3 characters joined together.

$$Thresh\_Wd_3 > 2.95 * Avg_{Wd} \quad (36)$$

iv) For composite character image consisting of two characters in shadow or two characters touching each other, the segmentation region is usually the mid region with approximate range equivalent to R_s^1 given by equation (37).

$$R_s^1 = Avg_{Wd} \pm 4 \quad (37)$$

v) For composite character image consisting of three character subimages, the first segmenting column lies in region R_s^1 and second segmenting column lies in region R_s^2 approximately near the twice of *Avg_Wd*, equation (38).

$$R_s^2 = 2 * Avg_{Wd} \pm 4 \quad (38)$$

Using above observations, the resultant segmenting column, required to separate the constituent characters, is calculated.

Respective width of characters	22	20	29	32	43	44

(a)

Mean width (as calculated while processing whole document) = 12
Width range for conjuncts with 2 characters = 22.2 : 35:4
Width for conjuncts with 3 characters > 35:4

Characters selected for further segmentation based on above data:

conjuncts with 2 characters

(b)

conjuncts with 3 characters

(c)

Figure 3.20: Analysing conjuncts on the basis of their width (a) characters with their respective width (b) conjuncts with two characters (c) conjuncts with three characters

Segmentation of conjuncts

Devanagari characters with their width satisfying equation (35) and (36) are marked for further segmentation. The corresponding segmentation region for such characters is mentioned in point (iv) and (v) of preceding section. The appropriate column for cropping out the constituent character subimages is found by applying the following algorithm, within the segmentation region of a composite character.

Step 1: Scan each of the character image to mark the rows that contain first and last white pixel using equation (39)-(41).

$$Row^{y} = \left\{ x \middle| I_D(x,y) = 1 \; ; \; \forall x, y \in R_s^i \right\}_n \tag{39}$$

$$Row\_Top^{y} = Row^{y}(1) \; \forall \; y \in R_s^i \tag{40}$$

$$Row\_Bottom^{y} = Row^{y}(n) \; \forall \; y \in R_s^i \tag{41}$$

Step 2: Next for each column, difference between lowermost and uppermost row is calculated. The column with minimum difference value is chosen for segmentation as shown in Figure 3.22(b).

$$Seg\_col = \min_{y \in R_s^i} (Row\_Bottom^x - Row\_Top^x) \quad (42)$$

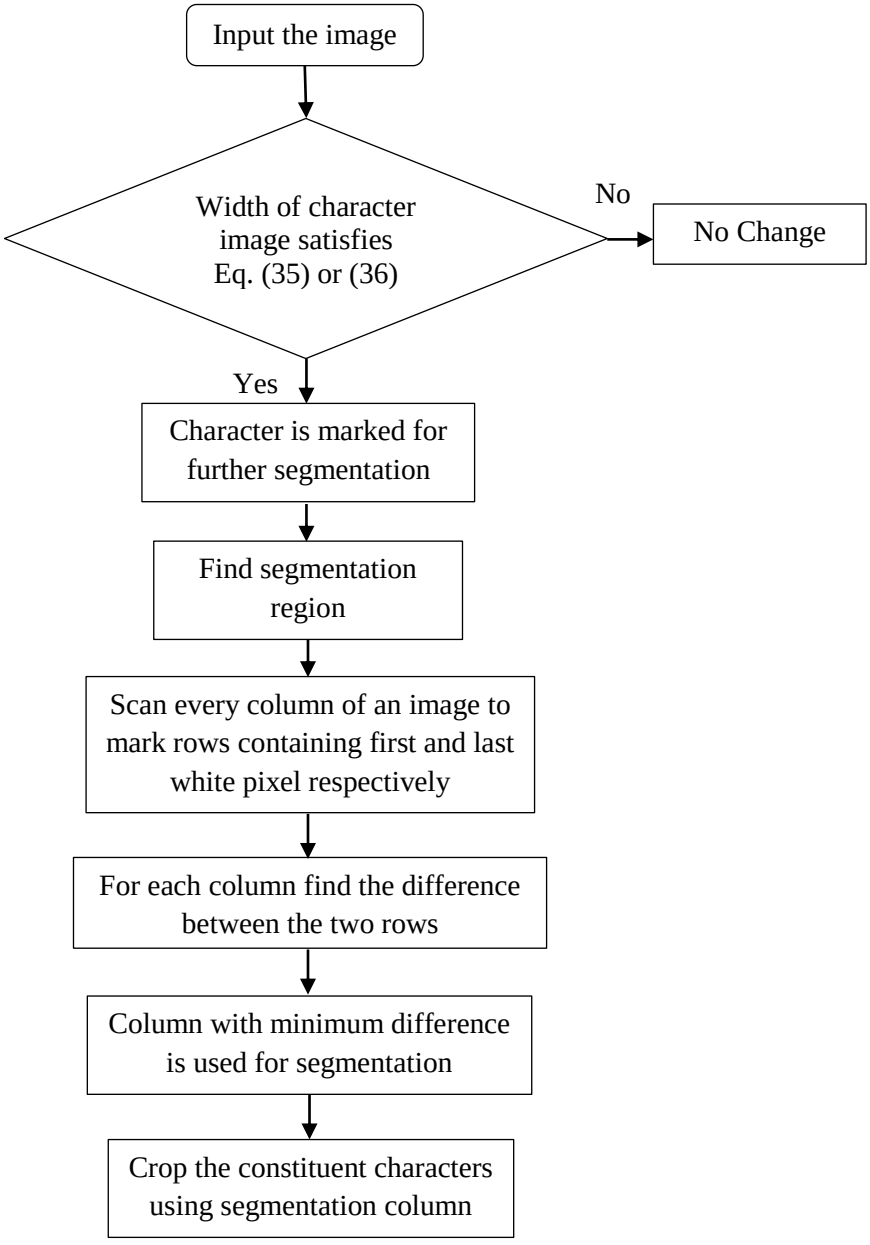


Figure 3.21: Flowchart for segmenting conjuncts

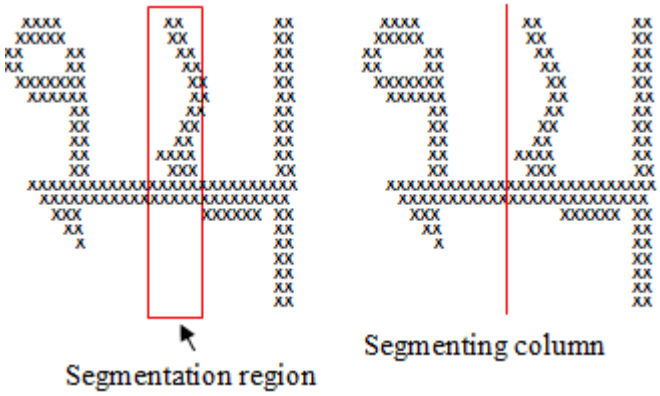


Figure 3.22 Conjunct segmentation (a) Segmentation region in conjunct character (b) Image with resulting segmentation column

For conjuncts made up of two characters, a single segmenting column is required. To determine segmenting column for such conjuncts equations (39)-(42) are used with R_s^i replaced with R_s^1. For conjuncts comprising three core characters first segmenting column is found using equations (39)-(42) with R_s^i replaced with R_s^1 whereas for second segmenting column R_s^i replaced with R_s^2 in equations (39)-(42).

Segmentation of shadow characters

Previous works of segmenting shadow characters [1] find an appropriate segmenting column to separate the constituent characters. When using segmentation column to separate shadow characters, a little portion of either character is clipped off.

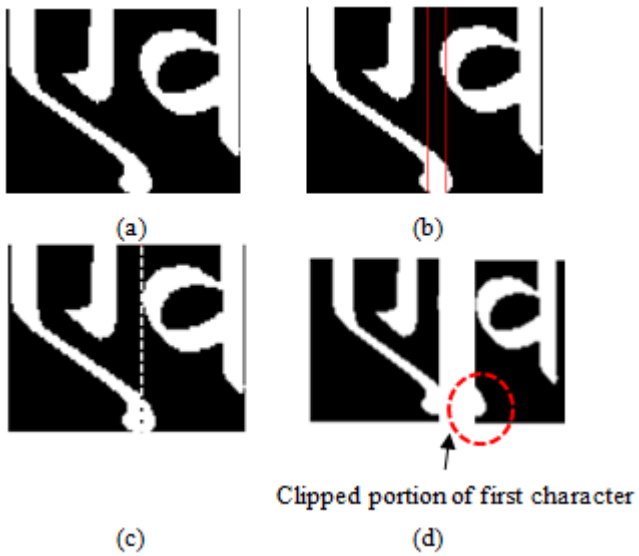


Figure 3.23: Characters in shadow and their segmentation: (a) Characters in shadow (b) Image showing segmentation region within the rectangle (c) Segmentation column represented with dotted line (d) Individual characters after segmentation

Segmentation of shadow characters, using above algorithm, is illustrated in Figure 3.23. As seen in Figure 3.23(d) segmentation crops a small portion off a first character. To avoid such errors we have used an algorithm based on connected component, as depicted in flowchart of Figure 3.24, to isolate shadow characters.

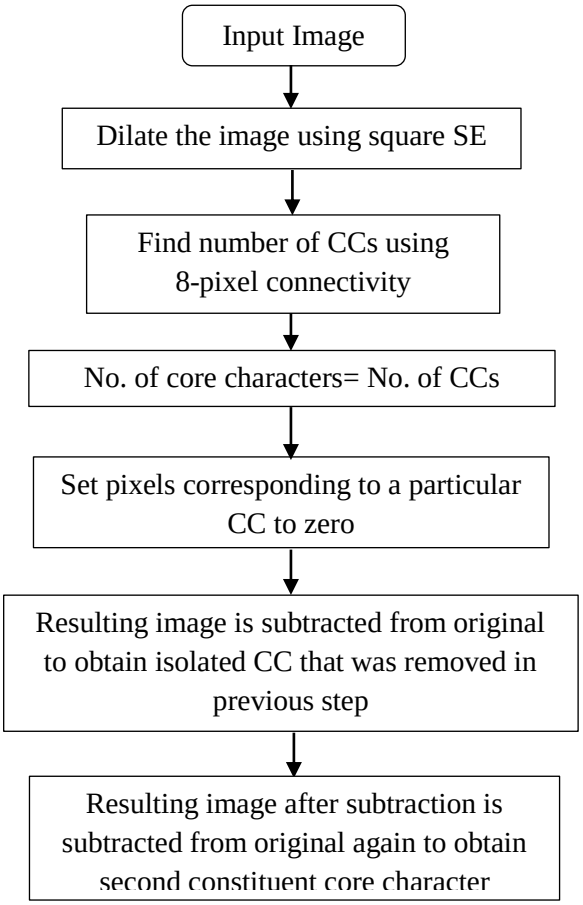


Figure 3.24: Flowchart representing method to isolate shadow characters

Before applying CC algorithm to separate shadow characters, image is closed using a square SE. This action fills any small gap present in constituent character subimages as demonstrated in Figure 3.25.

(a) (b)

Figure 3.25: Shadow characters (a) Example of shadow character with gap in constituent character shown with dotted circle (b) dilated image with number of core characters corresponding to number of CCs

Resulting dilated image has number of 8-connected components equivalent to total number of core characters. Figure 3.25(a) shows two characters in shadow with each other with three connected components and dilated image in Figure 3.25(b) has two core characters corresponding to two connected components.

In resulting dilated image number of CCs is calculated and corresponding pixels of one component are set to zero as shown in Figure 3.26. Subtracting this image from original gives an isolated character that was removed previously and further subtracting this isolated character from original image results in subimage of another constituent character.

(a) (b)

Figure 3.26: Shadow character segmentation (a) Dilated image with shadow character (b) constituents characters obtained after subtraction

3.5 Feature extraction and classification of modifiers

To distinguish different modifiers from one another, the modifiers are skeletonized down to unitary thickness before extracting features using 8-pixel adjacency. For each non-zero pixel $M_{IxQ}(r,c)$ (Figure 3.27) in a skeletonized modifier subimage, number of foreground neighbouring pixels (*NP*) can be found using equation (43).

$$NP = \sum_{c=c-1}^{c+1} \sum_{r=r-1}^{r+1} M(r,c) - 1 | M(r,c) = 1 \quad (43)$$

Extreme ends of skeletonized modifiers are the pixels with just one 8-adjacent element and are calculates using following equation (44):

$$M_{end}(r,c) = \{M(r,c) | NP = 1\} \quad (44)$$

For such pixels following features are calculated using equations (45)-(48):

$$\text{Left\_neighbours} = \sum_{r=r-1}^{r+1} M(r,c-1) \quad (45)$$

$$\text{Right\_neighbours} = \sum_{r=r-1}^{r+1} M(r, c+1) \tag{46}$$

$$\text{Top\_neighbours} = \sum_{c=c-1}^{c+1} M(r-1, c) \tag{47}$$

$$\text{Bottom\_neighbours} = \sum_{c=c-1}^{c+1} M(r+1, c) \tag{48}$$

M(r-1,c-1)	*M(r-1,c)*	*M(r-1,c+1)*
M(r,c-1)	*M(r,c)*	*M(r,c+1)*
M(r+1,c-1)	*M(r+1,c)*	*M(r+1,c+1)*

Figure 3.27: Example of 8-connected neighbours with origin at *M(r,c)*

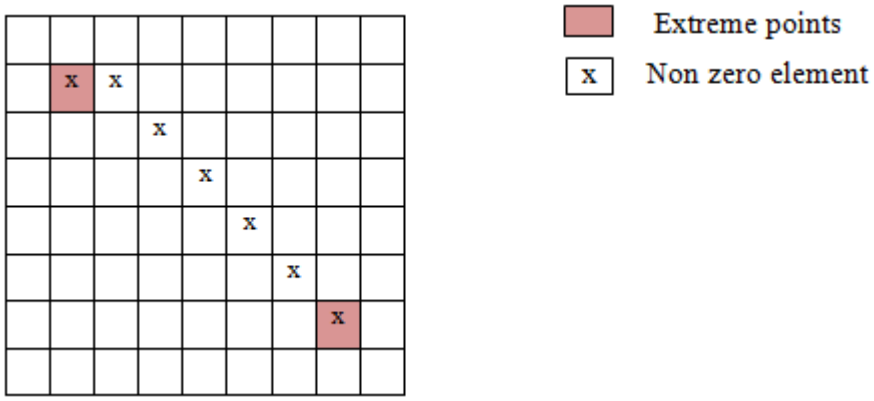


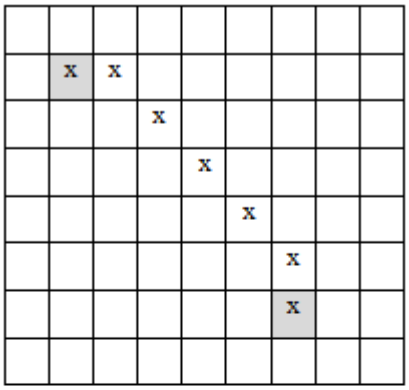
Figure 3.28: Modifier image showing extreme points

Most of the upper-strip elements have one CC except for (`) as shown in Figure 3.29 and hence it can easily be recognised from all other modifiers with single CC. The modifiers with a single connected component have two extreme points as shown in Figure 3.28. For both of these pixels respective features given by equations (45)-(48) are used to

differentiate between modifiers. An example illustrating how these features are used to differentiate between different modifiers is shown in Figure 3.30.

(a) (b)

Figure 3.29: Top strip modifier components (a) with one CC (b) with two CCs



$Left\_neighbours_{start} = 0$

$Right\_neighbours_{start} = 1$

$Top\_neighbours_{start} = 0$

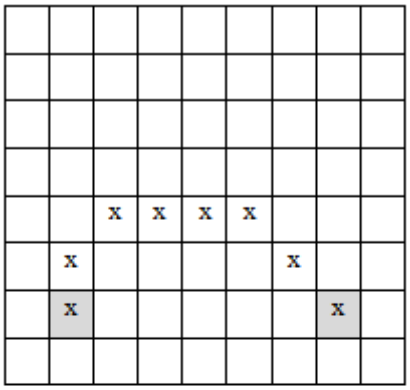
$Bottom\_neighbours_{start} = 0$

$Left\_neighbours_{end} = 0$

$Right\_neighbours_{end} = 0$

$Top\_neighbours_{end} = 1$

$Bottom\_neighbours_{end} = 0$



$Left\_neighbours_{start} = 0$

$Right\_neighbours_{start} = 0$

$Top\_neighbours_{start} = 1$

$Bottom\_neighbours_{start} = 0$

$Left\_neighbours_{end} = 1$

$Right\_neighbours_{end} = 0$

$Top\_neighbours_{end} = 1$

$Bottom\_neighbours_{end} = 0$

Figure 3.30: Example illustrating neighbouring features of two different modifiers

3.6 Feature extraction for character recognition

In the present section we discuss features which will be utilized to recognize Devanagari characters.

Zoning: To increase the character recognition accuracy, the image is partitioned into non-overlapping non-uniform regions zones as shown in Figure 3.31 and number of character pixels is calculated for each of these zones.

To find number of pixels in a particular zone, a mask of same size as that of character image is formed such that all the coefficients outside respective zone are zero as shown in Figure 3.32.

In general, for image I_{MxN} with corresponding mask w of same size, number of foreground pixels (NOP_i) in i^{th} region is given by equation (49).

$$NOP_i = \sum_{p=1}^{M} \sum_{q=1}^{N} I(p,q) * w_i(p,q) \tag{49}$$

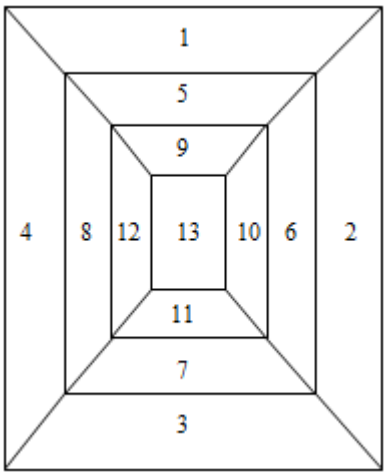
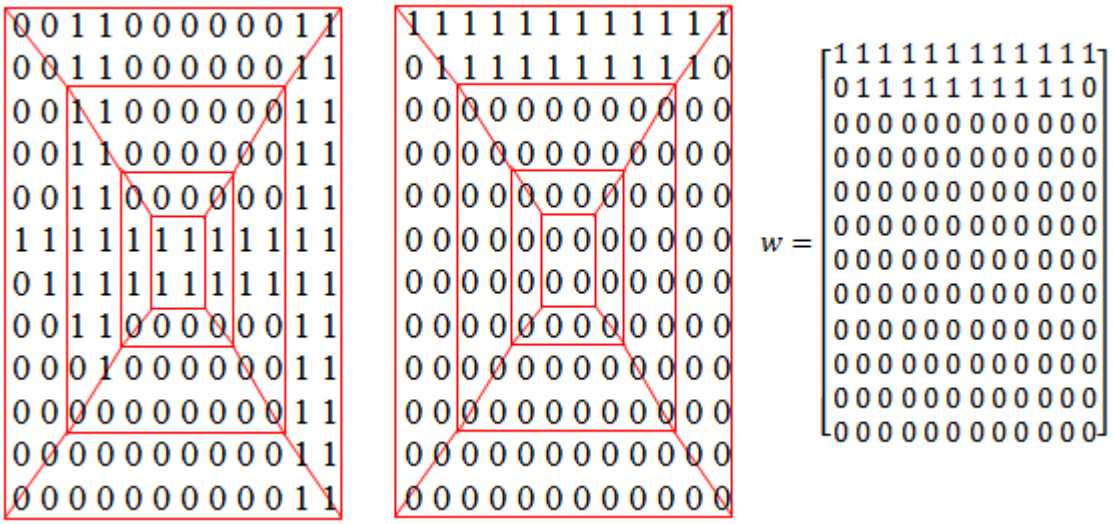


Figure 3.31: Grid showing 13 different non-uniform zones



(a) (b) (c)

Figure 3.32: Zoning and corresponding mask formation (a) Character image partitioned in different zones (b) mask corresponding to first region (c) corresponding matrix for the mask

Crossings: Crossings depicts number of transitions from foreground to background pixels along columns and rows in a character image I_{MxN}

Horizontal transitions:

$$HT(m) = card(\, I(m,n) | I(m,n) = 1 \wedge I(m,n+1) = 0\,; \quad \forall\, m \in M, n \in N-1 \tag{50}$$

Vertical transitions:

$$VT(n) = card(\ I(m,n) | I(m,n) = 1 \wedge\ I(m+1,n) = 0\ ;\quad \forall\ m \in M-1, n \in\ N \quad (51)$$

Projection histograms: Projection histograms compute number of foreground pixels in each row and column of character image I_{MxN}.

Horizontal projection histogram:

$$HP(m) = \sum_{n=1}^{N} I(m,n)\ \forall\ m \in [1,M] \quad (52)$$

Vertical projection histogram:

$$VP(n) = \sum_{m=1}^{M} I(m,n)\ \forall\ n \in [1,N] \quad (53)$$

Moments: Instead of using image moments to find weighted average of image pixels' intensities, we find different order moments for 1-D feature matrix obtained after calculating projection histogram and crossings for a character image using equations (50)-(53).

Various order moments such as mean, variance, skewness and kurtosis are computed for resulting 1-D matrix obtained using above equation. First order moment, mean calculates average value or the value around which central clustering occurs. The second order moment is variance and it characterizes variability around the mean. The third and fourth moments are skewness and kurtosis respectively. Skewness characterizes degree of asymmetry of feature vector around its mean whereas kurtosis measures peakedness or flatness of a distribution relative to normal distribution. These four moments are determined for four 1-D feature matrices obtained after computing projection histograms and crossings for a character image, thereby resulting in a 1-D matrix with total of 16 values. For projection and transition features vector *HT(m),VT(n),HP(m),VP(n)*, various order moments are found using formulas given in the table 1.

Table 3.1: Formulas for calculating different transition and projection moments

	Transition moments	Projection moments
Mean	Horizontal transition mean $\overline{HT} = \frac{1}{M}\sum_{j=1}^{M} HT_j$ Vertical transition mean $\overline{VT} = \frac{1}{N}\sum_{j=1}^{N} VT_j$	Horizontal projection mean $\overline{HP} = \frac{1}{M}\sum_{j=1}^{M} HP_j$ Vertical projection mean $\overline{VP} = \frac{1}{N}\sum_{j=1}^{N} VP_j$
Variance	Horizontal transition variance $Var_{HT} = \frac{1}{M}\sum_{j=1}^{M}(HT_j - \overline{HT})^2$ Vertical transition variance $Var_{VT} = \frac{1}{N}\sum_{j=1}^{N}(VT_j - \overline{VT})^2$	Horizontal projection variance $Var_{HP} = \frac{1}{M}\sum_{j=1}^{M}(HP_j - \overline{HP})^2$ Vertical projection variance $Var_{VP} = \frac{1}{N}\sum_{j=1}^{N}(VP_j - \overline{VP})^2$
Skew	Horizontal transition skew $Skew_{HT} = \frac{1}{M}\sum_{j=1}^{M}\left(\frac{HT_j - \overline{HT}}{\sigma_{HT}}\right)^3$ Vertical transition skew $Skew_{VT} = \frac{1}{N}\sum_{j=1}^{N}\left(\frac{VT_j - \overline{VT}}{\sigma_{VT}}\right)^3$	Horizontal projection skew $Skew_{HP} = \frac{1}{M}\sum_{j=1}^{M}\left(\frac{HP_j - \overline{HP}}{\sigma_{HP}}\right)^3$ Vertical projection skew $Skew_{VP} = \frac{1}{N}\sum_{j=1}^{N}\left(\frac{VP_j - \overline{VP}}{\sigma_{VP}}\right)^3$
Kurtosis	Horizontal transition kurtosis $Kurt_{HT} = \left\{\frac{1}{M}\sum_{j=1}^{M}\left(\frac{HT_j - \overline{HT}}{\sigma_{HT}}\right)^4\right\} - 3$ Vertical transition kurtosis $Kurt_{VT} = \left\{\frac{1}{N}\sum_{j=1}^{N}\left(\frac{VT_j - \overline{VT}}{\sigma_{VT}}\right)^4\right\} - 3$	Horizontal projection kurtosis $Kurt_{HP} = \left\{\frac{1}{M}\sum_{j=1}^{M}\left(\frac{HP_j - \overline{HP}}{\sigma_{HP}}\right)^4\right\} - 3$ Vertical projection kurtosis $Kurt_{VP} = \left\{\frac{1}{N}\sum_{j=1}^{N}\left(\frac{VP_j - \overline{VP}}{\sigma_{VP}}\right)^4\right\} - 3$

3.7 Recognition

Recognition stage comprises of two phases. In the first phase of recognition, library character samples are sorted according to minimum Euclidean distance between library feature vector and input feature vector. For the first phase, feature vector S_{reg} comprises of 13 features each corresponding to number of foreground pixels in ith region (equation (49)) and S_{reg}^{i} represents ith feature in the corresponding feature vector S_{reg}.

$$S_{reg} = [NOP_i] \ \forall \ i \in [1,13] \tag{54}$$

Euclidean distance between library feature vector S_{reg}^{i} and input feature vector X_{reg}^{i}can be calculated using equation (55).

$$D_{reg} = \sqrt{\sum_{i=1}^{13} (S_{reg}^{i} - X_{reg}^{i})^2} \tag{55}$$

Where S_{reg}^{i} is i^{th} region feature for library character sample and X_{reg}^{i} is i^{th} input feature.

For the second phase of recognition 15 sorted library character samples are selected with minimum Euclidean distance and from these selected samples nearest neighbour to input character is found. To find nearest neighbour Euclidean between input feature vector and library feature vector equation (56) is used.

$$D_{mom} = \sqrt{\sum_{i=1}^{16} (S_{mom}^{i} - X_{mom}^{i})^2} \tag{56}$$

Where S_{mom} is feature vector containing all the moments given in table 1 and S_{mom}^{i} represents i^{th} feature in corresponding vector. X_{mom}^{i} is i^{th} moment feature in input feature vector X_{mom}.

Input character is matched with the library sample that has minimum distance, D_{mom}.

3.7.1 Inadequate classification due to similarity in character pairs after removing header line

In some cases removing complete header line clips off significant part of a character and hence making it appear similar to another character. Some character pairs, shown in Figure 3.33(a), differ only in header line region. After removing header line, it becomes

difficult to distinguish these characters, causing classification errors as seen in Figure 3.33(b).

To distinguish these characters after removing header line, we compute number of CCs and on basis of number of CCs we classify them in the correct group as shown in flowchart in Figure 3.34.

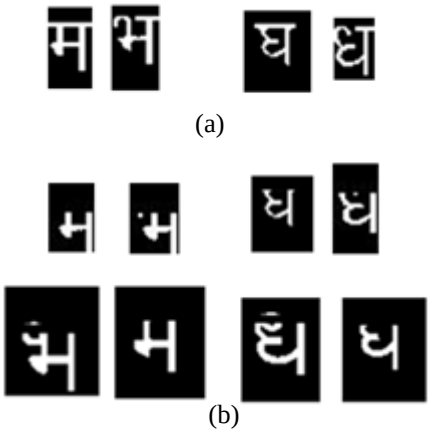


Figure 3.33: (a) Character pairs with similar structure except header line portion (b) indistinguishable character pairs after removing header line

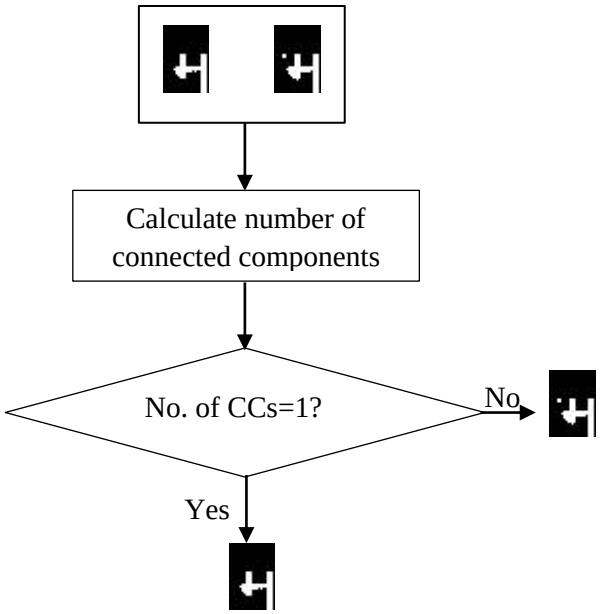


Figure 3.34: Flowchart distinguishing between similar looking characters after header line removal

3.7.2 Inadequate character classification due to rakar modifier

Combination of a consonant and rakar modifier goes unnoticed by segmentation algorithm because the resulting composite character has similar dimensions as any other basic core character. Few examples of consonant-rakar combination are shown in Figure 3.5. Probability of occurrence of such characters is far too minimal to have any considerable effect on the accuracy of proposed algorithm except for प्र. In transliteration algorithm, the character प्र is mapped to प in most of the cases. We use number of end points as a decisive feature to accurately classify these characters. For skeletonized character image [9], number of end points is computed using equation (44). If the character image has 4 extreme points it is classified as प्र and if the character image has 3 extreme points it is classified as प.

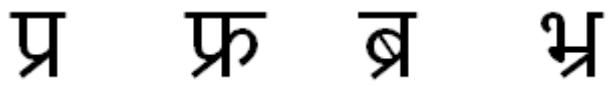


Figure 3.35: Few examples of consonant-rakar combination

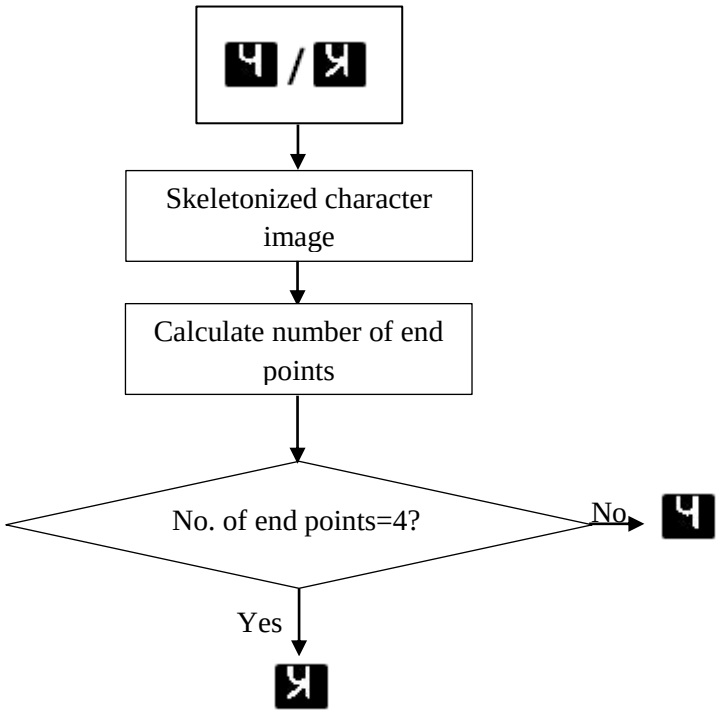


Figure 3.36: Flowchart for accurate classification of a consonant–rakar combination

3.8 Transliteration

The mapping between Devanagari and roman script character in a transliteration scheme should be as close as possible to the pronunciation of the source character in the target script. We have employed International Alphabet of Sanskrit Transliteration (IAST) scheme for romanization of recognized Devanagari characters. Figure 3.37 shows phonetically similar roman alphabets for each Devanagari character.

अ	आ	इ	ई	उ	ऊ
a	ā	i	ī	u	ū
ए	ऐ	ओ	औ	अं	अः
e	ai	o	au	aṁ	aḥ
ऋ	ॠ	ऌ	ॡ		
ṛ	ṝ	ḷ	ḹ		
◌ं	◌ः	◌ॅ			
ṁ	ḥ	□			
क	ख	ग	घ	ङ	
ka	kha	ga	gha	ṅa	
च	छ	ज	झ	ञ	
ca	cha	ja	jha	ña	
ट	ठ	ड	ढ	ण	
ṭa	ṭha	ḍa	ḍha	ṇa	
त	थ	द	ध	न	
ta	tha	da	dha	na	
प	फ	ब	भ	म	
pa	pha	ba	bha	ma	
य	र	ल	व		
ya	ra	la	va		
श	ष	स	ह		
śa	ṣa	sa	ha		

Figure 3.37: Phonetically similar Roman alphabet for each Devanagari character (IAST scheme)

CHAPTER -4

RESULTS AND PERFORMANCE ANALYSIS

This chapter of the dissertation presents the experimental results of transliteration of several Devanagari document images and detailed analysis of the results.

4.1 Experimental results

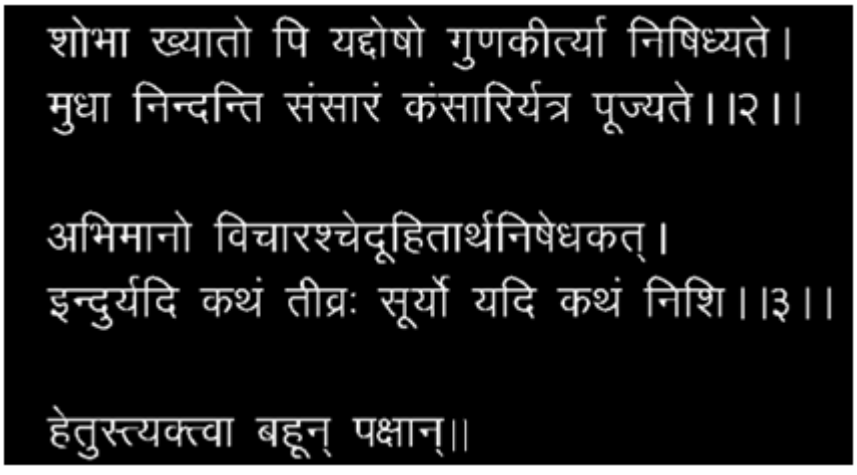
The proposed transliteration algorithm has been tested for several documents of varying Devanagari font styles. The proposed algorithm recognizes the characters extracted from corresponding Devanagari document image and results in a text file containing Roman character corresponding to each recognized Devanagari character.

The proposed algorithm has been implemented using Matlab R2014a programming. Segmentation, recognition and Romanization results for few document images have been presented in this section. The accuracy of algorithm depends on font style and on an average is about 80% when only first choice of true characters is considered. Out of composite characters marked for further segmentation, 5% are wrongfully segmented resulting in recognition and transliteration errors.

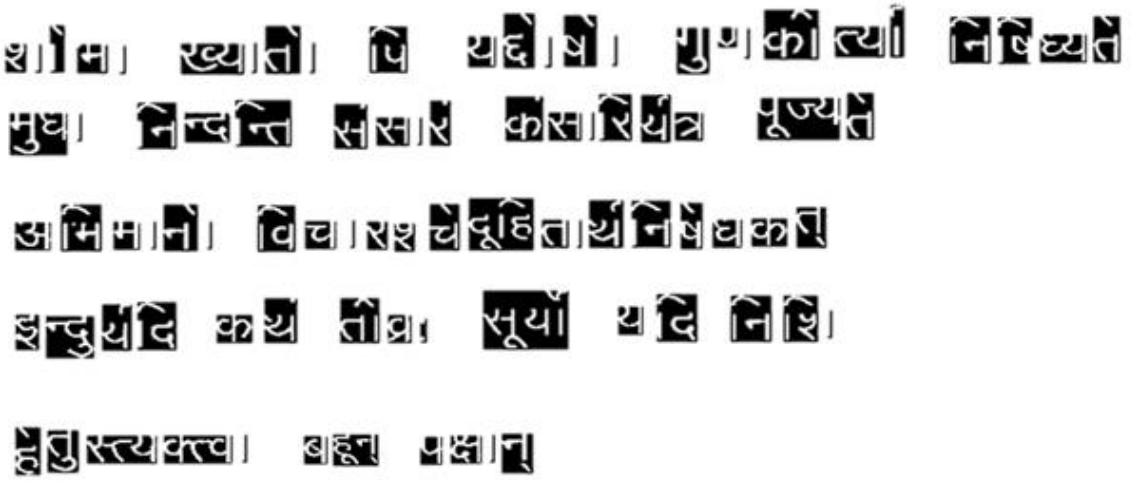
Following two points aid in interpreting the given transliteration results:

i) The characters underlined in red represent true characters that are not first choice.
ii) Underscore in transliteration result represents a modifier that has not been recognized.

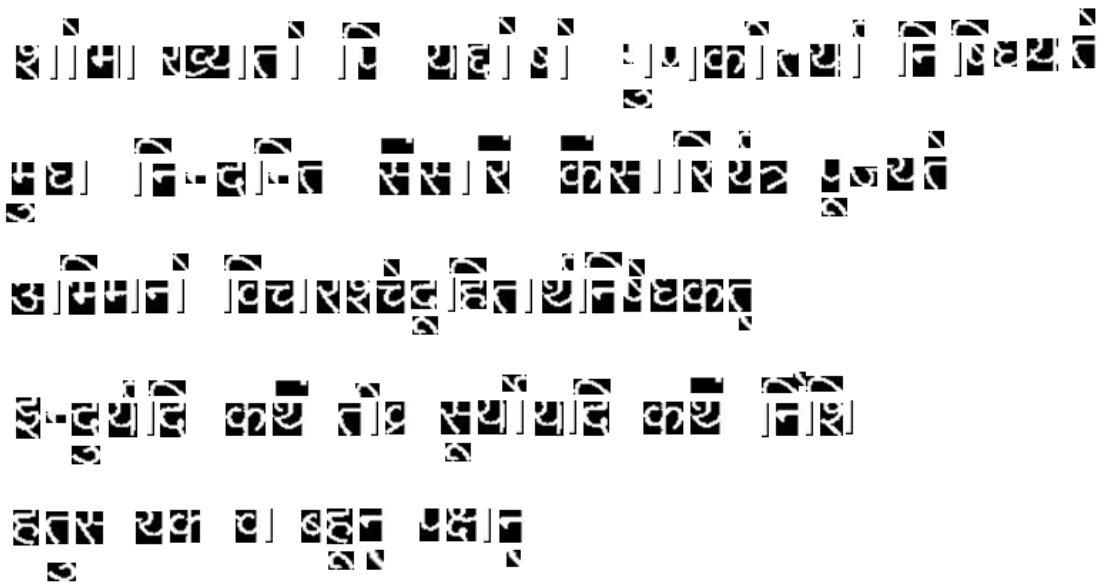
Results for Image 1:



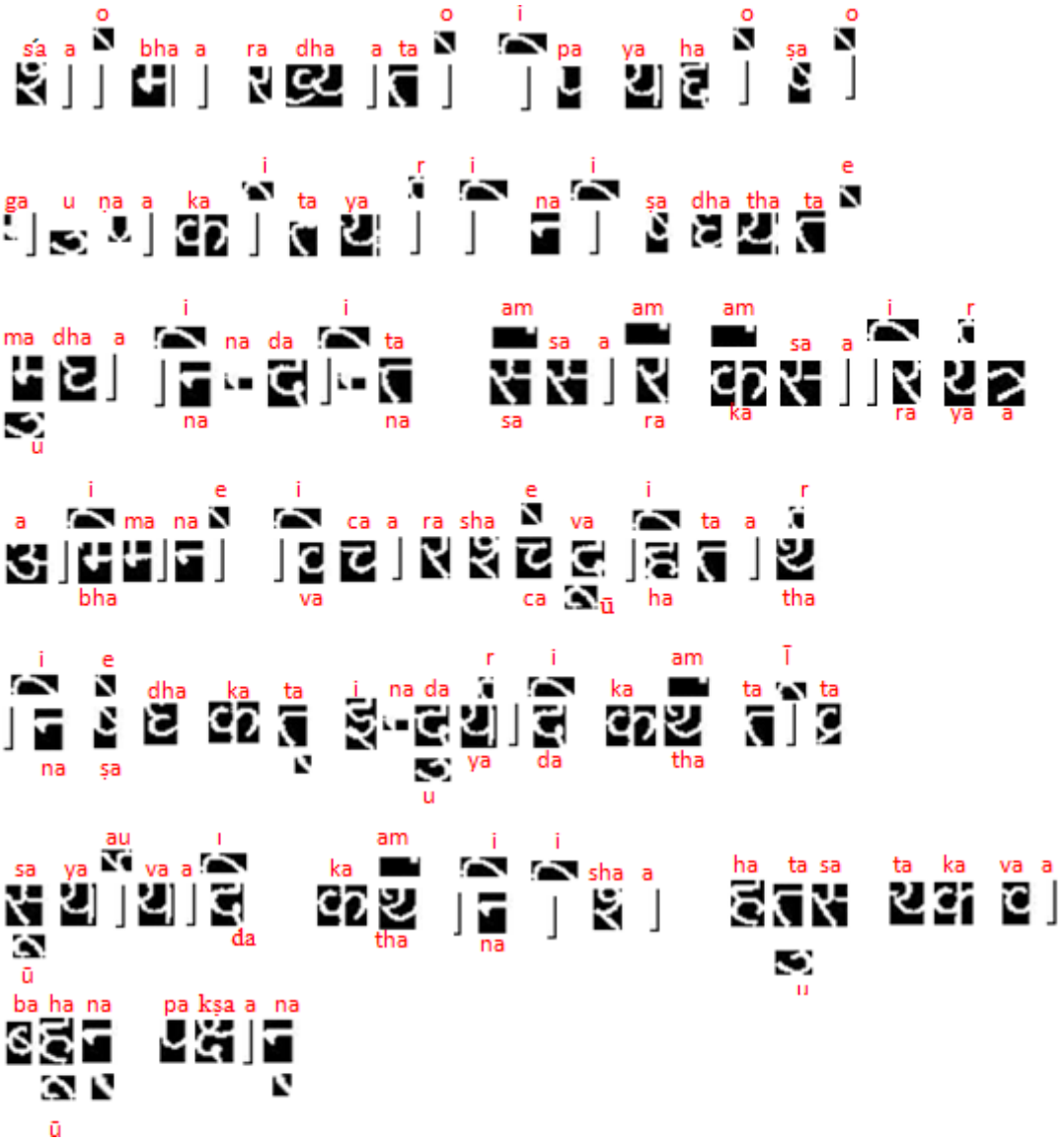
(a) Binary image corresponding to input image



(b) Preliminary character segmentation



(c) Image after composite character segmentation



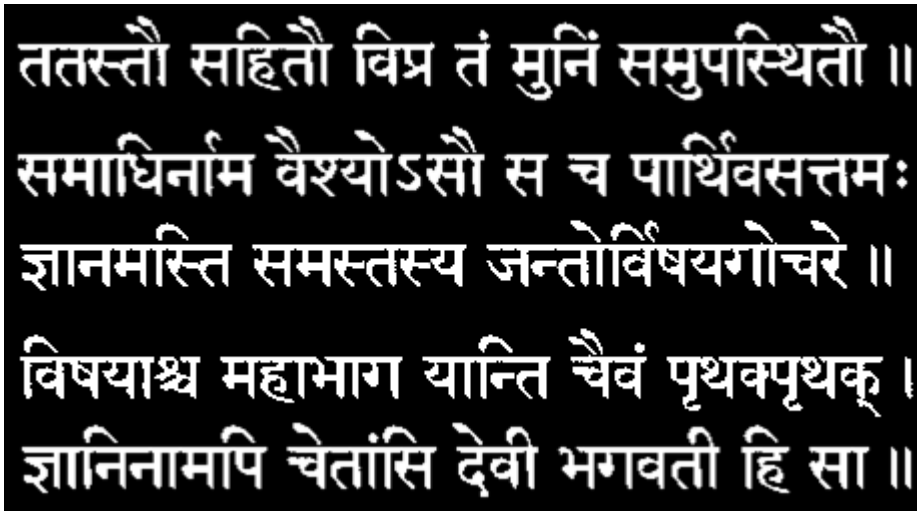
(d) Romanization of segmented characters

ś́aaobhaa radhaatao pai yahaoṣa gauṇaakaitayar naiṣaidhathatae
maudhaa nainadanaita saamsaaraam kaamsarairyaa paūkhaghatae
abhaimanae vaicaarashacaevauhaitaarthanaiṣadhakata
inadauryaida kathaam taīta saūyaau vaadai katham
naishaia hatausatakavao bahaūna pakṣaana

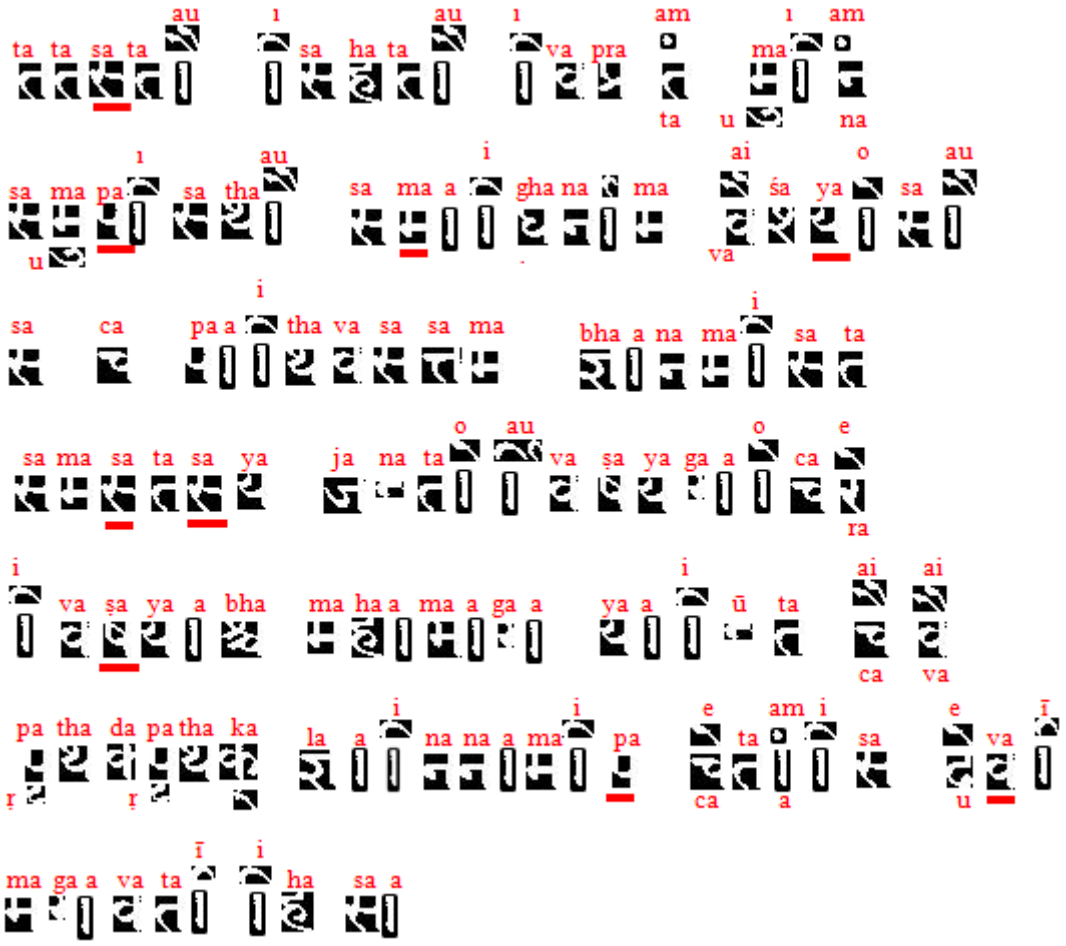
(e) Result of transliteration of Image 1

Figure 4.1: Results for Image 1

Results for Image 2:



(a) Binary image corresponding to Image 2



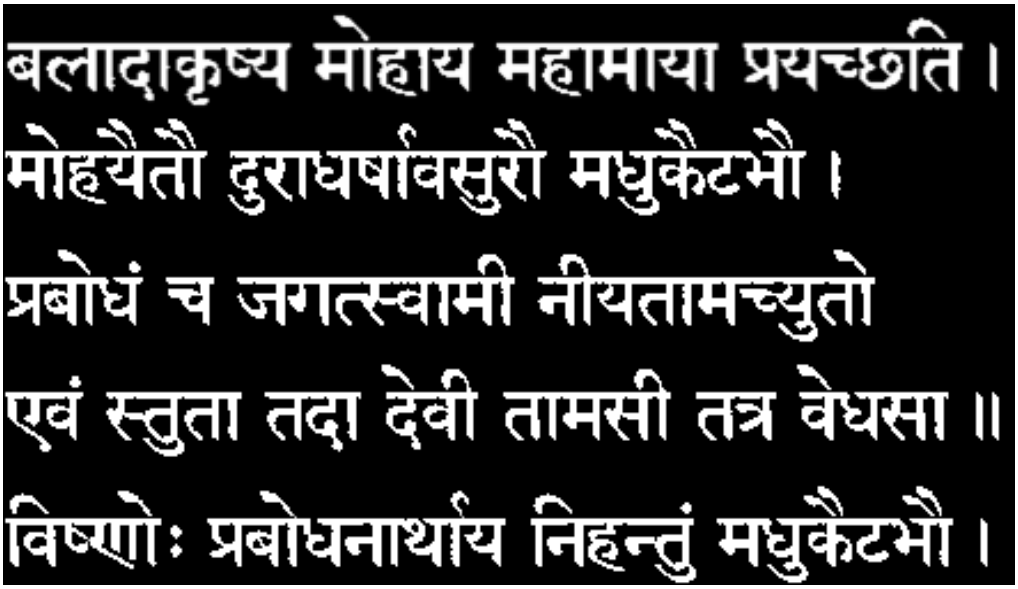
(b) Segmented characters with corresponding roman alphabets

Tatasataau saihataau vaipra taam maunaiam samaupasaithaau
Samaaghaina_ma vaaiśayaosaau sa ca paathaivasasama
Bhaanamasaita samasatasaya janataovaauṣayagaocarae
vaiṣayaabha mahaamaagaa yaaūta caaivaaipaṛthadapaṛthaka_
laanainaamapai caetaamasai daevaī magaavataī haisaa

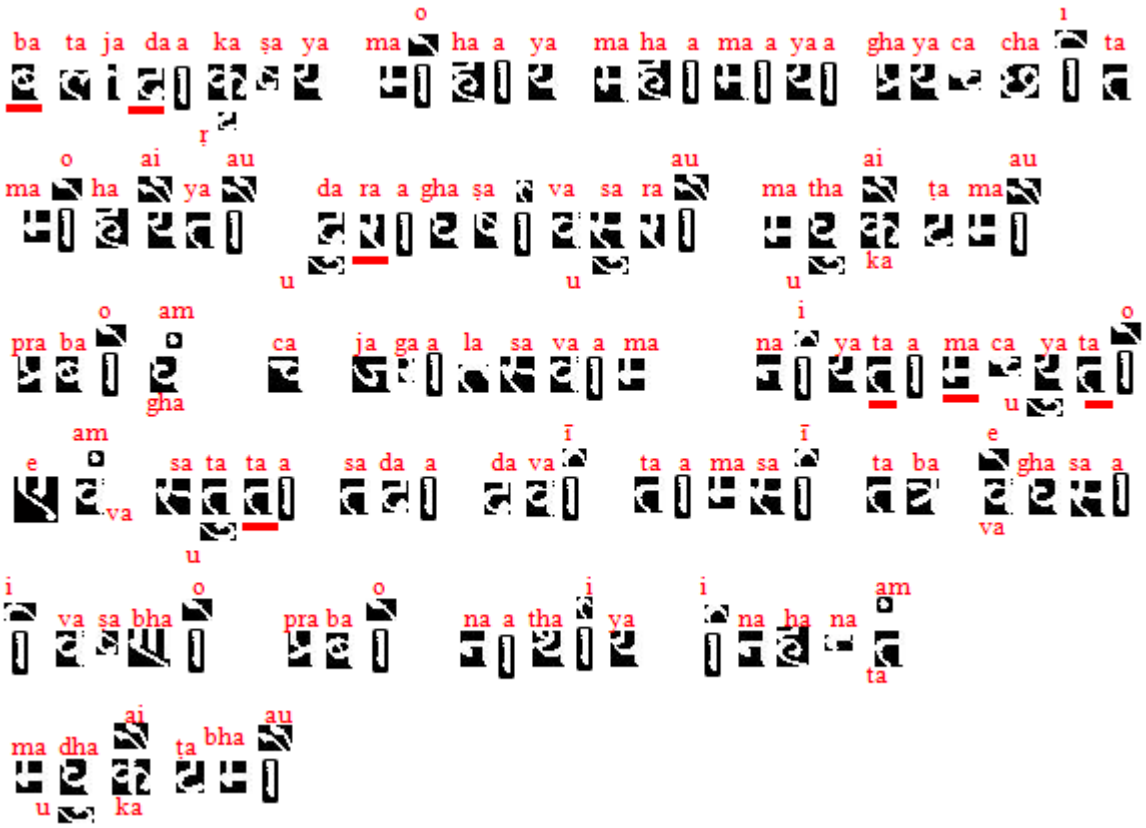
(c) Result of transliteration of Image 2

Figure 4.2: Results for Image 2

Results for Image 3:



(a) Binary image corresponding to Image 3



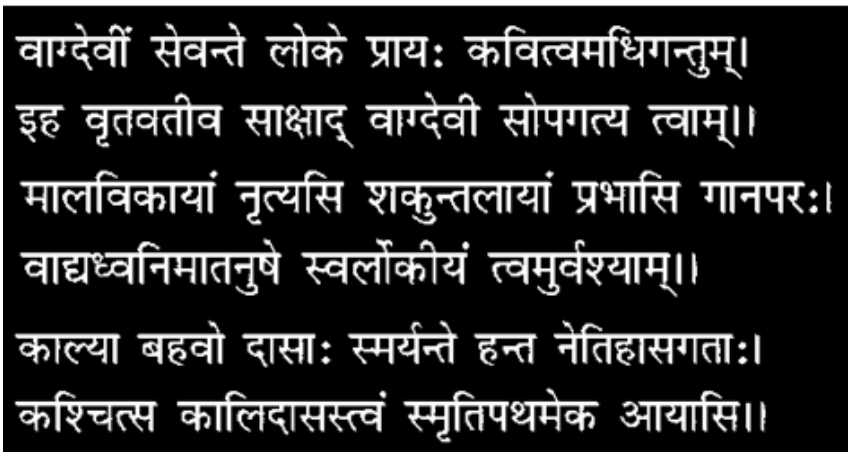
(b) Segmented characters with corresponding roman alphabets

batajadaakaṛṣaya maohaaya mahaamaayaa ghayacachatai
maohaaiyaau dauraaghaṣa_vasauau mathaukaaiṭamaau
prabaoghaam ca jagaalasavaama naiyataamacayautao
evaam satautaa sadaa davaī taamasaī taba ghaesaa
vaiṣabhao prabao naathaiya naihanataam madhaukaaiṭabhaau

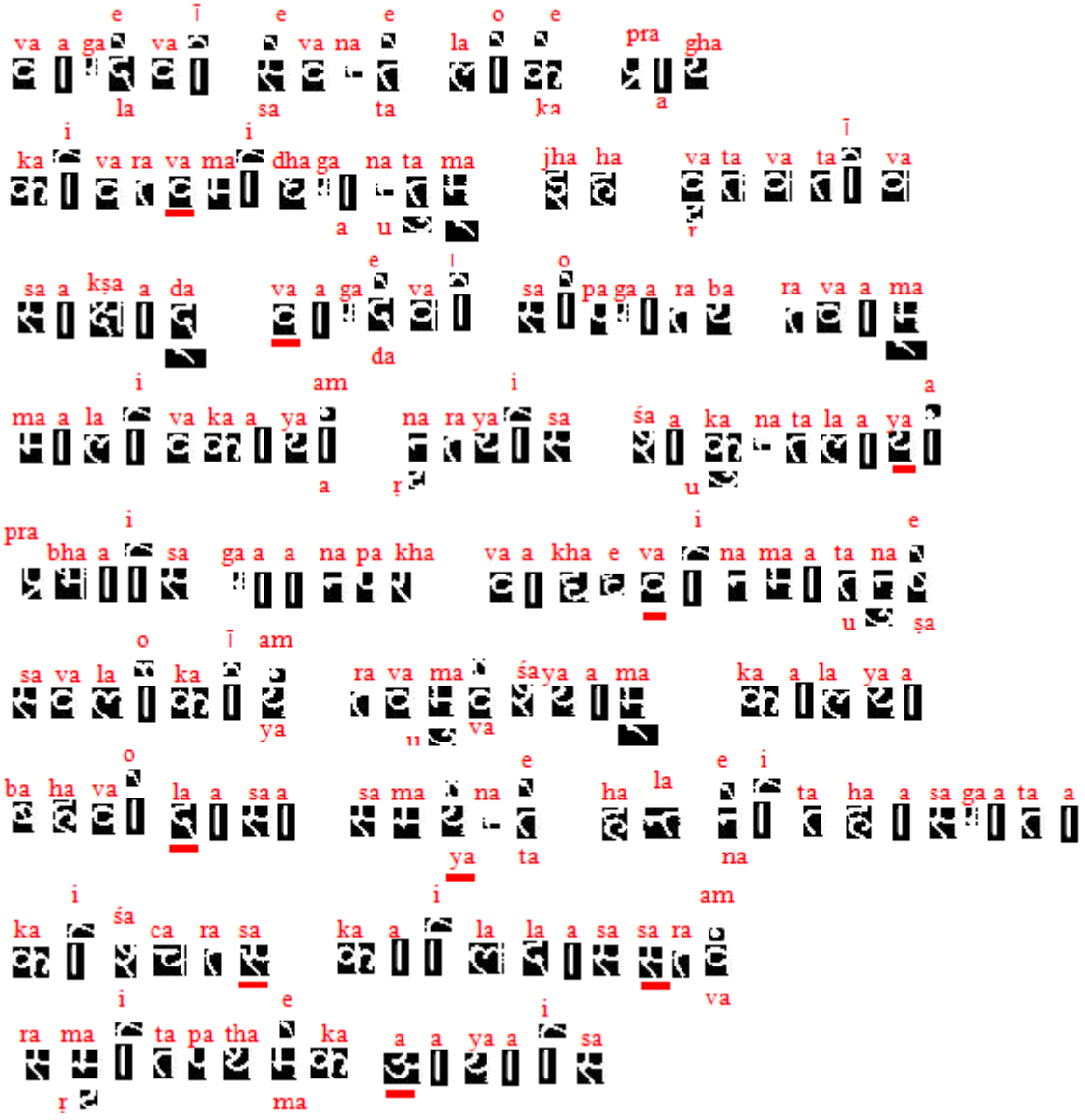
(c) Result of transliteration of Image 3

Figure 4.3: Results for Image 3

Results for Image 4:



(a) Binary image corresponding to Image 4



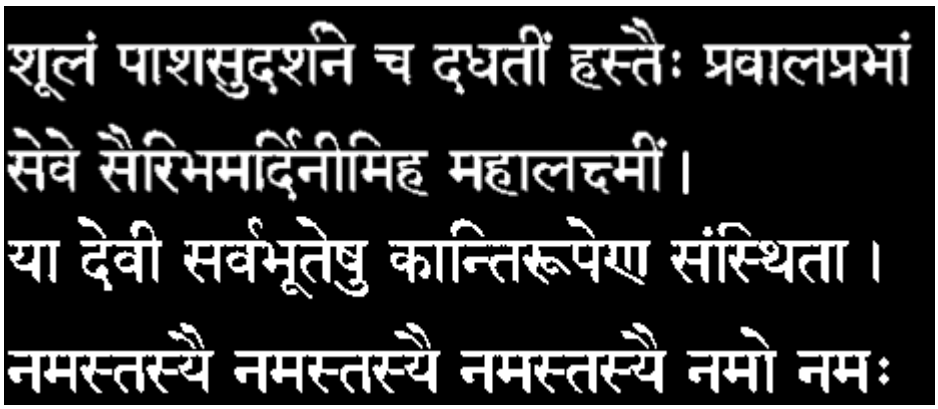
(b) Segmented characters with corresponding roman alphabets

Vaagalaevaī saevanatae laokae pragha kavairavamadhaigaanatauma_
Jhaha vaṛtavataīva saakṣaada_ vaagadaevaī saopagaaraba ravaama_
Maalavaikaayaaam narṛayasai śaakaunatalaayaa prabhasai gaaanapakha
Vaakhaevanaimaatanauṣae savalaokaīyaam ravamauvaśayaama_
Kaalayaa bahavao laasaa samaya_natae hala naetaitahaasagataa
Kaiśacarasa kaalailalaasasaravaam ramaṛtaipathamaeka aayaasai

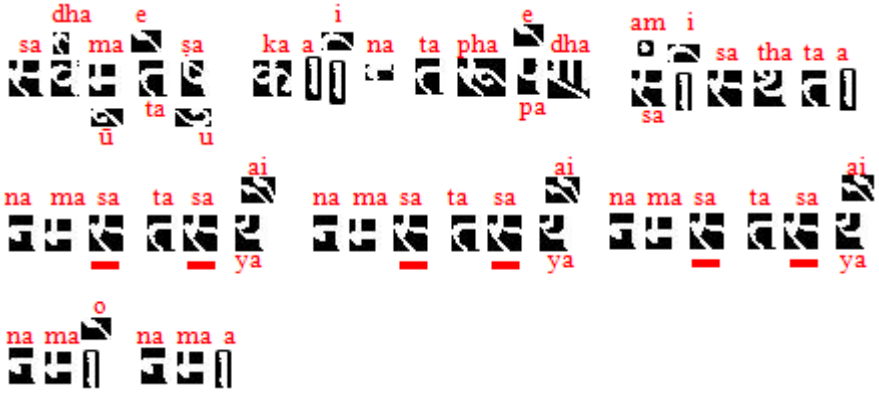
(c) Result of transliteration of Image 4

Figure 4.4: Results for Image 4

Results for Image 5:



(a) Binary image corresponding to Image 5



(b) Segmented characters with corresponding roman alphabets

Śaalaūam paaśasaudaśaina ca dadhataī hasatai pravalaprabhaam

Saevae saairaibhamaausanaīmaiha mahaalasamaau

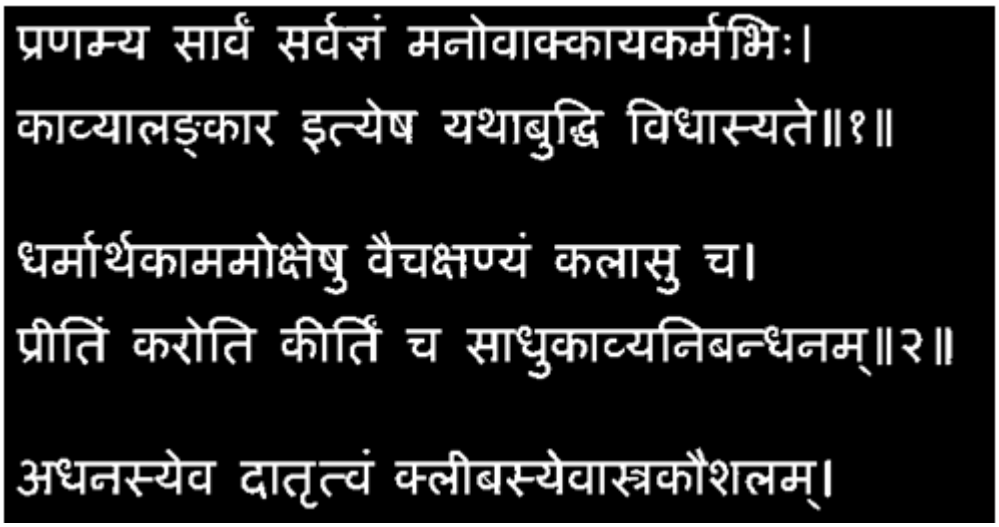
Saraojasaithataama yaa davaī sadh_amautaeṣau kaanaitaphapaedha

Saamsaithataa namasatasayaai namasatasayaai namasatasayaai namao namaa

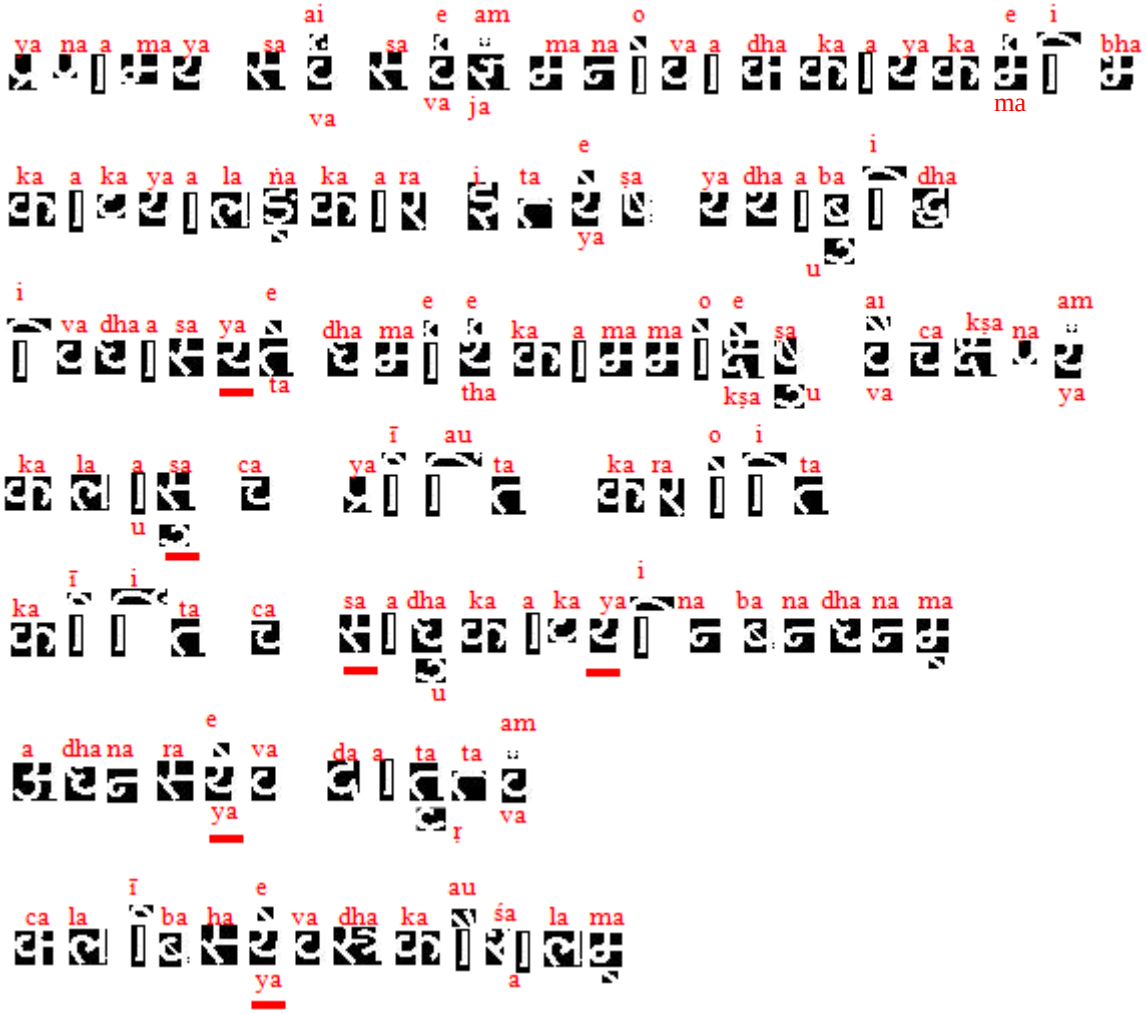
(c) Result of transliteration of Image 5

Figure 4.5: Results for Image 5

Results for Image 6:



(a) Binary image corresponding to Image 6



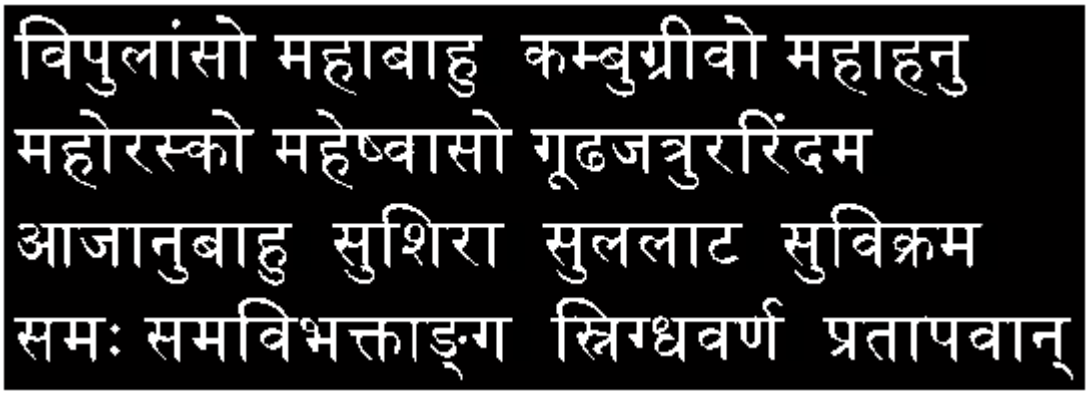
(b) Segmented characters with corresponding roman alphabets

yaṇamaya saaisavaejaam manaovaadhakaayakamaebhai
kaakayaalaṅakaara itayaeṣa yadhaabaudhai vaidhaasayatae
dhamaethaekaamamaokṣaesau vaaicakṣanayaam kalaasau ca
yaītaau karaotai kaītai ca saadhaukaakayanaibanadhanama
adhanarayaeva daataṛtavaam calaībahayaevadhakaauśalama_

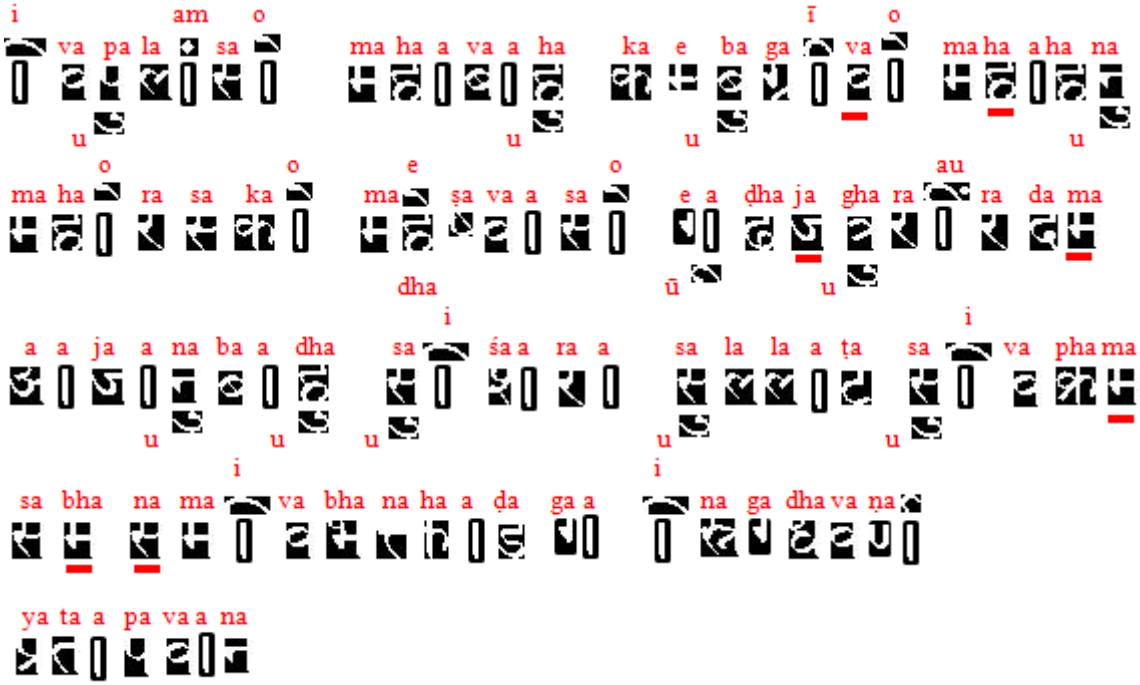
(c) Result of transliteration of Image 6

Figure 4.6: Results for Image 6

Results for Image 7:



(a) Binary image corresponding to Image 7



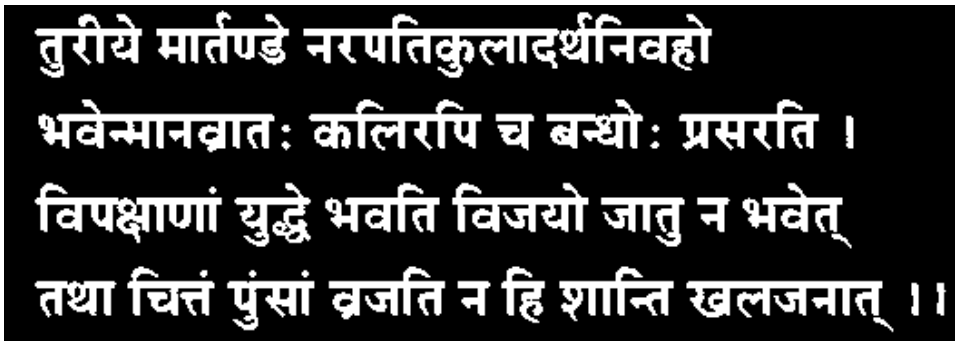
(b) Segmented characters with corresponding roman alphabets

Vaipaulaamsao mahaavaahau kaebaugaīvao mahaahanau
Mahaorasakao madhaeṣavaasao eaḍhajaghauraraaudama
Aajaanabaadha sauśaaraa saulalaaṭa sauvaiphama
sabhanamavaibhanahaaḍagaa naigadhavaṇa_ yataapaana

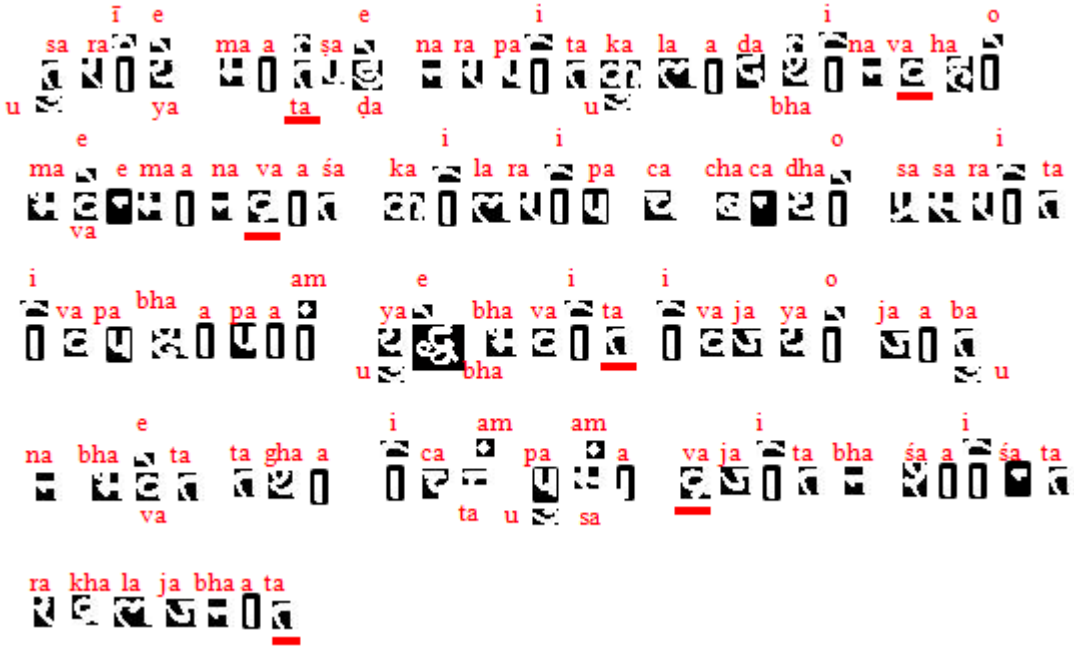
(c) Result of transliteration of Image 7

Figure 4.7: Results for Image 7

Results for Image 8:



(a) Binary image corresponding to Image 8



(b) Segmented characters with corresponding roman alphabets

Sauraīyae maata_ṣaḍae narapataikaulaadabha_naivahao

Mavaeemaanavaaśa kalairapai ca chacadhao sasaraita

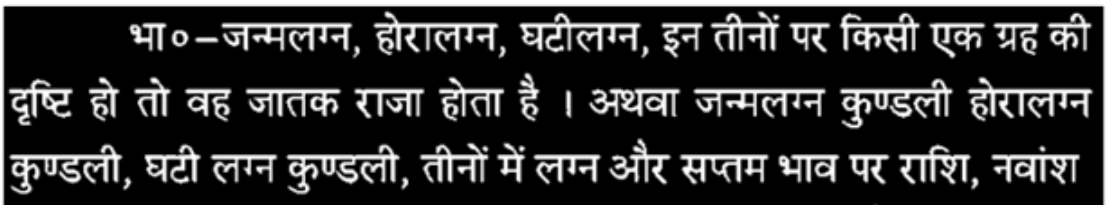
Vaipakṣaapaaam yaubhaebhavatai vaijayao jaabau na bhavaeta

Taghaa caitaam pausaama vajataibha śaaśata rakhalajabhaata

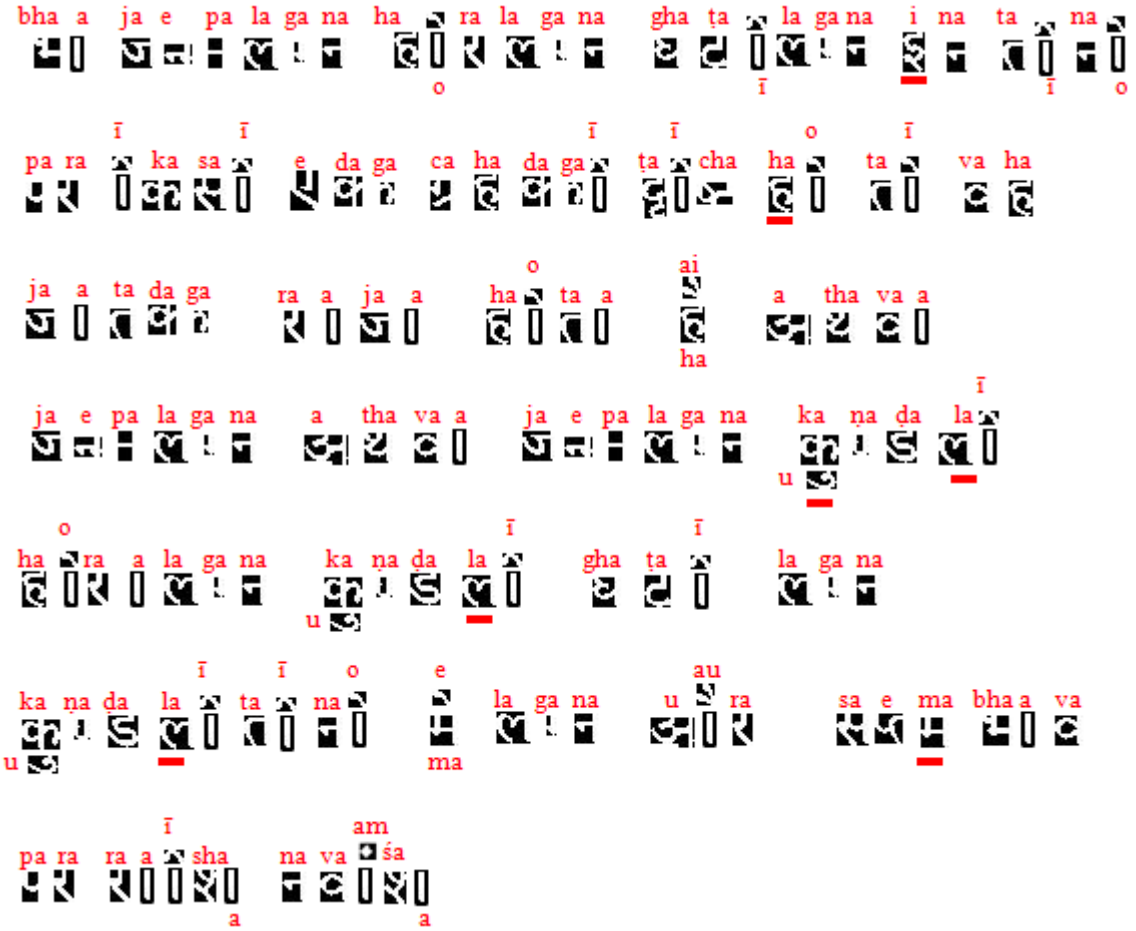
(c) Result of transliteration of Image 8

Figure 4.8: Results for Image 8

Results for Image 9:



(a) Binary image corresponding to Image 9



(b) Segmented characters with corresponding roman alphabets

Bhaa jaepalahana haoralagana ghaṭaīlagana ina taīnao para kaisaī edaga caha dagaī
Taīcha hao taī vaha jaatadaga raja haotaa hai athavaa jaepalagana kauṇadalaī haoraalagana
Kauṇadalaī ghaṭaī lagana kauṇadalaī taīnao mae lagana uaura saemabhaava para raaīshaa navaamśaa

(c) Result of transliteration of Image 9

Figure 4.9: Results for Image 9

4.2 Performance Analysis

To determine effectiveness of our algorithm, we have chosen to analyse results of character segmentation, recognition and transliteration process as the presented algorithm segments the lines and words in the document with 100% accuracy.

The proposed algorithm gives promising results for character segmentation of Devanagari document images and is able to perform preliminary segmentation with 100% accuracy. To evaluate efficiency of character segmentation process in the algorithm, we determine how well the presented algorithm can detect composite characters in an image and how precise are the segmentation cuts for such characters. In order to evaluate the performance of composite character segmentation we use following two parameters:

i) The **precision rate** is defined as ratio of correctly segmented composite characters to the sum of correctly segmented composite characters and false positives. **False positives** (FP) are those characters in the image that although are not composite characters but have been marked for further segmentation.

$$Precison\ rate = \frac{correctly\ segmented\ composite\ characters}{correctly\ segmented\ composite\ characters + FP} \text{ x } 100 \qquad (57)$$

ii) The **recall rate** is defined as ratio of correctly segmented composite characters to the sum of correctly segmented composite characters and false negatives. **False negatives** (FN) are those characters in the image that although are composite characters but have not been marked for further segmentation.

$$Recall\ rate = \frac{correctly\ segmented\ composite\ characters}{correctly\ segmented\ composite\ characters + FN} \text{ x } 100 \qquad (58)$$

Table 4.1 discusses the results of composite character segmentation. The data incorporated in the table has been obtained by executing character segmentation algorithm on several Devanagari documents. Out of total composite characters, proposed algorithm correctly detects about 85% of the characters and correctly segments 95% of detected composite characters. 99% of the composite characters that go undedicated are special cases of consonant combinations as shown in Figure 4.1. These special case conjuncts have same height and width as that of a regular core character.

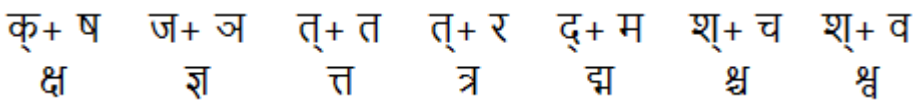


Figure 4.10: Special cases of conjunct combinations

Table 4.1: Performance of composite character segmentation

Image	Total composite char.	Accurately detected char.	Accurately segmented char.	FP	FN	P %	R %
1	10	9	8	1	1	88.89	88.89
2	9	7	7	0	2	100	77.78
3	7	5	5	1	2	83.33	71.42
4	18	17	17	1	1	94.44	94.44
5	10	10	10	0	0	100	100
6	14	11	11	0	3	100	78.57
7	6	3	3	0	3	100	50
8	5	3	3	1	2	75	60
9	5	4	2	4	0	33.33	100
Total	84	69 82.14%	66 95.56%	8	14	89.18	82.50

Table 4.2 evaluates the performance of character recognition and transliteration stage. The proposed method gives an overall accuracy of about 90% when some of the true characters are not first choice whereas this accuracy reduces to about 80% when all the true characters in transliterated text are first choice.

Table 4:2: Performance of Devanagari transliteration process

Image	Total char.	Total transliterated char.	Overall accurate transliteration	No. of Romanized char. that are not first choice	Error %
1	134	130	122	0	6.15
2	138	136	130	9	4.41
3	129	127	117	7	7.87
4	201	197	181	10	8.12
5	110	108	101	9	6.48
6	147	145	126	6	13.10
7	125	122	110	7	5.45
8	112	105	86	6	18.09
9	147	147	130	7	11.56
Total	1243	1217 97.90%	1103 90.63%	61 5.53%	9.36

CHAPTER -5

CONCLUSION AND FUTURE SCOPE

Most of the Indian literature such as Bhagavad Gita, Vedas, Mahabharata, and Ramayana is composed using Devanagari script. Such voluminous literature necessitates transliteration into roman script to make it more accessible to people who are unfamiliar with Devanagari. This thesis work attempts in Romanization of Devanagari document using character recognition with the help of underlying statistical and structural properties of character. The character recognition process interprets the document images and converts the text into editable format. Several techniques for character recognition have been proposed by researchers. A detailed study of these techniques has been carried out during the course of research and an effort has been made towards improving some of the previously defined techniques.

This dissertation proposes a complete character recognition system for Devanagari documents followed by transliteration of recognized Devanagari characters. This algorithm has been tested for several Devanagari documents with font size varying between 14-point to 36-point. Proposed method can efficiently segment skewed and overlapped lines using connected component analysis. The proposed connected component methodology for separation of overlapping lines quickens the process as compared to contouring the individual lines meanwhile keeping the individual lines intact. Proposed skew correction methodology works for full range of angles and is lot faster than traditional projection profile method cause of reduced skew angle variation range. We have profusely used structural properties for character segmentation process, discussing different possible scenarios like conjunct and shadow character segmentation. The described feature extraction methodology uses moment based features calculated for projection histogram and crossings together with zone based features. We have used two phase recognition scheme for classification procedure and discussed header line based errors encountered while recognition process. Recognition process is then followed by transliteration of Devanagari characters into corresponding roman alphabets based on phonetic similarities.

Performance of segmentation process is analysed using two commonly defined parameters precision rate and recall rate. Out of total composite characters, proposed algorithm correctly detects about 85% of the characters and correctly segments 95% of detected composite characters. 99% of the composite characters that go undedicated are special cases of consonant combinations. The algorithm gives a precision rate of about 89% and recall rate of about 80%. The presented transliteration technique transliterates 97% of the characters present in an image. The remaining 3% characters are usually modifiers. The algorithm gives about 90%

accuracy when true characters are not necessarily first choice. This accuracy reduces to 80-85% when only first choice of true characters is used.

The proposed method gives promising results however segmentation methodology is unable to detect consonant combinations that have similar height and width as that of basic core characters. Devanagari character set is already quite extensive when just basic vowels and consonants are considered. Adding any more consonant combinations will make the recognition system quite cumbersome. Another segmentation problem arises when character has rakar modifier because this combination also has similar dimensions as that of any basic vowel or consonant. Although in this dissertation we have considered just one combination of consonant and rakar modifier, our future effort will be to accommodate all such possible character combinations. Another area of improvement could be considering documents in noisy environments.

REFERENCES

[1] V. Bansal and R.M.K. Sinha, "Segmentation of touching and fused Devanagari characters", *Pattern Recognition*, Vol. 25, pp. 875-893, July 2002.

[2] M. L. Feng and Y. P. Tan, "Contrast adaptive binarization of low quality document images", *IEEE International Conference on Multimedia and Expo*, Vol. 1, pp. 339-342, June 2004.

[3] R.G. Casey and E. Lecolinet, "A Survey of Methods and Strategies in Character Segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, Issue 7, pp. 690-706, July 1996.

[4] O.D. Tier, A.K. Jain and T. Taxt, "Feature extraction methods for character recognition-A survey", *Pattern Recognition*, Vol. 29, Issue 4, pp. 641-662, April 1996.

[5] W. Niblack, *An Introduction to Digital Image Processing*, Prentice Hall, 1990.

[6] M. Sauvola and M. Pietikainen, "Adaptive Document Binarization", *Pattern Recognition*, Vol. 33, Issue 2, pp. 225-236, February 2000.

[7] C. Wolf and J.M. Jolion, "Extraction and Recognition of Artificial Text in Multimedia Document", *Pattern Analysis and Applications*, Vol. 6, Issue 4, pp. 309-326, February 2003.

[8] Y. Solihin and C. G. Leedham, "Integral Ratio: A New Class of Global Thresholding Techniques for Handwriting Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21, Issue 8, pp. 761-768, August 1999.

[9] T. Y. Zhang and C. Y. Suen, "A Fast Parallel Algorithm for Thinning Digital Patterns", *Communications of the ACM*, Vol. 27, Issue 3, March 1984.

[10] A. Papandreou and B. Gatos, "A Novel Skew Detection Technique Based on Vertical Projections", *Proceedings of International Conference on Document Analysis and Recognition*, pp. 384-388, September 2011.

[11] S. Li, Q. Shen and J. Sun, "Skew detection using wavelet decomposition and projection profile analysis", *Pattern Recognition Letters*, Vol. 28, Issue 5, pp. 555-562, April 2007

[12] B. V. Dhandra, V. S. Malemath, H. Mallikarjun and R. Hegadi, "Skew Detection in Binary Image Documents Based on Image Dilation and Region labelling Approach", *Proceedings of Eighteenth International Conference on Pattern Recognition*, Vol. 2, pp. 954-957, August 2006.

[13] M. Hanmandlu and O. V. R. Murthy, "Fuzzy model based recognition of handwritten numerals", *Pattern Recognition*, Vol. 40, Issue 6, pp. 1840-1854, June 2007.

[14] A.K. Das and B. Chanda, "A fast algorithm for skew detection of document images using morphology", *International Journal on Document Analysis and Recognition*, Vol. 4, Issue 2, pp. 109-114, December 2001.

[15] U. Pal and B. B. Chaudhuri, "Machine printed and handwritten text lines identification", *Pattern Recognition Letters*, Vol. 22, Issue 3, pp. 431-441, March 2001.

[16] U. Pal and S. Datta, "Segmentation of Bangla unconstrained handwritten text", *Proceedings of Seventh International Conference on Document Analysis and Recognition*, pp. 1128-1132, August 2003.

[17] N. K. Garg , L. Kaur and M. K. Jindal, "A new method for line segmentation of handwritten hindi text", *Proceedings of Seventh International Conference on Information Technology*, pp. 392-397, April 2010.

[18] G. Louloudis, B Gatos, L. Pratikakis and C. Halatsis, "Text line detection in handwritten documents", *Pattern Recognition*, Vol. 41, Issue 12, pp. 3758-3772, December 2008.

[19] A. Zahour, B. Taconet, P. Mercy and S. Ramdane "Arabic hand-written text-line extraction", *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pp. 281-285, September 2001.

[20] N. Priyanka, S. Pal and R. Mandal, "Line and Word Segmentation Approach for Printed Documents", *IJCA Special Issue on Recent Trends in Image Processing and Pattern Recognition*, Vol.1, pp. 30-36, 2010.

[21] U. Pal and P.P. Roy, "Multioriented and curved text lines extraction from Indian documents", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 34, Issue 4, pp. 1676-1684, August 2004.

[22] R. Manmatha and J. L. Rothfeder, "A scale space approach for automatically segmenting words from historical handwritten documents", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27 , Issue 8 , pp. 1212-1225, August 2005.

[23] S. Kompalli, S. Nayak, S. Setlur and V. Govindaraju, "Challenges in OCR of Devanagari Documents", *Proceedings of Eighth International Conference on Document Analysis and Recognition*, Vol. 1, pp. 327-331, September 2005.

[24] S. Kompalli, S. Setlur and V. Govindaraju, "Design and Comparison of Segmentation Driven and Recognition Driven Devanagari OCR", *Proceedings of Second International Conference on Document Image Analysis for Libraries,* pp. 98-102, April 2006.

[25] U. Garain and B. B. Chaudhuri, "Segmentation of touching characters in printed Devnagari and Bangla scripts using fuzzy multifactorial analysis", *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*", Vol. 32, Issue 4, pp. 449-559, February 2003.

[26] S. Kompalli, S. Setlur and V. Govindaraju, "Devanagari OCR using a recognition driven segmentation framework and stochastic language models", *Document Analysis and Recognition*, Vol. 12, Issue 2, pp. 123-138, July 2009.

[27] V. Bansal and R. M. K. Sinha, "A complete OCR for printed Hindi text in Devanagari script", *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pp. 800-804, September 2001.

[28] V. Bansal and R. M. K. Sinha, "Integrating knowledge sources in Devanagari text recognition system", *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, Vol. 30, Issue 4, pp. 500-505, July 2000.

[29] U. Pal , N. Sharma , T. Wakabayashi and F. Kimura, "Off-line handwritten character recognition of Devanagari script", *Proceedings of Ninth International Conference on Document Analysis and Recognition*, pp. 496-500, September 2007.

[30] T. Wakabayashi, U. Pal, F. Kimura and Y. Miyake, "F-ratio based weighted feature extraction for similar shape character recognition", *Proceedings of Tenth International Conference on Document Analysis and Recognition*, pp. 196–200, 2009.

[31] S. Arora , D. Bhatcharjee , M. Nasipuri and L. Malik, "A two stage classification approach for handwritten Devanagari characters", *Proceedings of International Conference on Computer Intelligence and Multimedia Application*, pp. 399-403, December 2007.

[32] V. Govindaraju, S. Kompalli, S. Setlur, S. Khedekar, F. Farooq and R. Vemulapati, "Tools for enabling digital access to multilingual indic documents", *Proceedings of the First International Workshop on Document Image Analysis for Libraries*, pp. 122–133, 2004.

[33] N. Sharma, U. Pal, F. Kimura and S. Pal, "Recognition of Offline Handwritten Devnagari Characters using Quadratic Classifier", *Computer Vision, Graphics and Image Processing*, Volume 4338, pp 805-816, December 2006.

[34] B.V. Dhandra, M. Hangarge and G. Mukarambi, "Spatial Features for Handwritten Kannada and English Character Recognition", *IJCA Special Issue on Recent Trends in Image Processing and Pattern Recognition*, Vol. 3, pp. 146-150, 2010.

[35] R. Jayadevan, S. R. Kolhe, P. M. Patil, and U. Pal, "Offline Recognition of Devanagari Script: A Survey", *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, Volume 41, No. 6, November 2011.

[36] M. Hanmandlu and O. V. R. Murthy, "Fuzzy model based recognition of handwritten numerals", *Pattern Recognition*, Vol. 40, pp. 1840-1854, June 2007.

[37] N. Arica and F.T. Yarman-Vural, "An overview of character recognition focused on off-line handwriting", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol. 31, Issue 2, pp. 216-233, August 2002.

[38] R. J. Ramteke and S. C. Mehrotra, "Feature extraction based on moment invariants for handwriting recognition", *Proceedings of IEEE Conference on Cybernetics and Intelligent Systems*, pp. 1-6, June 2006.

[39] R.M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, Addison-Wesley Longman Publishing, 1992.

[40] R.C. Gonzalez, R.E. Woods, *Digital Image Processing*, Pearson Education, 2011.

[41] A.K. Jain, Fundamentals of digital image processing, Prentice Hall, 1989.

[42] V. Govindaraju and S. Setlur, *Guide to OCR for Indic Scripts:Document Recognition and Retrieval*, Springer, 2009.

LIST OF PUBLICATIONS

J. Kaur and V. Kumar, “Devanagari transliteration: A complete character recognition and transliteration technique for Devanagari script”, Communicated to Optik - International Journal for Light and Electron Optics.

801463011

ORIGINALITY REPORT

3%	2%	2%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.mackenziegasproject.com Internet Source	<1%
2	newton.cs.concordia.ca Internet Source	<1%
3	Bukhari, Syed Saqib. "Generic Methods for Document Layout Analysis and Preprocessing", KLUEDO, 2012. Publication	<1%
4	Solihin, Yan Leedham, C.G.. "Integral ratio: a new class of global thresholding techniques for handwriting images.", IEEE Transactions on Pattern Analysis an, August 1999 Issue Publication	<1%
5	www.ciol.com Internet Source	<1%
6	Parrweejj, Fiirrojj. "An Empirical Evaluation of Off-line Arabic Handwriting And Printed Characters Recognition System", International Journal of Computer Science Issues (IJCSI), 2012. Publication	<1%

801463011

by Jasmine Kaur

FILE	JASMINE_801463011.PDF (4.71M)		
TIME SUBMITTED	13-JUL-2016 01:20PM	WORD COUNT	14478
SUBMISSION ID	689380875	CHARACTER COUNT	81908