Segmentation of Telugu Text using Fringe Maps

A thesis submitted during 2016 to the University of Hyderabad in partial fulfillment of the award of a Ph.D. degree in School of Computer and Information Sciences

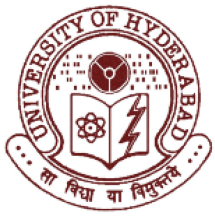
by

Koppula Vijaya Kumar

School of Computer and Information Sciences

University of Hyderabad
P.O. Central University, Gachibowli
Hyderabad – 500046
Telangana, India

September - 2016



CERTIFICATE

This is to certify that the thesis entitled **"Segmentation of Telugu Text using Fringe Maps"** submitted by **Koppula Vijaya Kumar** bearing **Reg. No. 04MCPC04** in partial fulfillment of the requirements for the award of **Doctor of Philosophy** in **Computer Science** is a bonafide work carried out by his under my supervision and guidance.

The thesis has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

(Dr. Atul Negi)
Supervisor
School of Computer and Information Sciences
University of Hyderabad
Hyderabad – 500046, India

Dean
School of Computer and Information Sciences
University of Hyderabad
Hyderabad – 500046, India

DECLARATION

I **Koppula Vijaya Kumar**, hereby declare that this thesis entitled **"Segmentation of Telugu Text using Fringe Maps"** submitted by me under the guidance and supervision of **Dr.Atul Negi** is a bonafide research work. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for the award of any degree or diploma.

Date :

Name: Koppula Vijaya Kumar

Signature of the Student:

Reg. No.: 04MCPC04

Acknowledgments

Firstly, I bow my head humbly before the Almighty God for making me capable of completing my Ph.D. Thesis, with his blessings only I have accomplished this huge task.

I am highly indebted to my supervisor Dr. Atul Negi without whose encouragement my work would not have attained the present form. He has been instrumental in inculcating the ideas. I sincerely acknowledge his inspiring guidance, affection, generosity and everlasting enthusiasm throughout the tenure of my research work..

I would also like to take the opportunity to express a deep sense of gratitude to my Doctoral Review Committee (DRC) members Dr. Chakravarthy Bhagvati and Dr. K.N.Murthy for their cordial support and constant encouragement during the period of research. I am grateful to Dr. C. Raghavendra Rao for the timely help and support in lien period of my supervisor.

I thank the Administration and the Management of CMR College of Engineering and Technology Hyderabad, in particular Sri Ch.Gopal Reddy, Secretary for his support.

I would like to extend my heartfelt thanks to my better half (Smt. K. Madhavi) for her invaluable support at home, without which I would have not accomplished the work.

Last but not the least, I thank my beloved kids (Dikshitha & Nihal Preetham) for sparing me from spending considerable time in their leisure with them.

My heartfelt appreciation and gratitude to all the people who helped me in completion of this doctoral work.

Abstract

Text segmentation is the process of dividing a document image into text lines, words and characters. It is one of the important steps in document image analysis (OCR systems). Performance of OCR systems depends very much on the accuracy of segmentation. Segmentation of Telugu text is a difficult task due to its complex orthography. From the literature various text segmentation approaches are seen, like projection profiles, piece-wise projections, RLSA, adaptive RLSA, CC based methods, background based method, etc. These methods produce erroneous segmentation on general Telugu documents because of close vertical spacing between the ascenders and descenders from vowel modifiers and consonant modifiers and touching of components. By observation of these approaches we abstract a generalized working principle of text segmentation which we categorize as text affinity and space affinity. We propose a Fringe map based holistic approach for text segmentation which attempts to balance both text and space affinity to produce good segmentation results. We study in detail Fringe maps, which are a kind of distance transform. Different operations upon them and properties of Fringe maps which are useful for text segmentation are shown. Operations like thresholding and projection profiles of fringe maps are presented. An important property of fringe maps that is peak fringe numbers (PFN) is the basis of the proposed approaches for segmentation. We define a PFN as the largest fringe value enclosed between two black pixels in the direction of interest. PFNs are used to find text affinity and space affinity. We propose sophisticated text line segmentation methods based on text affinity and space affinity using PFNs.

Proposed methods are tested on the corpus of 1025 pages data sets which are collected from Robust OCR project, University of Hyderabad and Digital library of India (DLI). A pixel level ground truth data sets was prepared for performance evaluation of proposed line segmentation methods. The best performance of the method shows 99.82% correct segmentation. Proposed text line segmentation method was experimentally shown to be versatile and tested on different Indic scripts and handwritten data sets collected from ICDAR 2013 Handwritten Segmentation Contest. 99.09% segmentation accuracy on Indic scripts and 91.1% segmentation accuracy on

handwritten documents was obtained. Fringe maps are extended to find word segmentation, character segmentation and baseline detection for Telugu text. Proposed methods produced good results when applied upon different Indic, Roman scripts and handwritten documents.

Contents

List of Figures

List of Tables

Abbreviations

AD	:AD is average distance between CG(TLS) and MAP(TLS)
AH	:Average height of CCs in a page image
APFN(TLS)	:Average PFN value of Text line segment
ARLSA	:Adaptive Run Length Smearing Algorithm
Bot(CC)	:Bottom of the CC in the MBR of CC
CC	:Connected Component
CG(CC)	:Centre of gravity of CC
CG(TLS)	:Centre of gravity of text line segment
DLI	:Digital library of India
DR	:Detection rate
DT	:Distance transform
FM	:A final performance metric F-Measure
H(CC)	:Height of a CC
HPP	:Horizontal Projection Profile
ICDAR	:International Conference on Document Analysis and Recognition
IPFN	:Internal peak fringe numbers
Left(CC)	:Left of the CC in the MBR of CC
MAP(TLS)	:Moving average of PFNs of Text line segment
MBR	:Minimum bounding rectangle
MCIT	:Ministry of Communication and Information Technology
MPSL	:Statistical mode of PSL
OCR	:Optical Character Recognition
OH	:Over height connect component
PFN	:Peak Fringe Number
PP	:Projection Profiles
PSL	:Piece-wise Separating Lines
RA	:Recognition Accuracy
Right(CC)	:Right of a CC in the MBR of CC
RLSA	:Run Length Smearing Algorithm
ROI	:Region of Influence
SVM	:Support vector Machine
TDIL	:Technology Development for Indian Languages
TLR(CC)	:Text line region of a CC
TLROI	:Region of Influence for text line segmentation
Top(CC)	:Top of the CC in the MBR of CC
UH	:Under height connect component
VOV(CCi, CCj)	:Vertical overlapping of connected components CCi and CCj
VPP	:Vertical Projection Profile

Chapter 1

Introduction

This chapter introduces the topic of text segmentation in Telugu OCR systems. We discuss the significance of OCR system and state of the art of text segmentation in Indic OCR systems. Telugu language is one of the most popular south Indian language. It has a complex orthography. We present the Telugu character representation and issues of text segmentation in Telugu documents. Finally we concluded this chapter with the contents of this thesis and contributions.

1.1 Text Segmentation and OCR Systems

Since the dawn of the printing technology revolution the written word has expanded the ease with which knowledge is shared. With progress in digital technology print has also an expanded scope. While currently many documents are being born digital, vast treasures of literature in libraries exist that are in print and handwritten formats. These documents are fragile and inaccessible to the public. In the recent years, OCR has facilitated availability of numerous books online, scholars can access and share knowledge easily with OCR technology. OCR has revolutionized the document management process, and can be widely used throughout entire range of industries or business. An OCR system converts scanned documents from image files into computer editable documents in encoded text format and it is searchable with text content. People do not require to manually retype important documents when

storing them into electronic databases. The result is accurate and efficient information processing in less amount of time.

Generally, an OCR system [18, 36, 43] may be said to have four stages of processing. 1. Preprocessing 2. Segmentation 3. Recognition and 4. Post processing as shown in Fig 1.1. In preprocessing step, the tasks are mostly binarization of scanned image, noise removal and skew correction. Segmentation is a process of dividing the document image into lines, words and characters or potential recognition units. The output of segmentation step is sent to the recognition system. The data generated by the recognition system is analyzed in post processing stage for syntax and semantic checks. The efficacy of an OCR system is very much influenced by the output of segmentation process.

In the last three decades extensive research has taken place in the field of Document

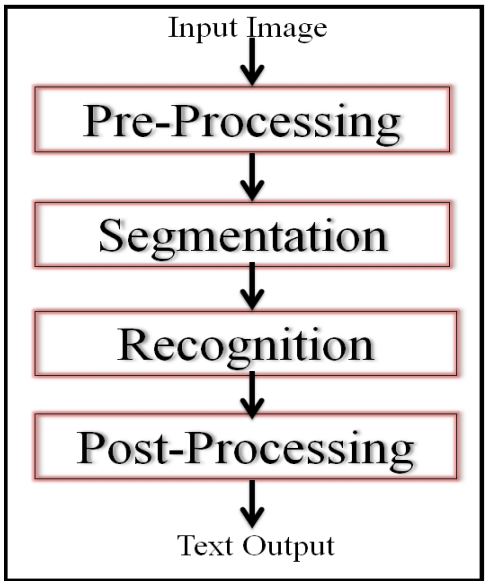


Figure 1.1: OCR system

image analysis [36, 43]. Most of the research is on printed and handwritten documents of Roman scripts. OCR for printed documents of Roman scripts is usually seen as a practically solved problem but this is not the case for Indic scripts. Efforts are on the way for the development of efficient OCR systems for Indian languages, especially for Telugu, a popular south Indian language.

1.1.1 Indic OCR Systems

OCR for Indian languages [61] in general is more difficult than for Roman scripts because of the large number of vowels, consonants, and conjuncts (combination of vowels and consonants). Further, most Indic scripts are spread over several zones. Segmentation has to deal with the positioning of the conjuncts and half syllables. These factors coupled with the inflectional and agglutinative nature of Indian languages make the OCR task quite challenging [18]. TDIL program was initiated by MCIT, Govt. of India for development of robust document analysis and recognition system for Indian languages. Research work of B B Chaudhari and U Pal [12] is very significant in bringing accuracy and performance to OCR systems for Indic scripts. They reported a complete OCR system for Bangla. Not only for Bangla they worked for developing OCR system for Oriya script [13,63]. Lehal presented an OCR system for printed Gurumukhi script [27,28]. A G Ramakrishnan [2,23] worked for developing OCR systems for Tamil and Kannada. A Negi, C Bhagvati, CV Lakshmi and CV Jawahar [6, 16, 26, 39] reported work on OCR systems for Telugu. While reviewing the research work on Telugu OCR systems [6, 16, 25, 26, 39, 50] by analysis we found that most of the work is on recognition. Given the very high accuracy classifiers like SVM etc. it was observed that the performance of OCR systems could be improved by minimizing the errors of segmentation phase.

Text segmentation in printed Telugu documents is quite challenging due to linguistic complexity. The script with its ascenders and descenders leads to frequent instances of descenders from previous line "Touching" the ascenders of the present line. Similarly while at times the close line spacing leads to an "overlapping" of the text in the projection profile [6,24]. Touching and overlapping occurs very frequently in Telugu documents [31]. In addition to this, most of the documents (For example as seen in the DLI www.dli.ernet.in) are non uniform print quality and primitive typesetting. Along with these factors coupling of text skew and document skew it

makes the text line segmentation task very complex. Word segmentation in Telugu is not simple because the character size and inter character gap differ from word to word, even within the same document page. Character extraction also more complex than in other scripts [22]. Consonant modifiers are placed either side of the base character (i.e between two adjacent base characters/Consonant) and assignment of these to appropriate base character is complex. Some times we may not find simple separation between characters then finding segmentation path between characters is difficult.

We did a comprehensive study on these problems and proposed a text and space affinity based algorithms using fringe maps in this thesis.

1.2 Telugu Script and its Representation

Telugu is a phonetic language spoken by more than 100 million people in south India. Telugu script is written from left to right, with each character representing a syllable. Telugu script has 16 vowels (V), which are called Achchulu and 36 consonants (C), which are called Hallulu. These are shown in Fig. 1.2 and Fig. 1.3 respectively. From the figure, it can be imagined that all of them are composed of circular segments with different radii.

Figure 1.2: Vowels of Telugu script (*Achchulu*).

Telugu script has a special symbol set called as *Maatra*, while corresponding to each vowel, a modifier is attached to *hallus* to modify their sound. In general we call it as vowel modifiers (VM). *Maatras* corresponding to each vowel in Fig. 1.2 are

Figure 1.3: Consonants of Telugu script (*Hallulu*).

shown in Fig. 1.4. In Fig. 1.5 each maatra has been attached to the first *hallu*. These sequence of characters obtained by adding *maatra* to *hallus*, are called *Guninthalu* in Telugu. From the figure it can be seen that some *maatras* are placed at the top, some at bottom right and some at bottom part of the consonant. Same *maatra* can be attached at different positions for different *hallus* and same *maatras* can occur in different shapes depending on the *hallu* to which it is attached.

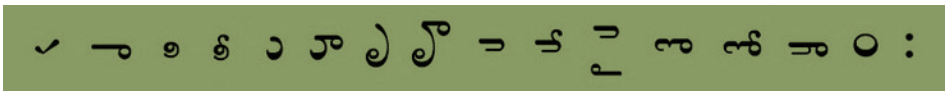


Figure 1.4: *Maatras* corresponding to each vowel

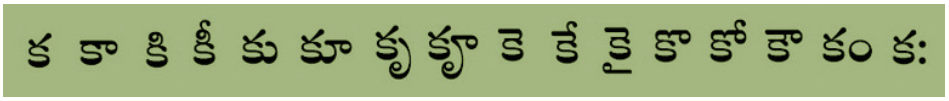


Figure 1.5: *Maatras* corresponding to each vowel attached to first *hallu*

Besides all of these, there are 'half consonants' called *Vottus*, since corresponding

to each consonant a vowel sound completes it. However the *vottus* are placed at a distance at the bottom or right corner of *Hallulu*. In general vottu is called as consonant modifier (CM). The *Vottus* for the respective *Hallulu* in Fig. 1.2 are shown in Fig. 1.6

Figure 1.6: The *vottus* for the respective *Hallulu*

A character (or Akshara) is said to be simple if it is an *achchu* or a *hallu* alone with a *maatra* (first two characters V, C in Fig. 1.7). A character is said to be compound if it is a *hallu* with a *maatra* and with any number of *vottus* i.e. in regular expression (C.VM)(CM)* (last two characters in Fig. 1.7). In practice, characters with more than two *vottus* are very rare.

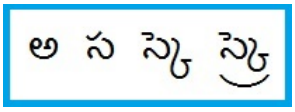


Figure 1.7: Example of simple and complex characters in Telugu Script

All the *achchus*, *hallus*, *maatras* and *vottus* together roughly provide 130 basic or-

thographic units, which are referred as glyphs that are combined together in different ways to represent all the frequently used syllables.

1.3 Motivation

The first comprehensive OCR system for Telugu is DRISHTI [6, 39]. It was designed for document images with simple layouts scanned at 300 dpi. However later versions [25] are part of project working towards a robust OCR system for various Indic scripts. In DRISHTI, word and glyph separation are the key steps. Word segmentation is done using a combination of Run-Length Smearing Algorithm (RLSA) and Connected-Component Labelling. Words are combined into lines using simple heuristics based on their locations. The performance of RLSA in accurately segmenting words is very high on good quality text but drops in the presence of complex layouts. In DRISHTI, text line is represented as bounding box of combined words. In the bounding box concept, the overlapping components of adjacent lines are truncated and passed to the recognition engine as separate components. This affects the performance of OCR system. C V Lakshmi et al [26] segmented Telugu text lines using projection profiles. However, Telugu text shows a projection profile with varying inter-peak and inter-valley distances. More than one peak is usually present within a single row of text. In Telugu scripts overlapping and touching of components occurs most frequently. These factors make the projection profiles prone to errors in identifying lines [6].

Character (or Akshara) segmentation is not presently in DRISHTI. Individual components are send to the recognition. After the recognition it combines the components into a character. Recognition errors causes misclassification of characters and wrong combination of components. It is clear that a lot depends upon the success of research for segmentation in Telugu script.

1.4 Segmentation Issues in Telugu Script

Segmentation of printed Telugu documents is quite challenging due to orthographic complexity. As discussed in the above section scripting a Telugu character is 'combination of consonant with consonant modifiers and vowel modifiers'. Text segmentation method makes errors because the consonant modifiers and vowel modifiers occupy the region of adjacent characters in vertical and horizontal direction. This we discuss in detailed in next sub-sections. A common source of digitized Telugu documents are in DLI. These documents are non uniform print quality and primitive typesetting. Because of this text lines are in different skew direction and skew within the text line. In addition to this, document skew and variation in space makes more complex the segmentation process.

1.4.1 Issues in Text line Segmentation of Printed Telugu

In the following we shall attempt to bring together the issues which are encountered when designing schemes for line segmentation. Segmentation of printed Telugu text into lines is a complex and challenging task due to variety of different situations. In particular difficulty arises with the situations:
1) Components of a text line is touching with components of neighboring text line.
2) No linear space between neighboring text lines (components of neighboring text are within the MBR of a text line)
3) Large variation of spaces between text lines
4) Text with curvilinear lines
5) Text with variation in the skew directions, etc.
Touching and overlapping of a text line occurs most frequently in Telugu documents [31]. Because the consonant modifiers and vowel modifiers of a text line occupies neighboring text line regions, text lines with overlapping line MBR's as shown in Fig. 1.8.

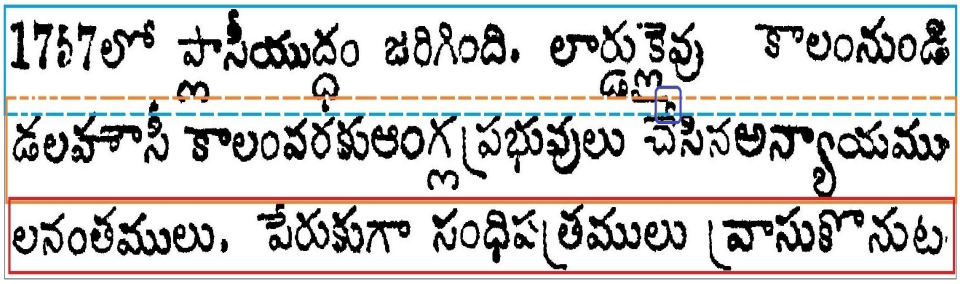


Figure 1.8: Telugu text with line MBRs, Overlapping line MBR region is in dotted lines and touching component is shown in rectangular box.

Most of the Telugu poetry documents contains space variations between text lines as shown in Fig. 1.9. In some algorithms, inter-line spacing is estimated from the document image and is used for line segmentation. However variations in spacing within a page result in split and merge errors. Telugu documents present in DLI are primitive typesetting and non uniform print quality, most of the documents are 18^{th} and 19^{th} centuries. Because of primitive typesetting and non uniform print quality, text lines are in different skew and curvilinear as shown in Fig. 1.10. In skewed documents line MBRs may overlap and their is possibility of touching of components. All these makes Telugu text line segmentation process more difficult.

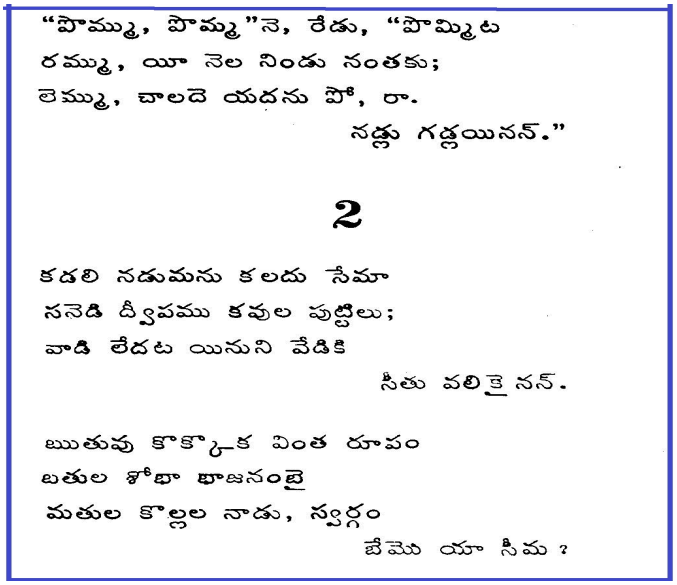


Figure 1.9: Telugu poetry documents with variation of space between text

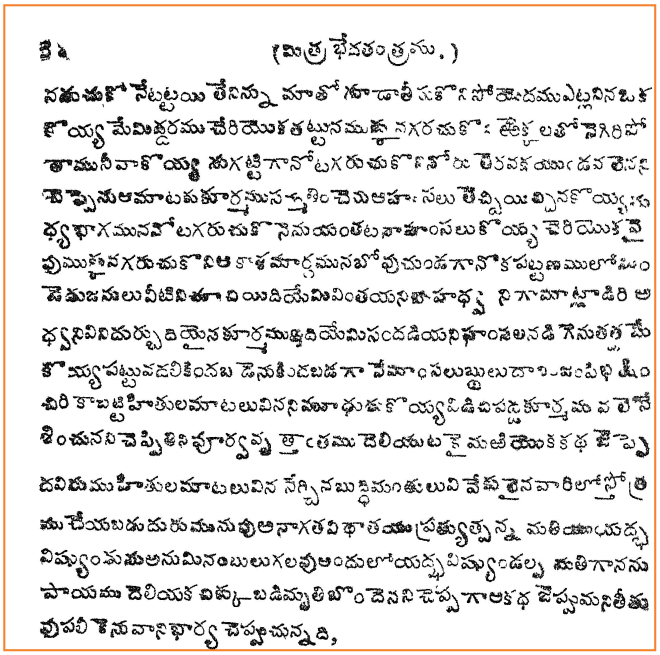


Figure 1.10: Telugu text skew

1.4.2 Issues in Word Segmentation of Printed Telugu Script

Word segmentation is a process of dividing input line image into word images. Word segmentation algorithms use the gap between the characters and character size for segmentation. Word segmentation in Telugu documents is complex task because the character size and inter character gap differ from word to word, even within the same document page. Fig. 1.11 shows the variation in gaps between words. and between characters.

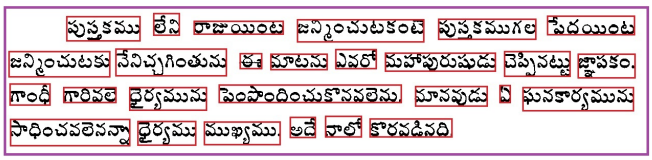


Figure 1.11: Word gaps and Character size variation

1.4.3 Issues in Character Segmentation of Printed Telugu Script

Character segmentation in Telugu is more complex than in other scripts. In European languages we may find distinct linear space between character. In Telugu most of the cases we may not find distinct linear space between characters because consonant modifiers are placed between two adjacent base characters. Identifying the components of a character is also difficult because some components staged vertically with two or more base characters as shown in Fig. 1.12.

Figure 1.12: Components of characters

1.5 Problem Statement

The objective of research is to design and develop analysis methods for segmentation of printed Telugu scripts. The aim main to segment the input binary text image into lines, words and Aksharas. The input page image is passed through the text line segmentation method for generating line images. Line images are divided into words. Akshara segmentation takes the word image and separate aksharas and identifies the regions of potential recognition units (base consonant, vowel,etc.).

1.6 Thesis Organization and Contributions

In this section we discuss the organization of thesis and the main contributions of this thesis. In chapter 2 we did a literature survey on text segmentation approaches.

The approaches like projection profiles, piece-wise projection, RLSA, Adaptive RLSA, CC based methods, White stream based method, voronoi-diagram based method and background thinning based methods are described. Failures and errors of these methods when applied on Telugu documents are discussed. From these approaches we derive a novel concept text affinity, space affinity and other attributes which are used for text segmentation. We proposed a method based on text affinity using connected components. This work is published in *Proceeding of ICETET-09*. Text segmentation approaches can produce high accuracy by using both text and space affinity. Chapter 3 give a conjunctive approach of both text affinity and space affinity based method using Fringe Maps. Fringe maps are a kind of distance transformation. Distance transform and fringe maps computation is described. Fringe maps are used in OCR for recognition of characters. We proposed operations on fringe maps like thresholding, projections, peak fringe numbers and region of influence of fringe numbers. These operations are used for segmentation of text. In chapter 4 we proposed a affinity based method for text line segmentation of Telugu text using PFNs. This method is tested on Telugu corpus collected from Robust OCR project at University of Hyderabad and DLI. The proposed method performance is evaluated quantitatively and qualitatively by comparing with other methods. This work is published in *Proceeding of ICDAR-2011 and VCON-2010*. Chapter 5 is extended the PFNs usage for word segmentation, baseline detection and character segmentation of Telugu text. word segmentation and baseline detection methods are generic and script independent. Hence we tested these methods on different scripts. Chapter 6 describes the proposed text line segmentation method in chapter 4 is script independent and robust. To prove the robustness of the method we tested this method on different Indic script collected from DLI and Handwritten documents from ICDAR 2013 handwriting contest. Finally we concluded in chapter 7.

Chapter 2

An Overview of Text Segmentation Approaches Based on Affinity Concept

There are intuitive notions behind the way text segmentation is done. In this chapter we review the essential principles behind text segmentation and see their performance. The most well known text segmentation approaches are the following: Projection profiles based approaches [11, 26, 31, 57], Piece-wise projection profile based approaches [45, 63], Run Length Smoothing Algorithm (RLSA) based approaches [11, 39, 40], Document spectrum [11, 42], Whitespace analysis [31], Hough transform [32] and Voronoi tessellation [57]. We observe that in Indic OCR systems for segmentation projection profiles, piece-wise projection, RLSA and connected component (CC) based methods are mainly used. After reviewing the existing approaches the fundamental notions of text and space affinity are proposed.

Algorithms have been called as top-down, where an entire page is segmented and further sub-segmented. Other so called bottom-up approaches start by collecting connected components and then aggregating them to larger units like words, and lines,..etc.

2.1 Projection Profile Based Methods

Projection-profile based methods are one of the earliest and also most popular top-down algorithms for printed and handwritten documents. The projection profile is obtained by summing the count of black (printed) pixel values along the coordinate axis for each value in the perpendicular axis. For example, projection profile in horizontal direction is obtained by summing black (printed) pixel values along the horizontal axis for each y value. *Profile(y)*=$\sum_{x=1}^{N} I(x, y)$ Where I is the image and N is width of Image. The most popular approach for detecting lines is based on Projection profiles. Peaks and valleys of profile are used for segmenting lines. Nagy [37], Srihari [59, 60] and Baird [3] used projection profiles for segmenting a document page into various units. Ha et al [15] used projection profiles of bounding box of connected components for segmentation and classification of page image into words, lines and paragraphs. However, this cannot be directly used unless white space between two neighboring text lines is distinct.The top-down approaches also have the disadvantage that they cannot process complex non-Manhattan layouts. C V Lakshmi [26] segmented Telugu text lines using projection profiles. However, Telugu text shows a projection profile with varying inter-peak and inter-valley distances. More than one peak is usually present within a single row of text. These factors make the segmentation method prone to errors. Figure 2.1 shows Telugu text and its projection profile. Bhagvati et al [6] made observations on the nature of the projection profiles of Telugu being different from other Indic scripts.

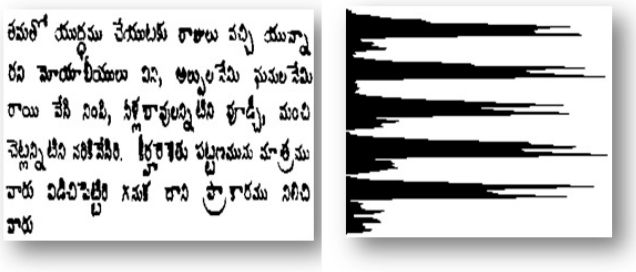


Figure 2.1: Telugu text and its projection profile

2.2 'Piece-wise' Projection Method

The concept of projection profiles fails if there are even mild overlaps between ascending parts of the script and the descenders from the previous line. The determination of the line gap becomes difficult. This happens because its difficult to disambiguate the horizontal context (position) of a gap when given a summary statistic of the projection (accumulation of full row of pixels). Piece-wise projection method [45, 63] divides the text into vertical stripes which reduces the horizontal context to only a single stripe. In each stripe, Piece-wise Separating Lines (PSL) are computed. In a given stripe, for all rows, the projection of black pixels is computed. The row where this sum is zero is marked. Few consecutive rows may get sum of all black pixels as zero. Then the first row of such consecutive rows is the PSL. The PSLs of different stripes of a text are shown by black horizontal lines in Fig. 2.2. All these PSLs may not be useful for line segmentation. Some potential PSLs are chosen from these. The normal distances between two consecutive PSLs in each stripe is calculated. The statistical mode (MPSL) of distances is take as a threshold to choose potential PSLs. If the distance between any two consecutive PSLs of a stripe is less than MPSL, then upper PSL is removed. PSLs obtained after this removal are the potential PSLs. The potential PSLs of Fig. 2.2 is shown in Fig. 2.3. The left and right co-ordinates of each PSL are saved for future use. By proper joining of these potential PSLs one may obtain text lines. It may be noted that sometimes because

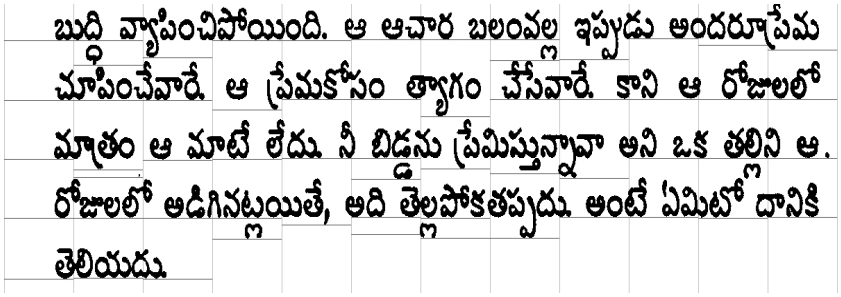


Figure 2.2: PSL in each stripe

of overlapping or touching of one component of upper line with a component of lower line, PSLs may not be found in some regions. We take care of these situations by

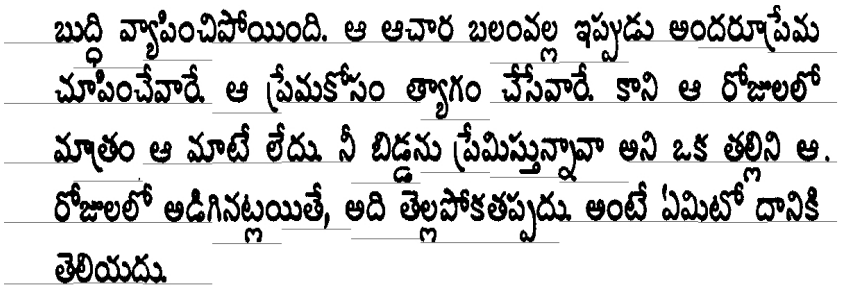


Figure 2.3: Potential PSL in each stripe

joining PSLs. Joining of PSLs is done in two steps. In the first step PSLs are joined from right to left and in the second step we check whether PSL joining in each line is spanning the width of the document image. We explain the process of joining PSLs to obtain segmenting lines.

To join a PSL of ith stripe, say K_i, to a PSL of *(i-1)*th stripe check any PSL, whose distance from K_i is less than MPSL in *(i-1)*th stripe. If a PSL exists, join left co-ordinate of K_i with right co-ordinate of the PSL in *(i-1)*th stripe. If PSL does not exist, extend PSL K_i in left direction horizontally until PSL reaches left boundary of the *(i-1)*th stripe. While extending if the PSL intersects a black pixel of a component of the *(i-1)*th stripe then the decision is that of whether this to be made a member of component either the upper line or lower line. Membership of a component is decided based on the distances from the point of intersection to the top and bottom point of the component. If the distances from the point of intersection to the top and bottom of the component is greater than half of the MPSL, it is assumed as a touching case. PSL is extended from point of intersection to either side of the component. In other case the PSL is extended along the nearest lower or upper contour of the component. Line segmentation results are shown in Fig. 2.4.

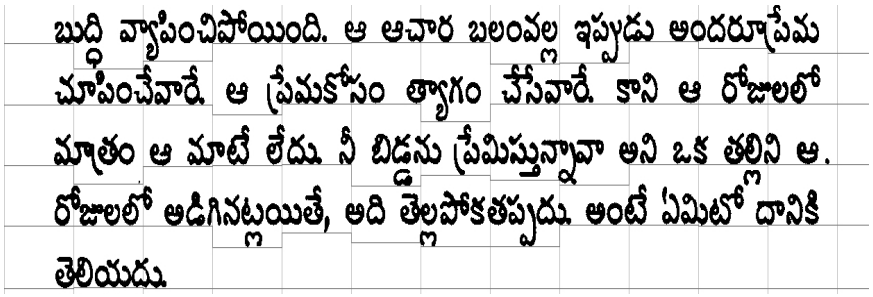


Figure 2.4: Line segmentation by piece-wise projection.

Pal and Datta, Tripathy and Pal, also Papavassiliou [45, 46, 63] used piece-wise projection method in text line segmentation. These methods cause errors while segmenting text lines from Telugu and complex documents. In Telugu, the peak-valley points are generated above and below the consonant modifiers, this mislead the segmentation path. Especially the methods [45, 63] use mode value to filter potential PSLs. However due to variations in space between lines, the use of the mode value filters out necessary PSLs which causes segmentation errors. Another important experimental parameter is the determination of the width of the stripe. Results are poorer for wider stripes, and ideally the stripe width should be near to the average width of a word's bounding box.

2.3 Run Length Smearing Algorithm

The Run Length Smearing Algorithm (RLSA) [39,64] is applied on binary images. The main aim of RLSA is to fill up the white pixel runs being in the vertical or horizontal directions. Given a direction (horizontal or vertical), RLSA eliminates white runs in rows of pixels whose lengths are smaller than a threshold smoothing value. In this method, continuous black pixels along the horizontal direction are smeared. If the distance between the black pixels is within a predefined threshold then the white pixels are filled with black pixels. If the threshold for smearing is taken larger than the inter word space then the bounding boxes of the connected components in the smeared image are considered as text lines. The horizontal smearing alone is not

sufficient to segment text lines in Telugu script, vertical smearing also is necessary to merge the consonant and vowel modifiers of a character. But vertical smearing may join conjuncts, vowel and consonant modifier of adjacent lines and results in merged text lines as shown in Fig. 2.5. RLSA was used in the Telugu OCR system DRISHTI [39]

Figure 2.5: Telugu text (left) and Run length smearing (right). Notice that text lines are merged because of consonant modifiers (shown in circle).

2.4 Adaptive RLSA

Adaptive Run Length Smoothing Algorithm (ARLSA) [40] is a revised version of the RLSA in horizontal direction. The drawbacks of the RLSA algorithm, such as grouping inhomogeneous components or different slanted lines are overcome in the ARLSA. Connected component analysis is necessary before applying ARLSA. Two types of background pixels (white) sequences are considered. The first type white pixel sequences which occur between two foreground pixels (black) which belong to the same connected component as shown in Fig. 2.6. In this case, all background pixels of the sequence are replaced with foreground pixels. The second type of background pixels sequence occurs between two different connected components in Fig. 2.7. In this case, constraints are set in regard to the geometrical properties of the connected components and the replacement with foreground pixels is performed when these constraints are satisfied.

We describe the ARLSA approach with the following. Let CC_i and CC_j be two connected components and $S(i, j)$ a horizontal sequence of background pixels between

Figure 2.6: Background sequence belonging to same connected component.

Figure 2.7: Background sequence belonging to different connected components.

CC_i and CC_j.
Four metrics are defined:

- $L(S)$: length of the sequence $S(i,j)$, that is the number of white pixels.
- $H_R(S)$: height ratio between CC_i and CC_j, which is defined as follows:
  $H_R(S) = \frac{max(h_i,h_j)}{min(h_i,h_j)}$ where h_i, h_j the heights of CC_i and CC_j, respectively.
- $O_H(S)$: the horizontal overlapping between the bounding boxes of CC_i and CC_j, which is defined by the following equation:
  $O_H(S) = max(Yl_i, Yl_j) - min(Yr_i, Yr_j)$
  where $\{Xl_i$,$Yl_i\}$ and $\{Xr_i$,$Yr_i\}$ are the coordinates of the upper left and bottom right corner of the CC_i's bounding box. The horizontal overlapping $O_H(S)$ is graphically demonstrated in Fig. 2.8. and it can be observed that when $O_H(S) < 0$, horizontal overlapping exists between the two connected components CC_i and CC_j.
- $N(S)$: a binary output function.
  $N(S)$ is set to 0 when a third connected component CC_k, $k \neq i,j$ exists in the 3×3 neighborhood of at least one pixel of the sequence $S(i,j)$. Otherwise,

$N(S)$ is set to 1. As shown in the example of Fig. 2.9, in the 3×3 neighborhood of the gray pixel sequences, no object other than CC_i and CC_j exists.

Based on the above metrics, a sequence of background pixels is replaced with foreground pixels only if: $(L(S) \leq T_l) \wedge (H_R(S) \leq T_h) \wedge (O_H(S) \geq T_o) \wedge (N(S) = 1)$ where T_l, T_h, T_o are predefined threshold values. The length threshold T_l of each sequence is related to the heights of the connected components. If h_i and h_j express the heights of CC_i and CC_j then $T_l = a \times min\{h_i, h_j\}$ where a is a constant value. Threshold T_h was set to 3.5 based on the following assumption: consider that between a lowercase character such as 'o' and a character with descender or ascender of the same font size, a height ratio between 2 and 3 is very common. Taking into account the fact that in historical documents the font size varies to a value 3.5 for T_h which gives enough tolerance against font size variation between characters of the same text line [40]. The horizontal overlapping threshold T_o is expressed as the percentage of overlap in regard to the component with the smallest height, that is: $T_o = c \times min\{h_i, h_j\}$ where c is set to 0.4. This means that at least 40% of the shortest component height must be covered in order for a link to be established. The constraint based on the function $N(S)$, ensures that background pixels will be transformed into foreground pixels, only if a third component pixel does not exist in the 3×3 neighborhood as shown in Fig. 2.9. Its purpose is to prevent the creation of false links between objects and therefore the integration into component groups of unwanted objects. It is very helpful in historical and degraded documents where text line spacing is narrow and characters from different text lines overlap.

2.5 Connected component Based Methods

Connected components based methods [29,34,42] aggregate neighboring connected components using heuristics based on the relationship between the neighboring components. Li et al. [29] discuss the text line detection task as an image segmentation problem. They use a Gaussian window to convert a binary image into a smooth gray-

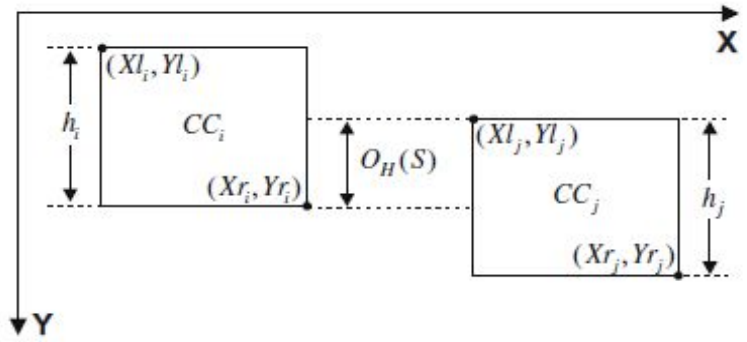


Figure 2.8: Horizontal overlapping of connected components.

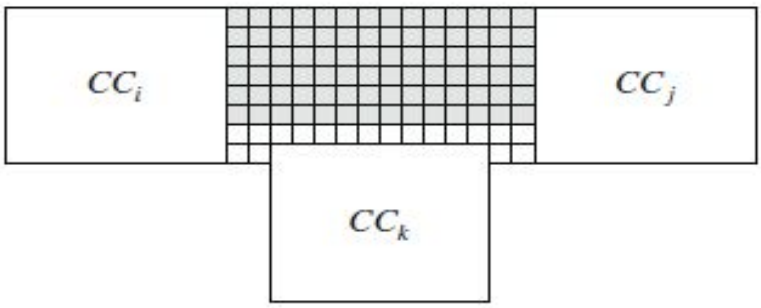


Figure 2.9: The constraint based on N(S) function. Gray pixels represent the background pixels which will be replaced with foreground pixels

scale image. Then they adopt the level set method to evolve text line boundaries and finally, geometrical constrains are imposed to group CCs or segments as text lines. They report pixel-level hit rates varying from 92% to 98% on different scripts and mention that the major failures happen because two neighbouring text lines touch each other significantly. These methods are sensitive to topological changes of the components in the text line.

2.6 Connected Component Based Method for Telugu [22].

Vijaya Kumar Koppula et al [22] proposed a connected component based method for segmentation of Telugu text line. It is a two step method, First we find the affinity between the neighboring CCs by their geometrical positions. Then identify and cluster the CCs of a line as explained here.

1. ***Establish The Relation Between Connected Components***
   Connected components $(CC_1, CC_2, ., CC_n)$ are generated from the text document binary image and each CC is circumscribed by a bounding box which can be represented by the coordinates top, bottom, left and right. i.e. (CC_T, CC_B, CC_L, CC_R). Then nearest neighbors of a CC on either sides which overlap vertically are find to establish the relationship between the connected components. Vertical overlapping is a spatial relation and is defined as follows:
   Any pair (CC_i, CC_j)

   (a) $CC_{iT} \leq CC_{jT} < CC_{jB} \leq CC_{iB}$.

   (b) $CC_{jT} \leq CC_{iT} < CC_{iB} \leq CC_{jB}$.

   (c) $CC_{iT} < CC_{jT} < CC_{iB} < CC_{jB}$,
   $(CC_{iB} - CC_{jT}) \geq_2^1 (height(CC_i)) and (CC_{iB} - CC_{jT}) \geq_2^1 (height(CC_j))$.

   (d) $CC_{jT} < CC_{iT} < CC_{jB} < CC_{iB}$,
   $(CC_{jB} - CC_{iT}) \geq_2^1 (height(CC_i)) and (CC_{jB} - CC_{iT}) \geq_2^1 (height(CC_j))$.

   In the above (a) and (b), is the situation where any one of the two (CC_i, CC_j) connected components totally overlaps with other. In (c) and (d) two connected components overlap each other, and length of overlap is greater than or equal to half of the heights. Distance measure between two connected components CC_i and CC_j is defined to determine the nearest neighbor of a CC. If $CC_{iL} \leq CC_{jL}$ then distance $d(i,j) =\mid CC_{jL} - CC_{iR} \mid$. Similarly if $CC_{iR} \geq CC_{jR}$ then distance $d(i,j) =\mid CC_{iL} - CC_{jR} \mid$. Hence the relations are established.

2. ***Identify And Cluster The Connected Components Of A Line***
   Construct an undirected graph in which each vertex represents a CC in the text image and edges are drawn to its nearest neighbors which are computed in the above step. Each connected graph represents a text line. When this process is applied to the entire page we may get m graphs which represents 'm' lines. Now, the CCs of a line are determined from the graph. If a pair (CC_i, CC_j), $CC_i \in l_k$ and CC_j is reachable from CC_i then CC_j is add to l_k, where l_k is k^{th} line. In some cases the consonant modifiers are not grouped with appropriate text line due to spatial relation constraint and it is segmented as separate line. An example is shown in Fig. 2.10. In Fig. 2.10 label 'X' shows a line which is not a proper text line. It is formed by the connection of consonant modifiers. These lines are small in height and may overlap with other lines.
   These lines are removed and the CCs of these lines are merged with its appropriate text lines using the following heuristic rules:

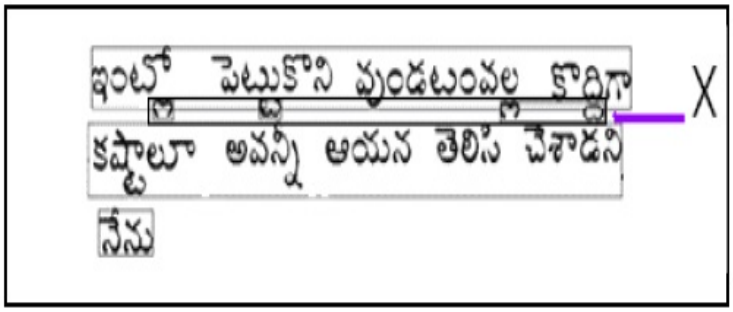


Figure 2.10: Line bounding box

- If any two lines (l_1 & l_2) overlaps vertically with each other and whose length of overlap is greater than or equal to half of height(l_1) or height(l_2) then l_1 & l_2 are merged.
- The lines, whose height is less than average height of CCs are removed and its CCs are connected to its nearest neighboring line.

2.7 Rationale and Description of Background Pixel Processing Approaches

While Kise et al [19] were first to apply segmentation of layouts through an explicit analysis of background white space, it was Baird [4] who spelt out the advantages of white space being generic in delimiting layouts. He also emphasized the simplicity of global analysis of space layout structure as compared to the multiple inter relations between text components. Two main characteristics very strongly supported background analysis. Firstly the 'nearly parameter free' characteristic, in addition to being generic, there are no prior assumptions of layout style to be made, or even those on text size. The second important characteristic was 'orientation free approach' to either page orientation or more importantly less sensitivity to text line orientation. To be fair Spitz [58] and Pavlidis [47] too had taken up the white stream analysis but not much was seen in their adoption. Here we briefly review some of the major approaches for background based segmentation.

2.7.1 White Streams Based Segmentation

Baird et al [4], Pavlidis and Zhou [47] proposed top-down approaches that divide a document image into smaller regions based on the structure of the background. The primary idea is to detect a set of maximal rectangles that do not contain any foreground pixels. All the maximal rectangles, i.e. white rectangles which cannot be further expanded, covering the background are enumerated. Then a partial order is specified according to the area and the aspect ratio of the rectangles. These rectangles divide the document image into regions that may contain text, images or graphics. A region classification algorithm could be applied to them before further processing. Antonacopoulos et al. [1] also proposed a similar approach. To obtain white rectangles, their method uses merging of white runs with predetermined tolerance of 'width change', so that it can isolate non-rectangular printed areas. The point is that the feature of maximal white rectangles is abandoned to handle pages with skew or non-rectangular layout. The common feature of the above three methods is that white

areas are represented as *rectangles* regardless of the skew and the shape of printed areas.

In order to obtain 2-dimensional isotropy, white areas should be represented as circles. Moreover, maximal circles are suited for simple representation. The method proposed by Normand et al. [41] is based on this viewpoint. They represent white areas as maximal regular octagons which are a good approximation of circles on digital images; the maximal feature is revived in their method. In the theory of digital image processing, a circle (or a set of equidistant pixels) is defined based on a variety of distance map definitions. Circles which maximally cover white areas can be obtained by skeletonization of background based on the distance transform. However, it is well understood that one of the major drawbacks of skeletonization is that the medial axes do not retain the connectivity of an original image. Since the task of page segmentation is considered to enclose printed areas, and the white areas originally enclose them, its clear that this is a flawed approach.

2.7.2 Background Thinning Based Segmentation

Kise et al. [21] present a method which is based on the thinning of the document background. The segmentation is determined by selecting subsequences of chains which circumscribe document blocks (loops). Algorithm attempts to filter away unnecessary chains and to retain such loops. Firstly, chains ending with a terminal pixel are removed. Remaining chains are analyzed in order to eliminate those lying inside the area of interest, and to preserve chains around the regions (e.g. between lines, columns). Two features are used: the distance of chain points from foreground pixels, and the so called 'difference of average line widths', which takes into account some characteristics of the adjacent foreground regions. The filtering process requires a critical tuning of some threshold which depends on the spacings of the input document.

2.8 Voronoi-diagram Based Segmentation Algorithm

Kise et al. [20] proposed the Voronoi-diagram based segmentation algorithm, which is also a bottom-up algorithm. In the first step, it extracts sample points from the boundaries of the connected components using a sampling rate s_r. Then, noise removal is done using a maximum noise zone size threshold t_n, in addition to width, height, and aspect ratio thresholds. After that the Voronoi diagram is generated using sample points obtained from the borders of the connected components. The Voronoi edges that pass through a connected component are deleted to obtain an area Voronoi diagram. Finally, superfluous Voronoi edges are deleted to obtain boundaries of document components. An edge is declared superfluous if it satisfies any of the following criterion:

1. The minimum distance d between its associated connected components is less than the inter-character gap in body text regions.

2. The minimum distance d between its associated connected components is less than the inter-line spacing times a margin control factor f_m, or the area ratio of the two connected components is above an area ratio threshold t_a.

3. At least one of its terminals is neither shared by another Voronoi edge nor lies on the edge of the document image.

The inter-character gap and inter-line spacing is estimate from the document. Variations in spacing may error in deleting the voronoi edge which results merging of lines and characters.

2.9 Fundamental Notions of Document Image Segmentation: Text Affinity and Space Affinity

In this section we attempt to bring out the essence of the various approaches and understand the common mechanisms. In the literature document image segmentation

is generally called as layout analysis. Structural layout analysis (Physical or geometric layout analysis) produces units of text which are organized hierarchically built upon the lowest level of a text "character". The collection of characters to words, words into lines, and then paragraphs, which in columns (if any) produces a "Text Block". The manner by which these units are produced by analysis in top-down or bottom-up manner gives the approaches those names.

Here it is assumed "Text block" is a homogeneous region with ideal geometry of regular boxed regions for lines, words and characters/recognition units in a MBR contains a CC.

However in every unit the background region pixels (whitespace or spaces) play an important role in implicitly limiting a unit. This is seen in the form of various spacings like inter-character spacing (printed text), inter-word spacing, inter line spacing, etc.

Where the ideal concept of text block segmentation gets into difficulty is when there are tables, photos, math symbols, etc which do not quite follows the homogeneity of the text regions.

Some approaches for text segmentation need pre-processing just so that the ideal rectangular notions can be applied. For example in the text by Gorman and Kasturi [43] the pre processing approaches have an ideal notion built into the approaches. These they call as projection profile method, baselines by Hough transform and Nearest neighbor (NN) clustering.

In Projection profile method, histogram of 'ON' pixels (text is ON) are accumulated along axes (horizontal or vertical commonly) and segmentation inferences are made by analysis of the histogram's peaks and valleys. Here text is along the peaks while spacing its accounting for the valleys. A similar observation is used for the separation of multi-column documents. They use term "plateaues" and "valleys", to describe higher histogram values (text) as against spacing (valleys) by Baird [18] variation CCs.

Hough transform approach [60], here its assumed that samples of text pixels are collected to cluster as a peak in Hough space to get the best angle that fits the baseline of text pixels. Nearest neighbor clustering approach is applied upon the direction

vectors for the centroids of components. Dominant direction indicates the direction where the ideal notion of line may be applied.
We observe a common axiomatic assumption in all three, that is text pixels must be aggregated, while background pixels are not much considered. This phenomenon we call as the "Text affinity" concept for building up a text unit, into a logical unit of a 'line'. We observe that notion of text affinity is applied with different constraints and parameters to obtain text lines. In the projection profile approaches the affinity constraint is specified in the direction of the projection where text presence is to be accumulated. In the Hough transform approach the Hough transform peak gives the angle constraint which again specifies the direction. Along direction if text affinity is applied then discontinuous groups of pixels are found and identified with the line. The notion of text affinity is made even stronger when applied to group of pixels from connected components in Ha et al [15].
While studying the role of text pixels in isolating blocks of text a natural question which arises is that why does background not play any role? Analogous to text affinity, we call this as "space affinity". When we look carefully it is seen that by complementarity (non text) and specification of background regions the geometrical limits of text blocks may be ascertained. Baird [4] very eloquently poses this problem. So we now put in another notion for computing text segmentation that is "Space Affinity".

Several useful properties of background based aggregation were given by Baird [4] such as generic values, lesser dependence upon parameters, line orientation and page orientation. Before Baird [4], Kise et al [21] were first to propose space affinity based approach and quite interestingly Spitz [58] also used white streams. So now while its possible to compute page layouts through text and space affinity. We must add certain more parameters to make them more successful.

Conceptually text segmentation can be viewed as the discovery of a structure of text in the image geometry. The better performing approaches give a closer match between the conceptual view to the input. Various document image segmentation

techniques have been proposed in the literature and their working may be based on either concept of grouping of text or background information. So now we reiterate that text segmentation can be said to be influenced by two concepts:

1. Segmentation by an aggregation of text components based on a notion of 'Text Affinity' as opposed to

2. Segregation of text based on background information (white space) a notion of 'Space Affinity'.
   Other attributes also influence Line segmentation.

While these are the major concepts from the context of other attributes like line orientation which can influence the direction in affinity is to be considered.

1. Orientation:-This is needed for both approaches. A reading order generally left to right in most occidental scripts and also in Indic script. In some scripts reading order is from top to bottom.

2. Context of Application:- Given a notion of line segmentation based upon any affinity the context of application is also very important. Local and global influences need to be balanced out. This we call as a context. Here whether the context may be global (full page), semi-global (page width), local (line) or stripes across lines.

2.9.1 Classes of Segmentation Problems

In our observation of approaches we find that almost universally every approach attempts to solve the problem exclusively through a proposition of text affinity or space affinity. However we state that a "Complex segmentation text problem" (document containing touching and overlapping components, text skew, document skew and space variations) may require interplay of both approaches. So while a simple text document segmentation (skew free document with distinct linear space between lines, words and characters) may be solved by a simple exclusive text or space affinity

approach but more complex solutions require more sophisticated specification of controlling it with orientation and context. Most complex specification of problems like free form handwriting segmentation requires both. We give a conjunctive approach of both text affinity and space affinity based method using Fringe Maps in the ensuing parts of this thesis. First however some background is required for our technique which makes use of fringe distances.

2.10 Conclusion

In this chapter, we studied different text segmentation approaches. These approaches are classified into top-down and bottom-up manners. In top-down approach page image is divided into paragraphs, lines, words and characters. Similarly in bottom-up approach characters are grouped into words, words to lines and lines to paragraphs. We extracted the characteristics of these approaches and conceptualized as Text Affinity based based and Space Affinity based methods. Text affinity is aggregation of text components. Space affinity is segregation of text components based on background information. All these approaches are based on either Text affinity or Space affinity. Text affinity based methods are sensitive to touching and skew. Similarly space affinity based methods are sensitive to space variations, skew and huge gap between words. In our observation both text and space affinity may be used to give good results on complex documents. Along with this attributes like orientation and context of application give better accurate results on text segmentation. To make clear our notions we first take up the necessary background through distance transform and fringe maps. Their utility for document image segmentation is demonstrated.

Chapter 3

Distance Transform and Its Applications in Document Image segmentation

3.1 Distance Transform

In this chapter we take up fringe maps which are a kind of distance transform. We study how fringe maps may be applied to segmentation of document images. A distance transform, also known as distance map, is a derived representation of a digital image [7]. The map labels each pixel of the image with the distance to the nearest pixel of the specific type of label. A label can also be called as an 'obstacle pixel'. A most common type of obstacle pixel is a boundary pixel in a binary image. The distance transform can be seen also as an operator applied to binary images. The result of the transform is a kind of gray level image that looks similar to the input image, except that the gray level intensities of pixels inside foreground regions are changed to show the distance to the closest boundary from each point. We examine this more formally. In Felzenszwalb and Huttenlocher [14] defined distance transform formally. Let G be a regular grid of binary image and $f : G \to \Re$ is a function on the grid. They define distance transform of f as

$$D_f(p) = min_{q \in G}(d(p,q) + f(q)) \tag{3.1}$$

where *d(p, q)* is some measure of the distance between p and q, where $p, q \in G$ and are points in the grid. Intuitively, for each point p we find a point q that is close to p, and for which *f(q)* is small. Note that if f has a small value at some location, D_f will have small value at that location and any nearby point, where nearness is measured by the distance *d(p, q)*.

However we note that in (3.1) the first is the conventional DT. We can take a special case of (3.1) as

$$D_p(p) = min_{q \in p}(d(p,q) + 1(q)) \tag{3.2}$$

Here if we consider P as a set of object pixels which are connected through either definitions of 4 or 8 connectedness. Also then we need

$$1(q) = \begin{cases} 0, & \text{if } q \in P \\ \infty, & \text{otherwise} \end{cases}$$

which makes 1(q) an indicator function for the membership to set of object pixels P. One way to think about the distance transform is to first imagine that foreground regions in the input binary image are made of some uniform slow burning inflammable material. Then consider simultaneously starting a fire at all points on the boundary of a foreground region and letting the fire burn its way into the interior. If we then label each point in the interior with the amount of time that the fire took to first reach that point, then we have effectively computed the distance transform of that region. Figure 3.1 shows a distance transform for a simple rectangular shape.

There are several different sorts of distance transform, that can be computed depending upon which distance metric is being used to determine the distance between pixels. The example shown in Figure 3.1 uses the 'chessboard' distance metric but both the Euclidean and 'city block' metrics can be used as well.

Common Distance Metrics:

1. Euclidean (L_2 norm)

0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0

(a)

0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	0
0	1	2	2	2	2	2	2	1	0
0	1	2	3	3	3	3	2	1	0
0	1	2	3	4	4	3	2	1	0
0	1	2	3	3	3	3	2	1	0
0	1	2	2	2	2	2	2	1	0
0	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0

(b)

Figure 3.1: a)Binary Image b)Distance Transformation using chessboard

0	0	0
0	1	0
0	0	0

(a)

$\sqrt{2}$	1	$\sqrt{2}$
1	0	1
$\sqrt{2}$	1	$\sqrt{2}$

(b)

0	1	0
1	0	1
0	1	0

(c)

1	1	1
1	0	1
1	1	1

(d)

Figure 3.2: Different Distance Transforms. a)Image b)Euclidean c) City Block d) Chessboard

2. City Block (L_1 norm)
3. Chessboard (L_∞ norm)

The Euclidean distance is the straight-line or shortest distance between two pixels.

$$\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2} \tag{3.3}$$

The city block distance metric measures the path between the pixels based on a 4-connected neighborhood pixels.

$$|X_1 - X_2| + |Y_1 - Y_2| \tag{3.4}$$

The Chessboard distance metric measures the path between the pixels based on an 8-connected neighborhood pixels.

$$\max(|X_2 - X_1|, |Y_2 - Y_1|) \tag{3.5}$$

Different distance transforms are illustrated in figure 3.2, computed based on these metrics. In figure 3.1(a) P has all 'ON'('1') pixels for which figure 3.1(d) was computed using L_∞ norm.

3.2 Procedures to Compute Distance Transform

Once the metric has been chosen, there are many ways of computing the distance transform of a binary image. One intuitive but extremely inefficient way of doing it is to perform multiple successive erosions with a suitable structuring element until all foreground regions of the image have been eroded away. If each pixel is labeled with the number of erosions that had to be performed before it disappeared, then this is just the distance transform. The actual structuring element that could be used depends upon which distance metric has been chosen. A 3×3 square element gives the 'chessboard' distance transform, a cross shaped element gives the 'city block' distance transform, and a disk shaped element gives the Euclidean distance transform. Of course it is not actually possible to generate a good disk shaped element on a discrete grid on a small scale, but there are algorithms that vary the structuring element on each erosion so as to approximate a circular element. The computation can be observed as the minimum convolution. Felzenszwalb and Huttenlocher [14] go to great pains to show a 'separable' algorithms that compute the minimum convolution for two dimensional grids. While they have worked with generalizations called as "sample functions". It is directly applicable in $2D$ images such as document images.

Approach to sequential DTs was first published in 1966 [55], and parallel ones in 1968 [54]. These papers present the basic idea, and some DTs. An original binary image, to which the DT is to be applied, consists of feature pixels with the initial value zero, and non-feature pixels with the initial value infinity, i.e., a suitably large number. All DTs are described in graphical form as "masks", see figure 3.3. The constants c_n, are the local distances that are propagated over the image. The size of the neighborhood can vary. In figure 3.3, a 5×5 neighborhood is illustrated. The computation of the DT is either parallel or sequential. In the parallel case the center of the mask in the figure 3.3(a) is placed over each pixel in the image. The local distance in each mask-pixel c_n, is added to the value of the image pixel "below" it (including the central zero). The new value of the image pixel is the minimum of all the sums. The process is repeated until no pixel value changes, i.e., the number of

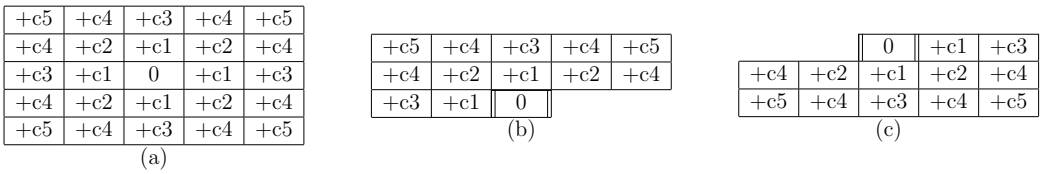


Figure 3.3: Masks describing the distance transformations. a)Parallel computations b) & c)Sequential computations

iterations is proportional to the largest distance in the image. The parallel procedure is thus:

$$v_{i,j}^{m} = minimum_{(k,l)\in mask}(v_{i+k,j+l}^{m-1} + c(k,l)) \tag{3.6}$$

where $v_{i,j}^{m}$ is the value of the pixel in position *(i,j)* in the image at iteration m, *(k, l)* is the position in the mask (the center being (0,0)), and *c(k,l)* is the local distance from the mask. A small example of the parallel step is shown in figure 3.4(a).

The sequential procedure also starts from the zero/infinity image. The symmetrical parallel mask is split into two masks, shown in figure 3.3(b) & (c). The masks are passed over the image once each: the forward one from left to right, and from top to bottom, and the backward one from right to left and from bottom to top. The new value of the "central" image pixel is the minimum of the sums of the image pixel values and the local distances c_n, as before. After these two passes the distance transform is computed. The sequential step is thus:
Forward:
for i = (size + 1)/2 . . . . . lines do
for j = (size + 1)/2 . . . . , columns do

$$v_{i,j} = minimum_{(k,l)\in forwardmask}(v_{i+k,j+l} + c(k,l)) \tag{3.7}$$

Backward:
for i = lines - (size - 1)/2 . . . . . 1 do
for j = columns - (size - 1)/2 . . . . . 1 do

$$v_{i,j} = minimum_{(k,l)\in backwardmask}(v_{i+k,j+l} + c(k,l)) \tag{3.8}$$

where "size" is the side-length of the mask and the rest of the notation is the same as in (3.4). A small example of the sequential step is found in figure 3.4.

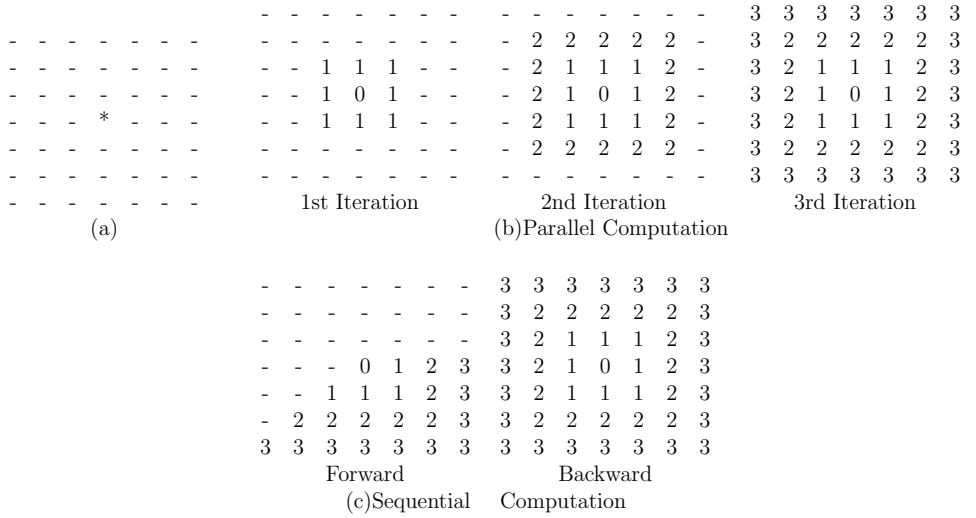


Figure 3.4: Stages in computation of a DT. a)Original Image b)Parallel Computation c)Sequential Computation

3.3 Fringe Maps in OCR

Brown [9] arrived at fringe distances to find the similarity between two images. It was from the need to compare similarity which was robust to minor image variations. Unlike Gaussian blurring which needed a parameter fringe distances can be computed for any image without a parameter. Efficiency of computation of fringe distance was also a motivating factor with integer only computation. Fringe distances were proposed to efficiently match a template image against an input for recognition of the input against the stored templates.

To compute the fringe image in our case we require a binary input image. Document pages could be binarized using any of the excellent methods by either global or local thresholding [56]. First we observe carefully the algorithms given by Brown [9]. The top level function finds the similarity distance between two inputs *a, b*. Which already have been transformed into fringe maps. The main function which compute the fringe map is given as getd(binary, *d_map*), Where binary is the input binary image and *dmap* is 2D array where the fringe map is computed and store for lookup by the top level function distance(*dmap_a*, *dmap_b*). Now we analyze the operations performed in getd().

Algorithm 1: Fringe Map getd() Function

Input : A binary Image 'binary'

Output: Distance Map of binary image 'dmap'

```
1  Intialize dmap(i,j) to '0' where (i, j) ∈ P, for all other dmap(i, j) is
   'empty'; P is set of object pixels.;
2  wcount = Number of white pixels in the binary image ;
3  Intialize fringe_value = 1;
4  Store (i, j) in the list where dmap(i, j) = 0;
5  repeat
6      Read each element (x, y) from the list and examine its 8 neighbors ;
7      if neighbor is 'empty' then
8          Store dmap(x, y) = fringe_value ;
9          decrement wcount ;
10         Store (x, y) in templist;
11     Store list = templist;
12     Increment fringe_value ;
13 until wcount = 0;
```

In computing the fringe distance we observe the algorithm getd() shows that it is equivalent to a DT with L_∞ distance metric.

The concept of fringe maps is related to distance transforms for binary images. In a fringe map each pixel is represented with a fringe number (fringe value) at its position. Every print (black) pixel has a fringe number of zero. Background (white) pixels have a fringe number which is a positive integer, that is the distance from the nearest black pixel using a L_∞ distance metric. In other words, A white pixel with fringe value x states that:

1) It is x pixels away from its nearest black pixel.

2) There is at least *(x-1)* number of white pixels in all 8 directions of that white pixel.

For us the second point is very useful. It helps us to quantify the interstices between

the components in the image. figure 3.5 shows sample text and its fringe map.

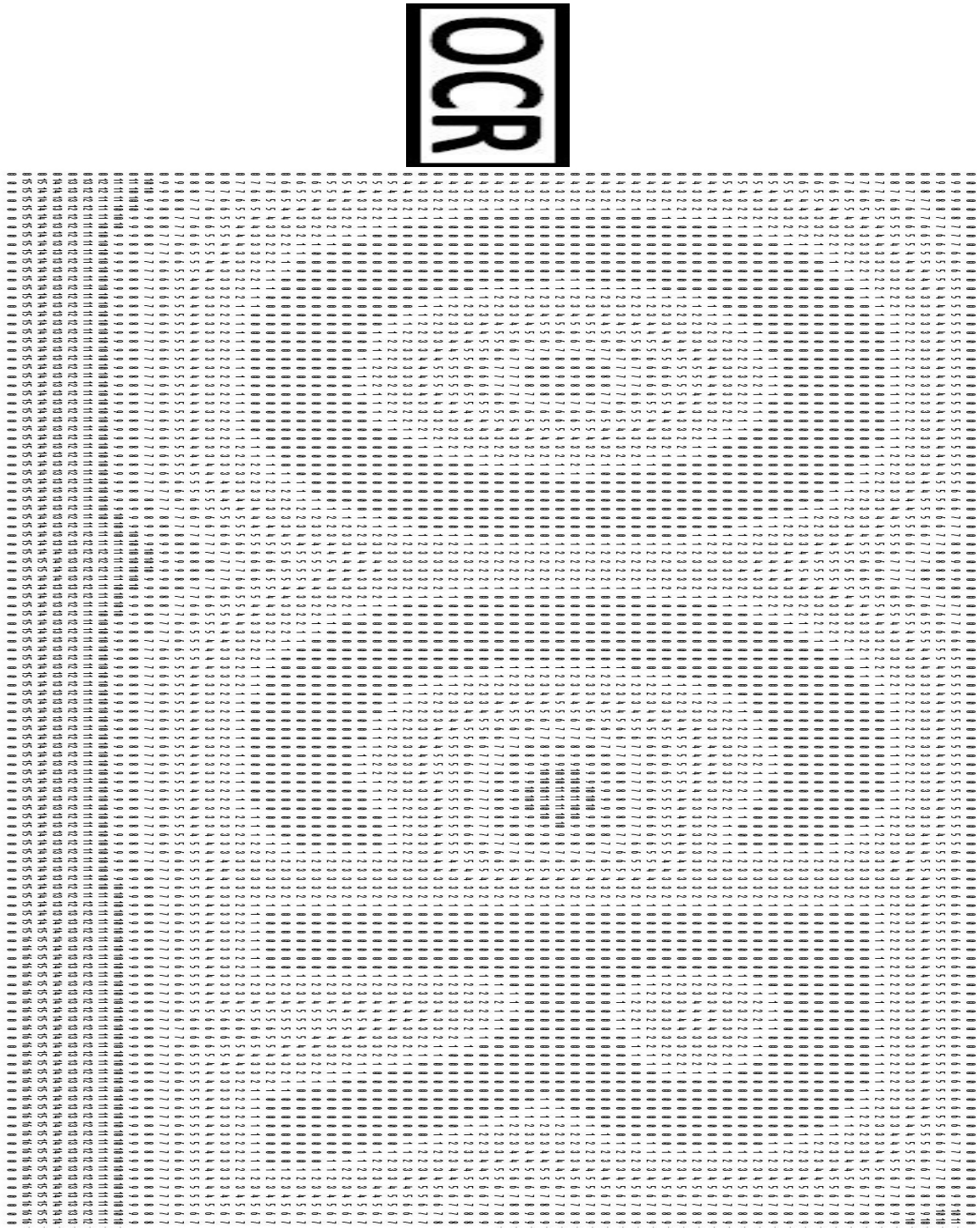


Figure 3.5: An Example text and its Fringe map

R.L Brown [9] proposed fringe maps for OCR. Negi et al [39] adopted it as an approach for the recognition of Telugu characters and proposed an OCR system for Telugu. Reducing the very large number of recognition symbols to basic templates of order of few hundred symbols. The fringe distance is used as distance measure for the comparison of Telugu character binary images. Fringe distances compare only the black pixels and their positions between the templates and the input images. Fringe maps reduce the computing cost in the template matching. Fringe distance may be even more efficiently computed by precomputing and storing the distances of the nearest black pixels of each pixel position of the template.

3.4 Fringe Maps in Segmentation

The focus of Negi and Brown's work [9,39] was a recognition but it did not suggest any robust approach for segmentation. Here we take up an important aspect of work which is a novel concept for document image segmentation. As we have already seen, any of the efficient distance transform computation approaches could be used to compute the fringe map of an input document image. Here we shall investigate and show some operations on fringe maps and their utility in document image analysis. First we shall study the essential aspects of the statistical distribution of the fringe values for a typical document image. We expect input is skew free binary image for which fringe map is already constructed. An example image and its histogram of fringe map is shown in figure 3.6. We observed in figure 3.6 histogram of fringe values are stable (low variation between consecutive frequencies) at some levels . To have a better empirical view for analysis we find distance between the frequencies in the histogram $f_i = |f_i - f_{i+1}|$, where $f_i = 1$ to n, $f_0 = 0$ as shown in figure 3.7. The distance between the frequencies in the histogram is smoothened for sharp peaks and valleys as shown figure 3.7. We used moving average for smoothing with a window size 5, $f_i = \frac{f_{i-2}+f_{i-1}+f_i+f_{i+1}+f_{i+2}}{5}$. We assumed peaks and valleys in the smoothened histogram may be used as threshold for doing experiment on fringe map.

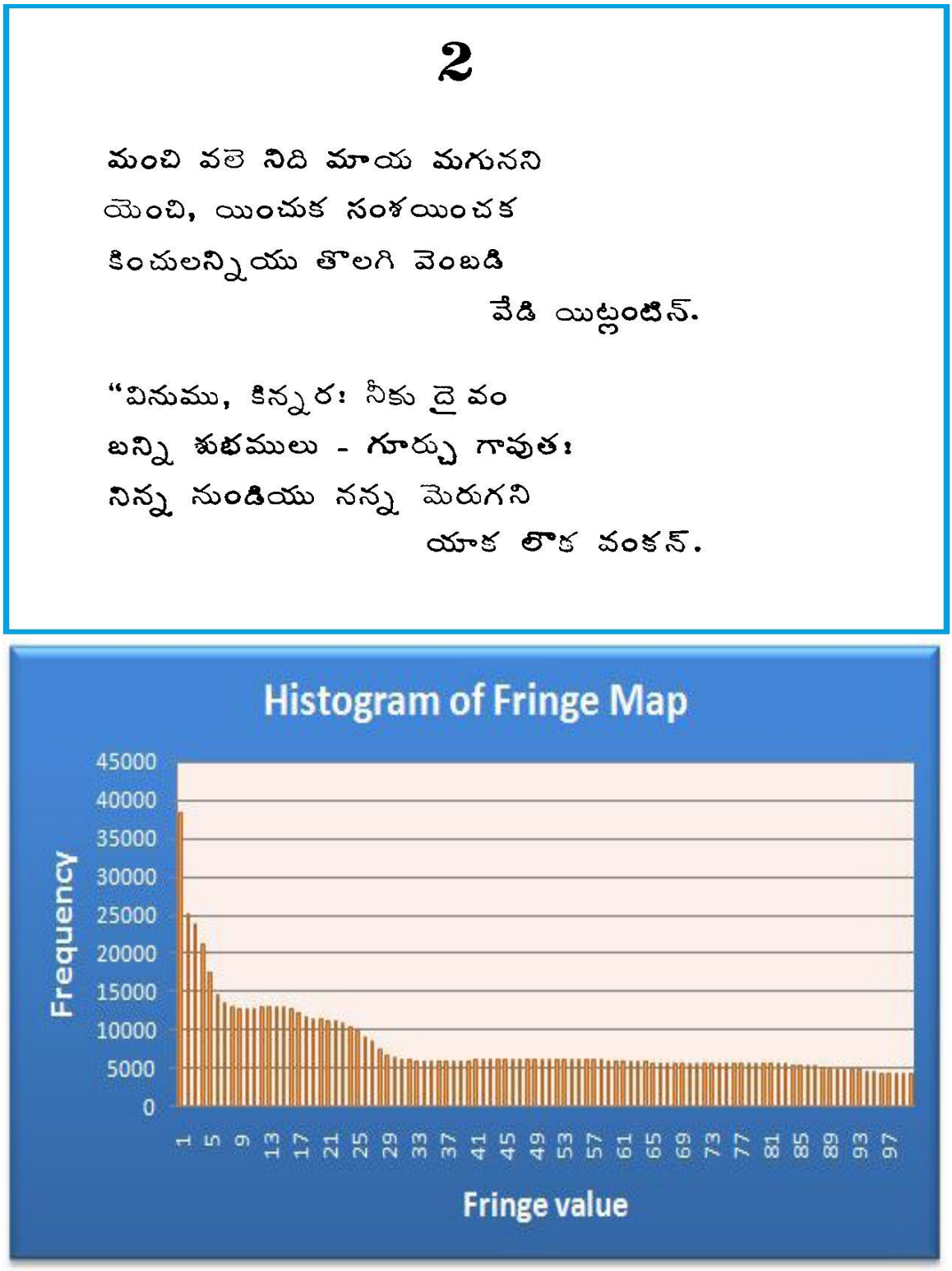


Figure 3.6: An example image and its histogram of fringe map image.

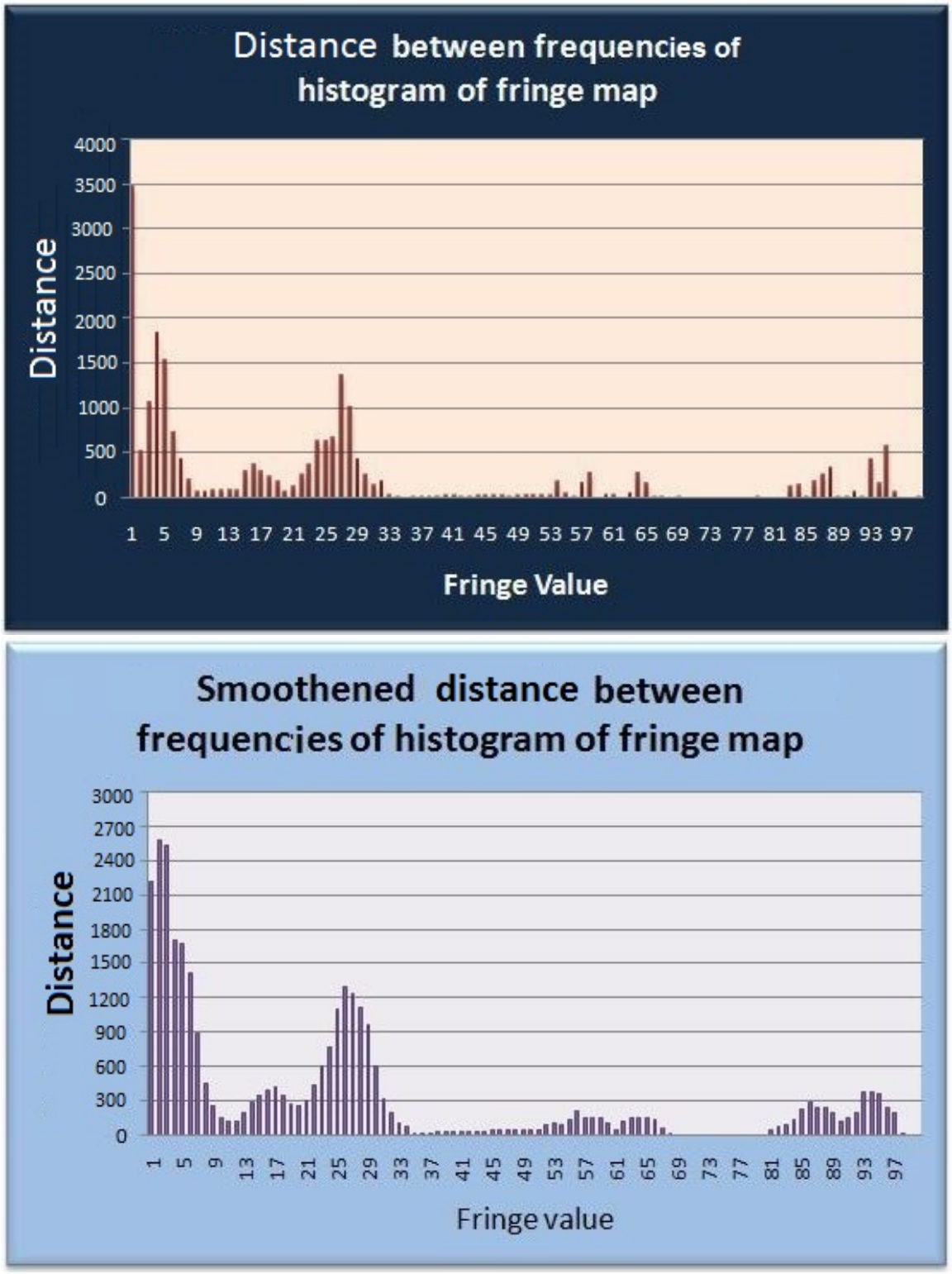


Figure 3.7: Smoothed histogram of fringe map image of figure 3.6.

3.4.1 Segmentation Using Threshold operation on Fringe Maps

Thresholding is one of the widely used methods for image segmentation. It is useful to discriminating foreground from the background. By selecting an adequate threshold value T, the gray level image can be converted to binary image. The binary image should contain all of the essential information about the position and shape of the objects of foreground. Fringe map may be seen as one kind of gray level images. As we discussed above threshold operation can be applied on fringe map for segmenting different units. We experimented with the image in figure 3.6. Here we take different threshold values T=2, 10, 16, 20, 26, 34, 56, 61, 64, & 69. These values are peaks and valleys of smoothened distance between frequencies of histogram of fringe map image in figure 3.7. The resultant images with different threshold operations are shown in figure 3.8. Result shows that significant threshold operations can segment text into words, lines and paragraphs. figure 3.8(b) shows the threshold segmenting words. Similarly figure 3.8(c) & (d) showing candidate text lines and figure 3.8(e) & (f) are segmenting image into paragraphs. However finding the best threshold is a difficult task. An analysis of the histogram shows both global and local information. It appears better segmentation of text needs localized information. We can't apply global threshold for segmenting the image where space variation due to overlapping of components, touching of components and text skew within the line.

Figure 3.8: Threshold operation on Fringe map of image in Fig. 3.6. (a)T=2,(b)T=10, (c)T=16, (d)T=20, (e)T=26, (f)T=34, (g)T=56, (h)T=61, (i)T=64, & (j)T=69.

Here we see in the figures the gradual evolution of the segments. At one end we

have individual connected components. When examining the valley and peaks with increasing value we observe the following. Thresholding at the first valley that is *10* appears to segment out words. The value *16* is close to the several peak in the smoothed histogram. Now certain components that are lines or parts of lines seem to be segmented. At the next threshold value that *20* is the following valley. Now we see that while components have merged into lines mostly but vertical joining has emerged. In subsequent thresholds these merge into the text blocks. The thresholds beyond this show how entire printed matter merges into a single text block.

3.4.2 Segmentation Using Horizontal and Vertical Projection Profile of Fringe Map

Projection profile is one of the popular method used for segmentation. A projection profile is a histogram of the number of black/white pixel values accumulated along parallel lines taken through the document [18]. Histogram of black/white pixels in row wise is Vertical Projection Profile (VPP) which is used for text line segmentation. Similarly histogram of black/white pixels in column wise is Horizontal Projection Profile (HPP). It may be used for word and character segmentation. The mathematical representation for Vertical Projection Profile (VPP) and Horizontal Projection Profile (HPP) are following:

$$VPP(y) = \sum_{x=1}^{N} I(x,y) \qquad HPP(x) = \sum_{y=1}^{N} I(x,y) \tag{3.9}$$

Projection of black pixel values in fringe map is same as in binary image. But fringe map of inverse binary image projection shows more precise peaks than binary image projection as shown in figure 3.9. Observe that projections in the figure 3.9(b) and (c) look similar in the positioning of peaks and valleys. However an absolute value is different and greater in the fringe profile. At times this may be useful for segmenting noisy documents.

Similarly when we look for space affinity projection i.e. projection profiles of white pixels. The vertical projection profiles of binary image in figure 3.10(a) shows ragged

peaks and valleys as shown in figure 3.10(b). There is no distinct peak through which we can find segmenting path. However, when we find vertical projection profiles of fringe map for the image figure 3.10(a). It shows distinct peaks which can be used for segmenting lines. The result of vertical projection profile is shown in figure 3.10(c). We assumed the peaks as segmenting points and plotted in the image as segmenting line is shown in figure 3.11. In the profile of fringe maps the maxima peaks are easier to identify and give a more direct indication of the positioning of the segmenting line. This is very clearly seen in figure 3.11. This gives tentative lines for segmentation of text lines. As we discussed in section 3.4.1 taking global information for projection profiles may be sensitive for skewed document images. Always global and local information are required for designing efficient segmentation algorithms.

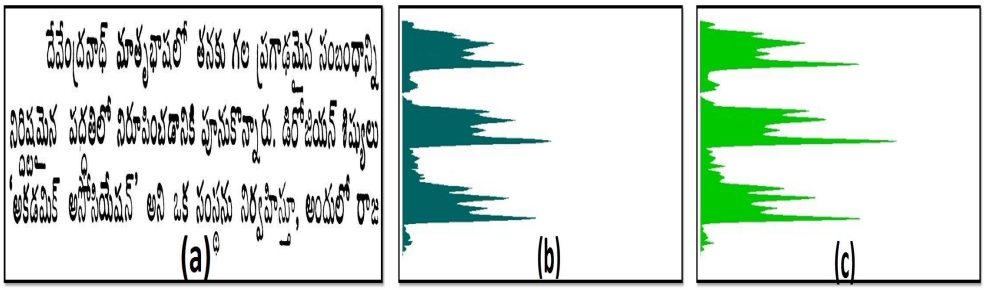


Figure 3.9: Projection profiles of black pixels. (a) Input Image, (b)Projection of black pixel in binary image, (c) Projection of black pixel in fringe map of inverse binary image.

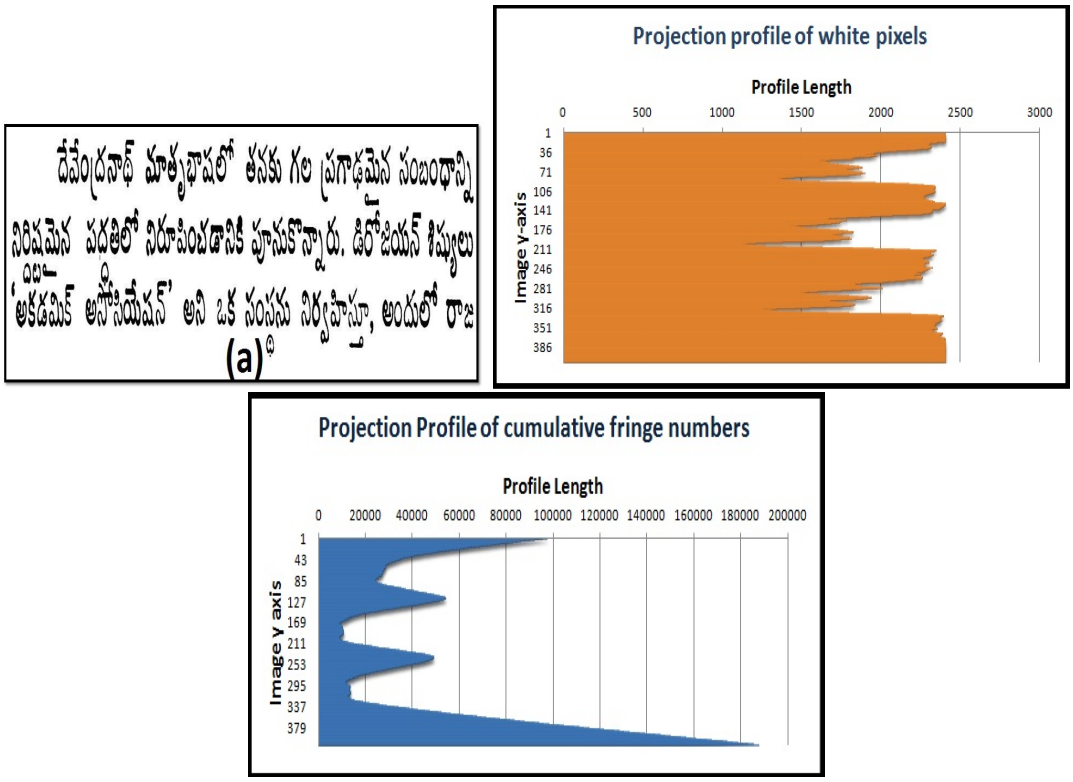


Figure 3.10: Projection profiles of white pixels. (a) Input Image, (b) Projection of white pixel in binary image, (c) Projection of cumulative fringe numbers in fringe map of binary image.

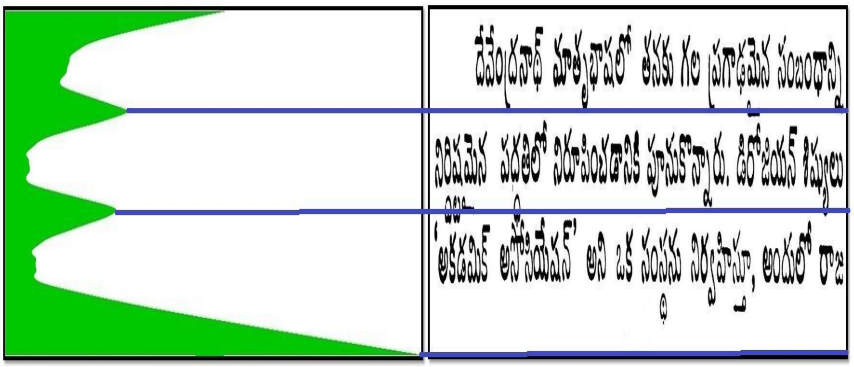


Figure 3.11: Peaks of Vertical projection profiles of white pixel in fringe map are plotted on the image.

3.4.3 Peak fringe number

Now we look at more applications of the maxima of fringe values. We use maxima of fringe numbers to find interstices between components. This is made more formal using the PFN definition.

Definition: PFN

In a fringe map, Peak Fringe Number (PFN) is defined as a maximum positive fringe number (white pixel) enclosed between two zeros (black pixels) in the direction of interest.
An example text in Fig. 3.12(a) and PFNs in its fringe map is shown in Fig. 3.12(b), PFNs in vertical direction are shown in circles and PFNs in horizontal direction are shown in boxes.

Significance

When we compute the fringe map for an entire page we see more interesting things. We find PFNs in the fringe map for a given binary image where the direction of interest is vertical (perpendicular to text orientation) and horizontal (same as text orientation) because we assumed the text runs in a horizontal direction.

The PFNs are present at various locations in the fringe map. We observe that PFNs are located *inside* and *outside* the connected components of character objects. The PFNs outside the connected components are used for generating segmenting paths between columns, lines, words and characters. Fig. 3.13 shows all PFNs superimposed over the image. Here this gives raw indication on the segmenting application of PFNs. This needs more refinement as we shall see in the following chapter.

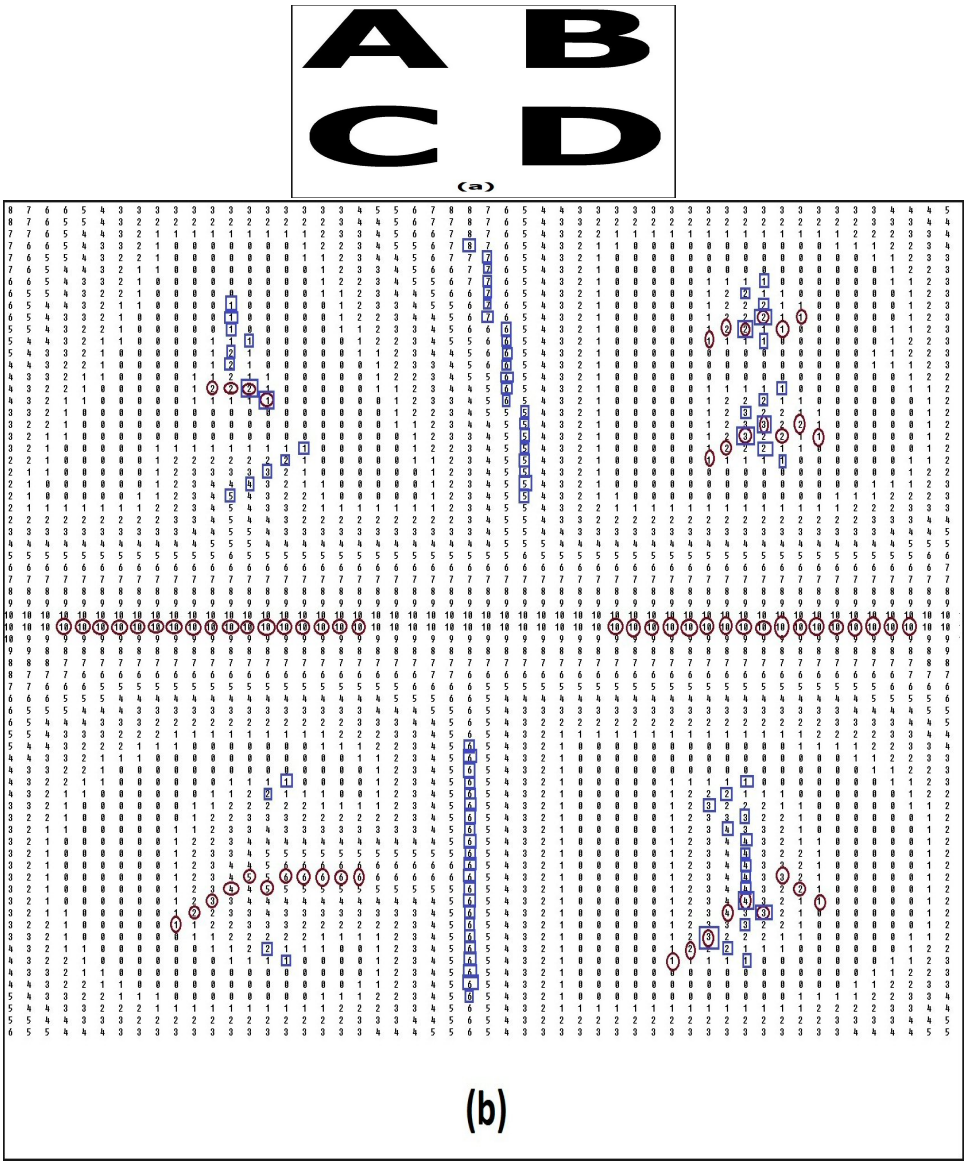


Figure 3.12: (a) English Text (b) Fringe map of image (a), PFNs in horizontal direction are in boxes and PFNs in vertical direction are in circles.

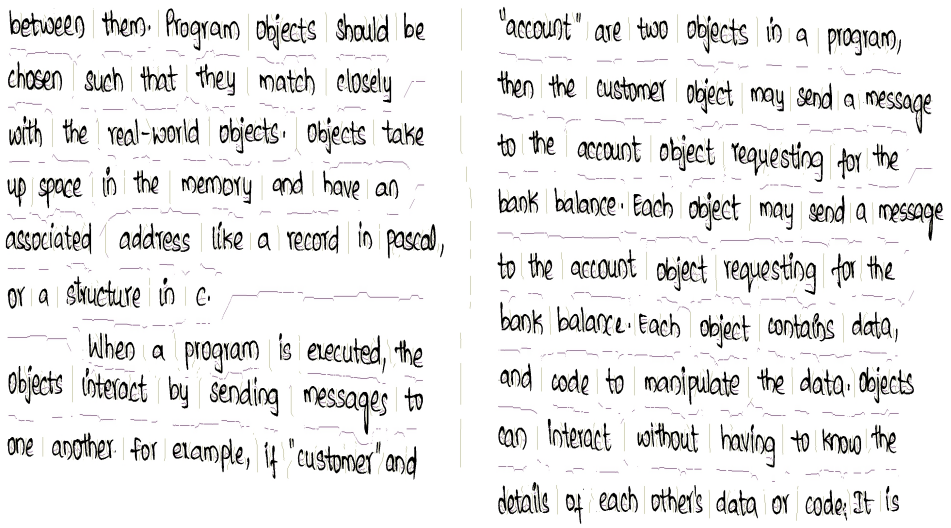


Figure 3.13: PFNs of an image in vertical and horizontal direction are shown in different colors (shades).

Distributions of PFNs and Filtering operations

Now we show the different distributions of PFNs in the fringe map. The distribution of PFNs in vertical direction is shown in Fig. 3.14. This distribution represents the mixture density of the distributions of PFNs between lines and PFNs within the connected components (CCs) in those lines. These PFNs reflect the interstices between lines and inside components. We aim to exploit the PFNs between lines to segment text lines. We can extract these PFNs by filtering out other PFNs that lie inside CCs from the mixture of PFNs found in vertical direction. The distributions of PFNs inside the individual CCs and between text lines is shown in Fig. 3.14(b) & Fig. 3.14(c). From these distribution graphs we observe that the numeric fringe value of PFNs inside the CCs is less than the PFNs values between the text lines. Therefore we can use this observation to filter out such PFNs. We use an adaptive concept which is computed for each document image. We label connected components in the document and filter PFNs between the pixels of same label which gives better filtering.

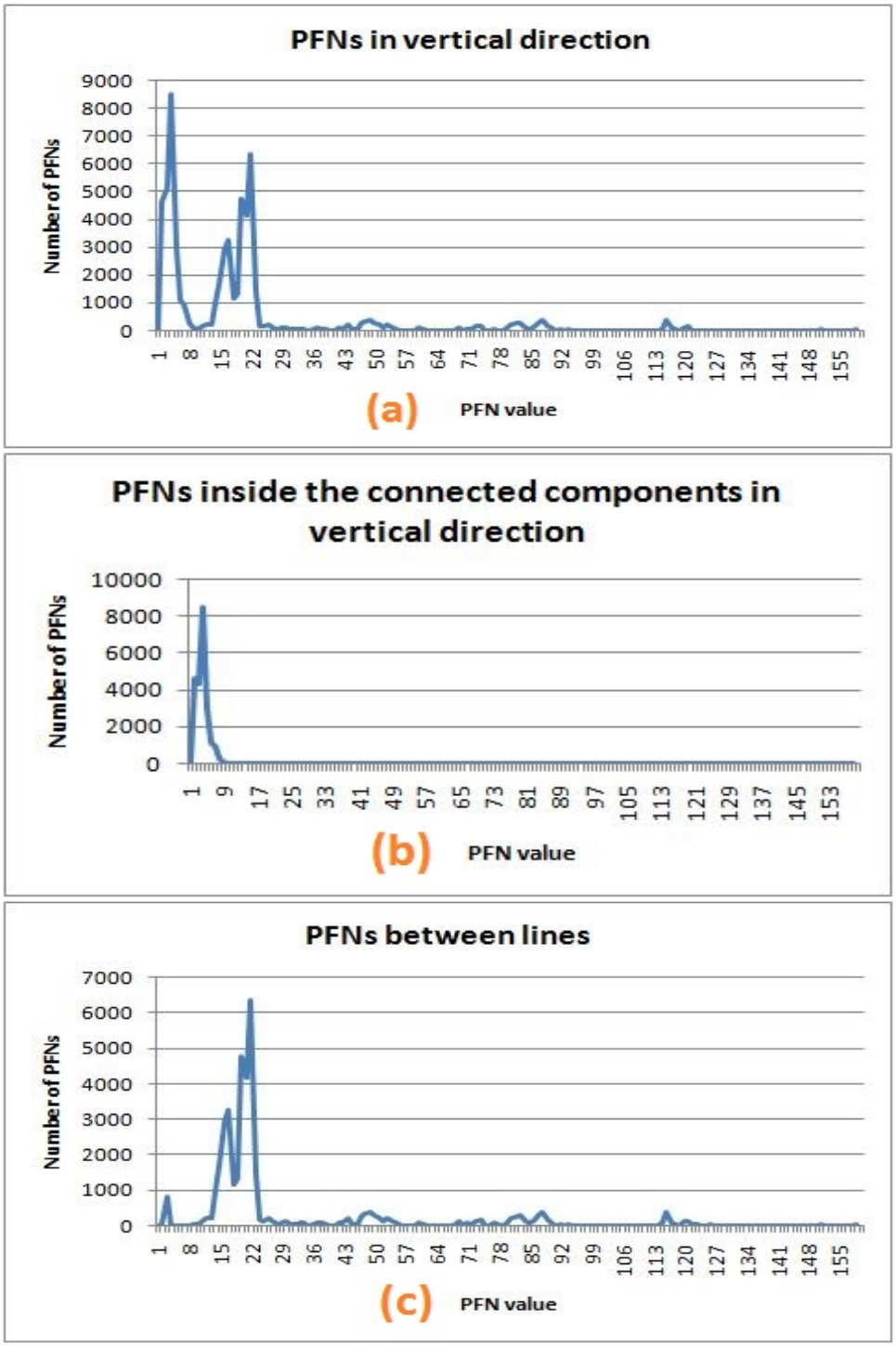


Figure 3.14: Distribution of PFNs in vertical direction of an image figure 3.13. (a) All PFNs, (b) PFNs inside the CCs, (c) PFNs between text lines.

In the same way the distribution of PFNs in horizontal direction can be analysed. This represents the mixture model of the distributions of PFNs inside the components, between characters and between words and columns. Fig. 3.15 shows different distributions of PFNs in horizontal direction. Fig. 3.15(a) shows the distribution of all PFNs in horizontal direction, Fig. 3.15(b) the distributions of PFNs inside the components, Fig. 3.15(c) the distributions of PFNs between the characters and Fig. 3.15(d) the distributions of PFNs between words and columns.

Once PFNs are filtered and categorized then we can use these PFNs for segmenting characters, words and columns. In general, the variation between interstices inside the components and between the characters is very low as shown in Fig. 3.15(e) with the superimposed distributions of PFNs. However as shown in Fig. 3.15(e) the intersection point of the distributions of PFNs between the characters and between the word and columns is the better choice of a threshold for aggregating the components of a word. We assume PFNs greater than the threshold are those between words and columns. Finding an exact intersection point is difficult. However we find this by taking threshold T1 as the mean of Peak of histogram of the PFNs outside the components, and weight average of PFNs outside the component. We observe that as expected the PFNs greater than or equal to T1 with some tolerance are usually found between the words and columns. Thus we have filtered out the desired PFNs from all the PFNs in horizontal direction. These PFNs can now be used for segmenting words and columns. The PFNs remaining after this filtering are between characters and inside the components. As we discussed above the filtering method of PFNs inside the CCs is used here to separate the PFNs between characters, and PFNs inside the components. The PFNs between characters are used to segment characters.

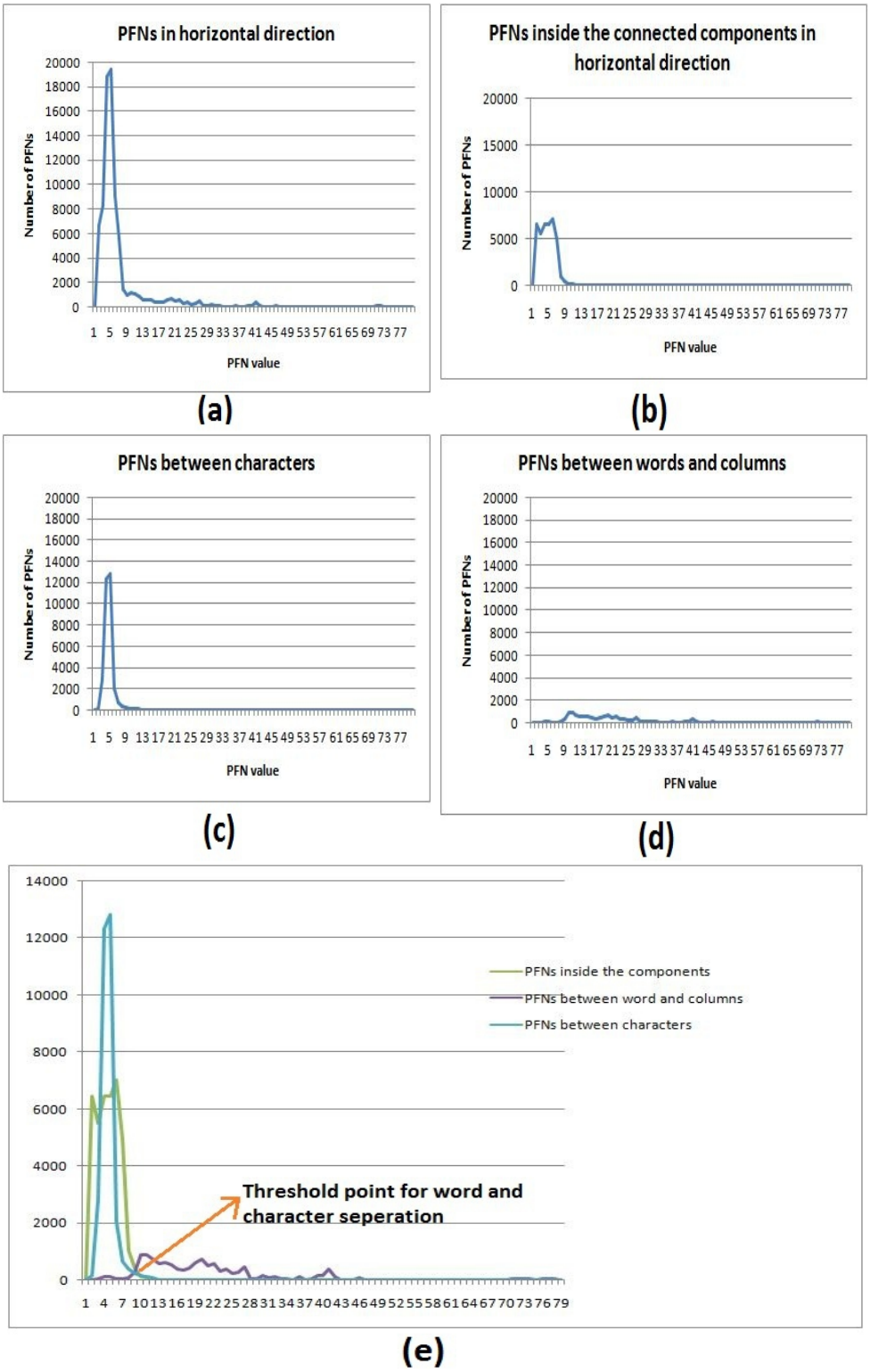


Figure 3.15: Distribution of PFNs in horizontal direction of an image figure 3.13.

Region of Influence of PFN

As we discussed in section 3.3 the fringe number 'X' have $(X-1)$ white pixels in all 8 directions. This we use to find the region of influence of a PFN, $ROI(PFN)$. Let $A = F(i,j)$ be a PFN with fringe value 'X' where F is the fringe map. Then $ROI(A)$ is $(X-1)$ pixel distance in all directions, in other words we can show in bounds,

- $Top = (i - X)$
- $Bottom = (i + X)$
- $Left = (j - X)$
- $Right = (j + X)$

3.5 Conclusion

In this chapter our most important observation is the concept of 'Peak Fringe Number', as a local maxima of the fringe value. Fringe values were also shown to be useful for segmenting text. Fringes derive from the distance transform. Like in more conventional approaches we find that the projection profile approach using fringes is also useful in making a segmentation. Analysis on the PFN values is useful to indicate inter line or inter word separations. In the following chapter we propose several detailed approaches that bring together the affinity notions for segmenting using PFNs as a tool.

Chapter 4

Text Line Segmentation for Telugu Script using Fringe Maps

4.1 Introduction

In the overall process of document image segmentation there is a basic dichotomy of text and image blocks. For a text block further sub-division is needed into the units of lines and words etc. Text line segmentation is one of the most important steps in document image segmentation and efficiency of an OCR system depends very much on this segmentation phase. Errors caused in line segmentation cannot be overcome by subsequent stages in the OCR process. Segmentation of text lines is apparently a simple process in documents where the spacing between text lines is large and distinct. However in the case of printed Telugu documents several challenges exist. Some of the challenges are

1. Skew variability [52], that is in addition to any skew in the document image, skew may be different between various text lines. At times we may even have skew variation within a text line(text set along a curve etc.).

2. Line proximity, here white space between lines (vertical) is less and overlapping and touching of components of adjacent text lines may be seen.

3. Complex Multi-Component Orthographic Units: This is seen in several Indic

scripts such as Telugu, Tamil, Bangla, and Malayalam etc. Here the complex orthography has multi-component units in a close spatial context. The problem is compounded by some of these components being at different vertical positions as compared to the main text baseline, as in Telugu.

In the literature (as seen in chapter 2), the problem of segmenting documents is tackled classically by the use of techniques such as projection profiles [31, 38, 57], Run Length Smoothing Algorithm (RLSA) [39, 57] and the connected component [22,42,57] algorithms. These methods fail [33,34] to produce accuracy in segmentation of documents for the scripts mentioned earlier. Here in this chapter we propose fringe based methods for segmentation of text lines which extend the basic observations and concepts of previous Chapters(2,3).

4.2 Various Algorithms using PFNs

We show in this chapter a progression of the methods that exploit the key concept from fringe maps that is the PFN. In successive approaches we show how the concept must be adapted for increasingly complex situations. General steps in our algorithms are:

1. Generate Fringe map for the input binary image
2. Compute PFNs
3. Filter PFNs
4. Linking PFNs between the adjacent text line
5. Computing segmenting path

4.2.1 Notation in Text Line Segmentation

In this subsection we define some notation that will be useful for the formulation of text line segmentation approaches using fringes. By observing Fig. 4.11 we can see the following

- Top(CC) : Top of the CC in the minimum bound rectangle (MBR) of CC
- Bot(CC) : Bottom of the CC in the MBR of CC
- Left(CC) : Left of the CC in the MBR of CC
- Right(CC) : Right of the CC in the MBR of CC
- AH : Average height of CCs in a page image
- H(CC) : Height of a CC
- CG(CC) : Centre of gravity of CC is average of the row position of PFNs inside the CC.
- P : Highest Peak of histogram of PFNs
- G : Estimated gap between text lines of page image. $G = 2 \times P$
- TLR(CC) : Text line region of a CC. Top, left and right are same as CC bounds, Bottom is *(Top(cc)+AH+G)*
- UH : Under height connect component, if $H(CC) < AH/2$
- OH : Over height connect component, if $H(CC) > TLR(CC)$
- VOV(CC_i, CC_j) : Vertical overlapping of connected components CC_i and CC_j. if $Top(CC_j) \leq Top(CC_i) \leq Bot(CC_j)$ or $Top(CC_i) \leq Top(CC_j) \leq Bot(CC_i)$
- TLROI(PFN) : Region of Influence for text line segmentation. Top and Bottom are same as ROI(PFN). Left extends upto the nearest black pixel in the i^{th} row towards left from j^{th} column, and Right extends upto the nearest black pixel in the i^{th} row towards right from j^{th} column, where i, j are i^{th} row and j^{th} column in the fringe map F.

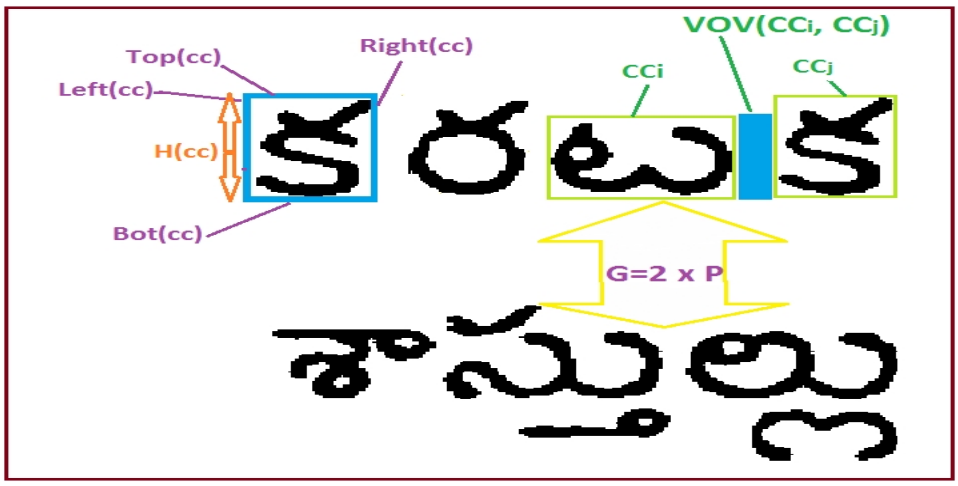


Figure 4.1: Notation in text line segmentation.

4.2.2 Generation of Segmenting Path from PFNs in a Fixed Size Window

We first start with the most elementary concept. This approach we call as the 'static window' approach. Static window based algorithm segments lines in three stages as shown in Fig. 4.2. The first stage generates a fringe map for the given input

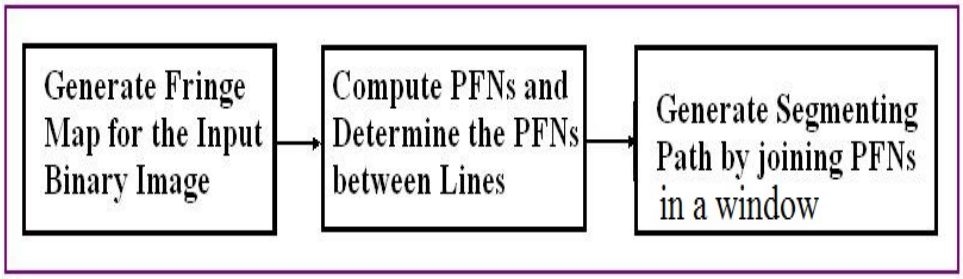


Figure 4.2: Block diagram of the Static window based method.

binary image. In the second stage, Peak fringe numbers (PFNs) are computed in the fringe map. A filtering operation on the PFNs is performed. Then the PFNs between text lines are determined. In the last stage, a segmenting path between lines is generated by joining the PFNs. We give some details about these steps in the

following.

We start with a fringe map generated for the given input binary image. We then scan the fringe map along columns and locate the PFNs by the process in the chapter 2. The PFNs may be present inside the connected component or outside the connected component (between the lines) as shown in Fig. 4.3. We are interested in only the PFNs outside the connected components through which we can find a segmenting path between lines. Generally, we observe that the values of PFNs which are inside the connected components are smaller than the values of PFNs outside the connected components. Therefore we can use this simple observation to separate the PFNs, and distinguish between those PFNs that we need. A simple threshold based method on the PFN value could be used to separate the required PFNs. Here we use the choice of arithmetic mean of all the PFNs to be used as a threshold T. We observe that as expected the PFNs greater than or equal to T are usually found between the text lines. Thus we have filtered out the PFNs, whose fringe number is less than T. The remaining PFNs play an important role in generating segmenting path between lines. Fig. 4.4 shows these PFNs, whose fringe number is greater than or equal to T.

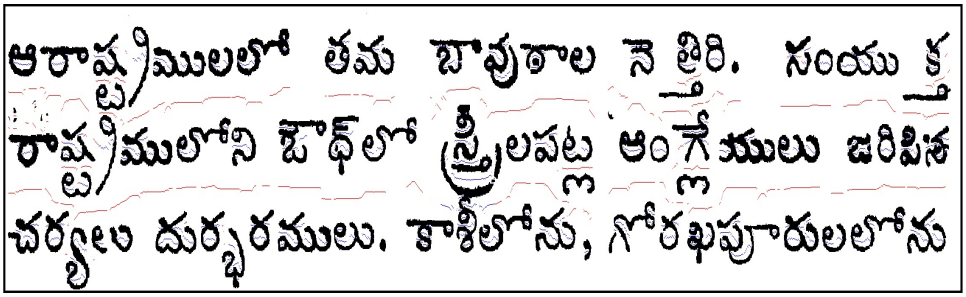


Figure 4.3: PFNs are shown in Color (shaded).

Text Line Segmenting Path Generation

Now a text line segmenting path is found that separates adjacent text lines by joining the PFNs. It is quite possible that the filtering process leaves gaps and we may not be able to join PFNs easily. Prospective points for the segmenting path are

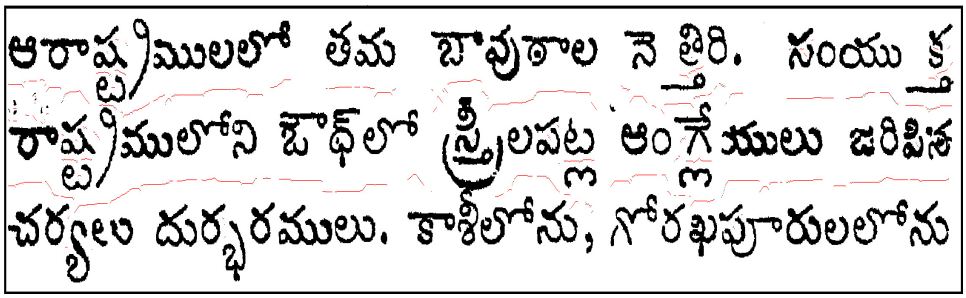


Figure 4.4: The remaining PFNs after filtering are shown.

those PFNs whose fringe value is greater than or equal to the arithmetic mean of the PFNs *M1*. We start the generation of segmenting path from these points. The following procedure explains the search and joining of these PFNs.

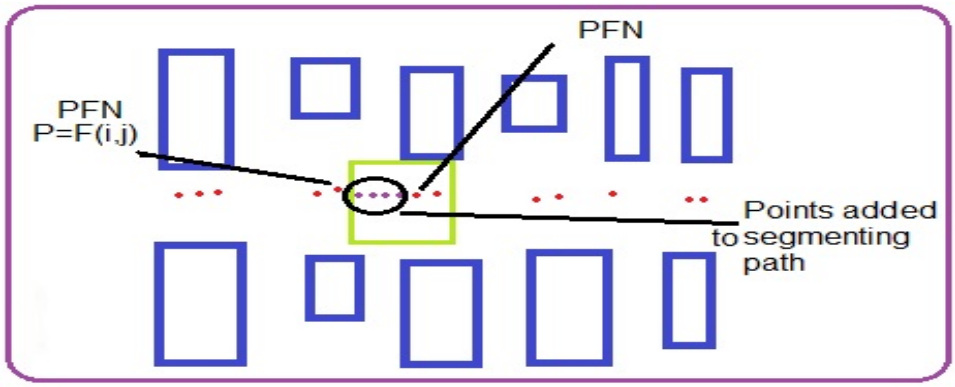


Figure 4.5: Joining of PFNs.

Joining of PFNs and Generating Segmenting Path We scan the fringe map along columns from left to right and search for PFNs with fringe value greater than or equal to *M1* and build a partial segmenting path by joining together the PFNs. The joining of PFNs then is subject to the following considerations within the window. Figure 4.5 shows the joining of PFNs. Consider a PFN point p at *(i,j)*, we search for PFN nearest to it in a square window of size $N \times N$ where $N = 2 \times M1$ with boundaries: Top as *(i - M1)*, Bottom as *(i + M1)*, Left as *(j+1)* and Right as $(j + 2 \times M1)$.

Now we consider the cases where for *j+1* we add points (interpolated) to the segmenting path called as segment points according to:

1. PFN is found at *(m,n)* then we set the local maximum fringe number as a segment point in each column of the window from *(j+1)*th column upto the $(n-1)^{th}$ column.

2. In the extreme situation where no PFN is found in the window, then we set a local maximum fringe number less than or equal to *M1* as a segment point in each column of the window from $(j+1)^{th}$ column to $(j+M1)^{th}$ column.

The above procedure results in isolation of a set of points (segmenting points) along the potential segmenting path. We connect sequentially the two nearest segmenting points and make a partial segmenting path. These are now joined by the process explained here.

Now to join the resulting partial segmenting paths we look backward from right to left. We scan the fringe map along columns from right to left and search for the end points of partial paths to join. We join points such that a point P at i^{th} row and j^{th} column, *P(i,j)* is an end point of a partial path, then we search for a PFN or a segmenting point nearest to it in the window of size *N*. This window is given by $N = 2 \times M1$ with boundaries, Top as *(i - M1)*, Bottom as *(i+ M1)*, Right as *(j-1)* and Left as $(j - 2 \times M1)$. Again two cases may arise:

1. PFN or segment point is found at position *m, n*. Then in each column of the window from $(j-1)^{th}$ column to $(n)^{th}$ column we set as segment points those points whose fringe value is a local maximum.

2. No PFN or segment point is found in the window. Then we set each point with local maximum fringe value less than or equal to *M1* as a segment point in each column of the window from $(j-1)^{th}$ column to $(j - 2 \times M1)^{th}$ column.

The result of path generation is shown in figure 4.6.

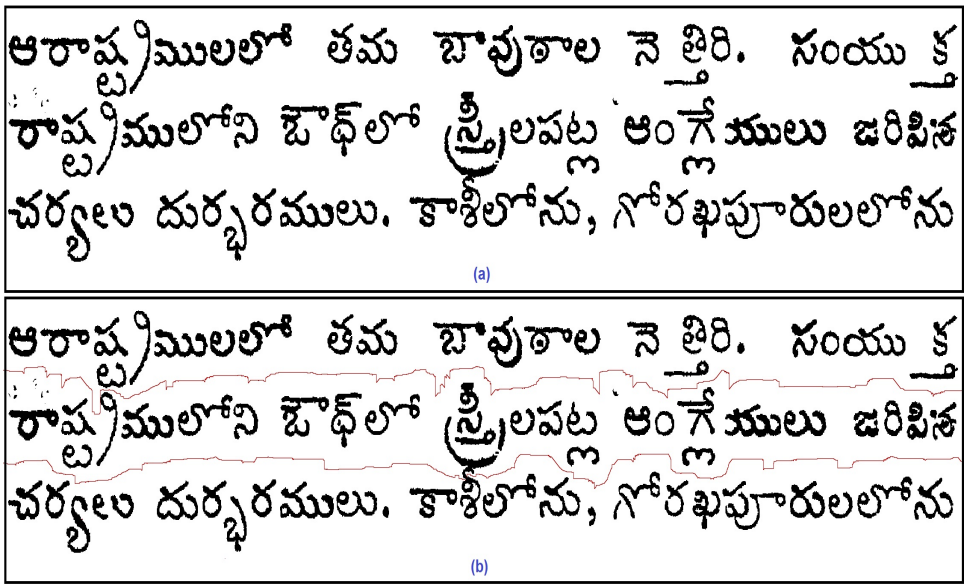


Figure 4.6: Result of segmenting paths generation method using PFNs in a Fixed Size Window (b) for the input image (a).

This method does well for most of the documents, but fails to segment text lines of Telugu script in certain cases where: 1. Variation in spacing between text lines is due to presence of consonant modifiers and vowel modifiers. 2. The filtering operation leaves a huge gap between the PFNs. Then determining which PFNs should belong to the current segmenting path is problematic. 3. The huge space between words misleads the direction of the path.

Figure 4.7 shows the results of a fringe based method for text line segmentation using window based approach. The segmenting path changes direction at the marked box because the space between words is wide and the gap between the lines is narrow. Window size is a parameter than can control the path, and prevent the PFNs of adjacent line from merging. The PFN filtering operation at times leaves a huge gap between PFNs. Thus all of these factors make this method prone to errors in generating an accurate segmenting path. It is possible to know a window size only after experimentation. Therefore this a drawback of the approach.

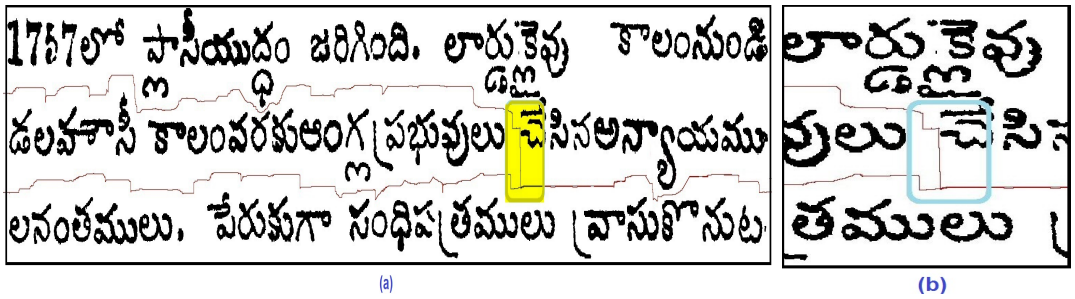


Figure 4.7: (a) Segmentation error result of static window based approach.(b) Zoomed part of selection shown shaded in (a).

4.2.3 Region of Influence & Text and Space Affinity based method

In the previous method we tried to join PFNs to generate segmenting paths. The affinity between PFNs is restricted to a fixed size window. To overcome this limitation we now propose an improved method where first we establish space affinity between the text lines using a region of influence of each PFN (we can call it as a dynamic window). Then the text affinity between the connected components is computed based on the overlapping of CC. After finding the text and space affinity, the text line segments are generated by grouping the components based on the affinity. In the following we explain details of this method and the procedure is shown in algorithm 4.

For a given input binary image, we generate a fringe map and find PFNs in vertical direction. The PFNs located within the CCs are filter as we discussed in the section 3.4.3 that the PFNs enclosed between the pixels of same label, which are now called as Internal peak fringe numbers (IPFNs). These shall be useful to find the centre of gravity of a CC. As we discussed in the section 3.4.3 the remaining PFNs after filtering are between the text lines which is shown in figure 3.14(c). Now we attempt to get more contextual information by observing the PFN histogram. We also wish to relate significant aspects of PFN histogram which has bearing in the line segmentation. We use these PFNs to estimate the gap between the text lines. The histogram of PFNs are shown in figure 3.14 (c). The highest peak of the histogram P is the most

frequent space between the text lines. However this can be taken to estimate the gap $G = 2 \times P$. It is seen that most text line segmentation methods find touching components to be obstacles in finding the text segmenting path when proceeding from left to right (or vice versa). Even smaller sized components could disturb the potential segmenting path. To mark these components, we compute the average height AH of the CCs. We consider the height of a CC, H(CC) which is less than half of AH. These CCs are marked as `under height` CC. Similarly for a CC if H(CC) is greater than AH+G, then it is marked as `over height` CC. In a top down manner the overall approach is presented in Algorithm 4. The details of this are presented in Algorithm 2 & 3. In algorithm 4 the above described procedure is performed from step 1 to step 4.

Associate PFNs to CC

However to the find space affinity between adjacent text lines we use the PFNs between them. Here we aim to group the PFNs that lie between the two adjacent lines. This is difficult at times due to large separation between the PFNs. Gaps may arise at locations due to filtering operation and non existence of PFNs in vertical direction. In some cases PFNs may have affinity between two different adjacent lines due to huge word and space variations. Grouping of PFNs from two different adjacent lines leads to an incorrect segmentation as shown in Figure 4.8. Therefore, to establish better affinity between PFNs we associate the PFNs to each CC. To do this, we consider a region for each CC. This region is defined as the text line region of that CC, TLR(CC). The region extends from the Top(CC) to the bottom with a distance of (AH + G). All the PFNs within the TLR(CC) are associated to that CC. If the PFNs do not exist in the region then the fringe value located at (TOP(CC)+AH+P) from left to right of a CC are assumed as PFNs. In Algorithm 4 from step 5 to step 8 it locates the PFNs and associate the PFNs to the CC.

Region of Influence of PFN for Text Line TLROI(CC) Formation

As we discussed in the section 3.4.3 ROI of a PFN is extended isotropically about it. Here we extend and adapt the ROI of a PFN in a different manner so as to

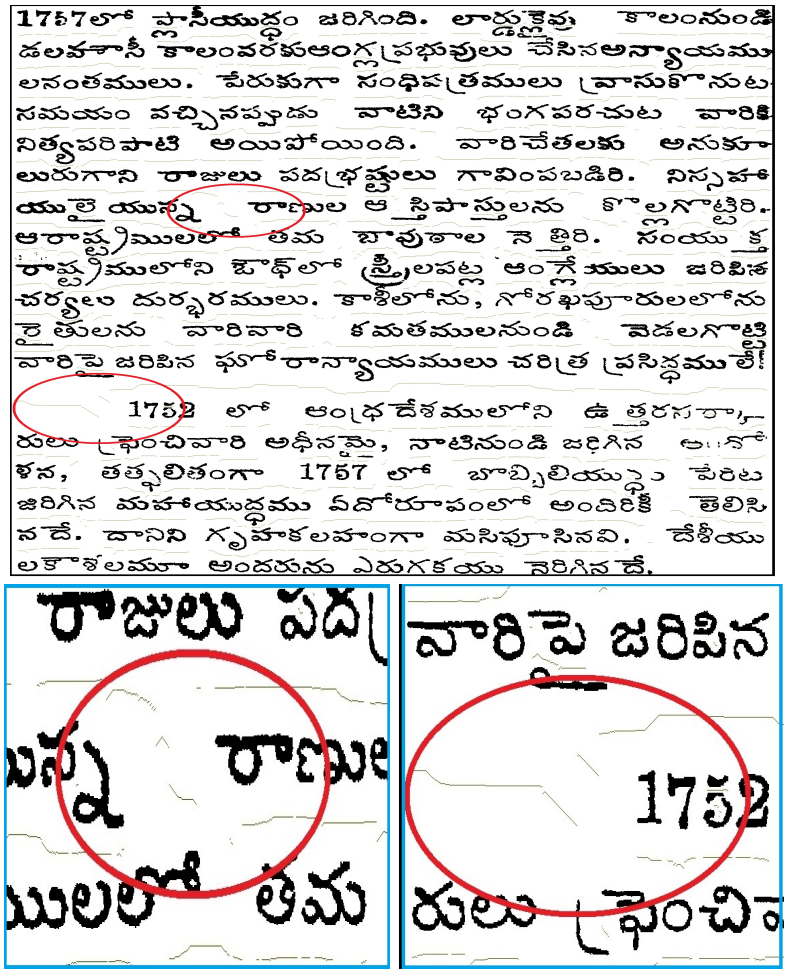


Figure 4.8: PFNs having affinity between two different lines are shown in oval ,oval portions are zoomed.

improve the line segmentation. Here however this ROI needs to be extended more in horizontal direction for linking PFNs. Therefore we call it as text line region of influence of a PFN or TLROI(PFN). Let X=*F(i,j)* where X is the fringe value and i, j are i^{th} row and j^{th} column in the fringe map F. Then boundaries of TLROI(X) are defined as *Top bound* as $(i - X)$ and *Bottom bound* is $(i + X)$, both are same as in ROI(PFN). *Left bound* extends upto the nearest black pixel (zero fringe value) in the i^{th} row towards left from j^{th} column, and *Right bound* extends upto the nearest black pixel (zero fringe value) in the i^{th} row towards right from j^{th} column. Figure 4.9 shows the TLROI(X) graphically.

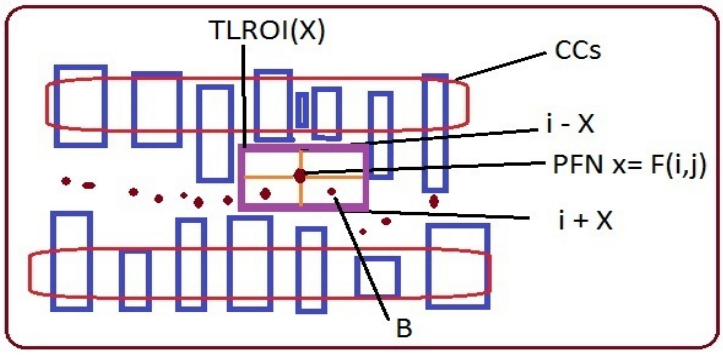


Figure 4.9: Text Line Region of Influence of PFN X at location *F(i,j)*, CCs are in rectangle and dots are PFNs.

Establish Affinity between PFNs

Now we use the TLROI(PFN) to establish the space affinity between PFNs of CCs. Let CC_i and CC_j be the two CCs and PFN_i and PFN_j be the PFNs of i^{th} and j^{th} CC. If (($PFN_i \subset TLROI(PFN_j)$) or ($PFN_j \subset TLROI(PFN_i)$)) then the two PFNs are having space affinity. Figure 4.9 shows the PFN X and its TLROI in the rectangle. The PFN B is within the region of TLROI(X), therefore the PFN X and B have affinity.

Construction of Text Line Segment TSL

Here we construct text line segments using text and space affinity. The text lines build with both affinities are more accurate then with only one affinity. The CCs with vertical overlapping and its TLROI(PFNs) that intersect are grouped to form text line segments. Let CC_i and CC_j be the two CCs and PFN_i and PFN_j be the PFNs of i^{th} and j^{th} CC. Then two components are joined if ($PFN_i \subset TLROI(PFN_j)$ or $PFN_j \subset TLROI(PFN_i)$) **and** VOV($CC_i$, CC_j). Figure 4.10 shows the result of text line segments. Along with this, we compute text line segment parameters. These parameters are used to extend text line segment to image boundaries if they are partial and assign remaining CCs to text lines.

Computing Text Line Segment Parameters

1. The path of centered moving average of PFNs of Text line segment MAP(TLS) is generated from left to right, this acts as a tentative segmenting path between adjacent text lines. Computing the path starts from the left most CC's PFN of a TLS to the right. Window is taken $2 \times AH$ size on either sides of the point. Centered Moving Average of $PFN_i = \frac{PFN_{i-2\times AH}+...+PFN_{i+2\times AH}}{4\times AH+1}$

2. The average PFN value of Text line segment *APFN(TLS)* is used in determining the overlapping and touching components.

3. The centre of gravity of text line segment *CG(TLS)* is found using the IPFNs which are filtered in the above steps. *CG(TLS)* is defined as a fictitious line passing through the 'centre' of the text line. It is also computed similarly as a centered moving average of PFNs but here they are IPFNs (PFNs inside the CCs).

4. Boundaries of a Text line segment is found as Left bound of TLS is taken as left bound of the left most CC in the TLS, similarly Right bound of TLS is right bound of the right most CC in the TLS, Bottom is the MAP(TLS) and Top is taken as the 2 × AD from the MAP(TLS) to the above where AD is average distance between CG(TLS) and MAP(TLS).

Algorithm 2, TLSP finds text line segments and its parameters. In Step 1 to Step 15 the algorithm finds text line segments. Initialization is done by adding the component CC_1 to the queue Q and list T where Q and T are a temporary list. To flag the assignment of CC_1 to a line we set Assign[1]=1. Then from Step 3 to Step 9 each CC from Q and is picked and matched for the space and text affinity with other CCs which are not in Q. If the CC_i component has the affinity then CC_i is added to Q, and T, and also Assign[i]=1. This is repeated until Q is empty, then the CCs in T are stored in TLS_k where TLS_k is k^{th} text line. Step 17 to 26 computes parameters of each TLS_i by varying i from 1 to k. In Step 18 and Step 19 finds left and right boundaries of text line segment. In step 20 j is varied from left to right boundary of a TLS_i to find the parameters MAP, CG, AD and APFN. MAP

line of TLS_i is taken as the bottom bound. The points above the bottom bound of TLS_i with a distance $2 \times$ AD from left to right of TLS_i is taken as top bound of TLS_i.

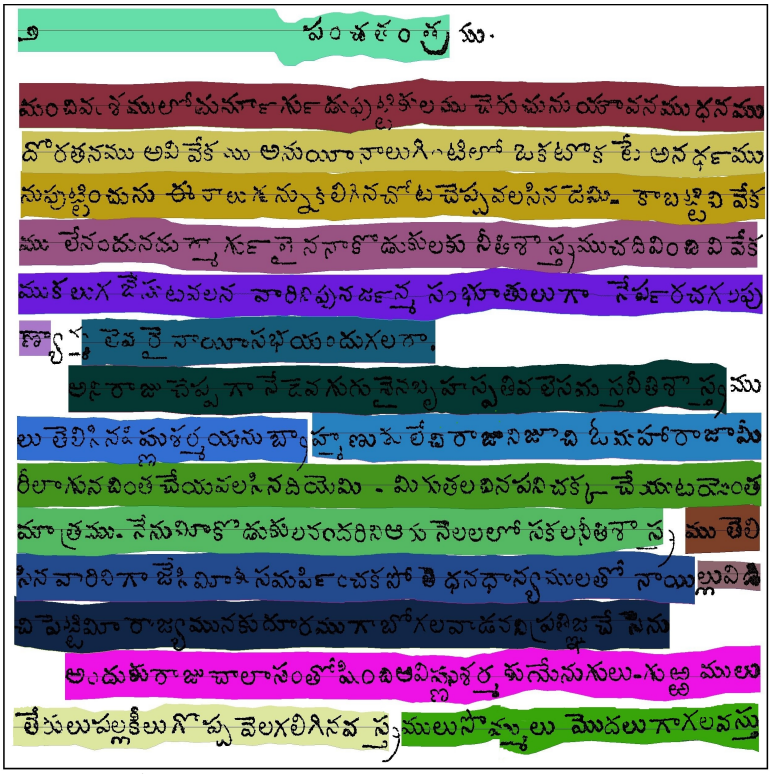


Figure 4.10: Text Line Segments.

The result in Figure 4.10 shows that some text line segments do not extend fully to the width of the page. Here we find the affinity between text line segments and merge them. Let TLS_i and TLS_j be the two TLSs. Then two text line segments are joined if (Top(TLS_j) $\leq CG(TLS_i) \leq Bottom(TLS_j)$ and Top(TLS_i) $\leq CG(TLS_j) \leq Bottom(TLS_i)$). After merging, the text line segments are checked to see whether they extend up to the image boundary limit (width of page). For those text line segments which are not extending up to the image boundaries, MAP(TLS_i) is extended horizontally. Now the MAP(TLS_i) is taken as the segmenting path. The text line segment parameters are then recomputed for these text lines.

The following is about Algorithm 3. Here it works upon the TLS found by algorithm

2. Some components are unassigned due to lack of both affinities and some may be touching or were not considered in algorithm 2 due their height criterion not being satisfied. The connected components are assigned with Algorithm 3 (CC-Assign) to the text lines and labelled with line number. Here we get two cases, one is the CC is totally included in TLS boundaries and other is when CC overlaps of two or more TLSs. In case 1 the CC is assigned to the TLS_i and marked with label i. Where as in case 2 it needs to be determined whether it is a single component overlapping with neighboring TLS region or touching with two or more CCs. This is determined with the heuristics in Step 7, 10 and 13. If it satisfies the Step 7 then the CC belongs to the upper text line. Similarly if it satisfies Step 10 then it belongs to the lower text line. Figure 4.11(a) shows the CC (circle with blue color) is assigned to the upper text lines and the CC (circle with yellow color) is assigned to the lower text line. If Step 13 is satisfied then the CC is taken as a case of touching of CCs. Therefore the CC is cut at the intersecting point with MAP(TLS). The part from TOP(CC) to Intersecting point is assigned to upper text line. The remaining part is processed again with algorithm 3. Figure 4.11(b) shows the touching CCs (blue color within the circle).

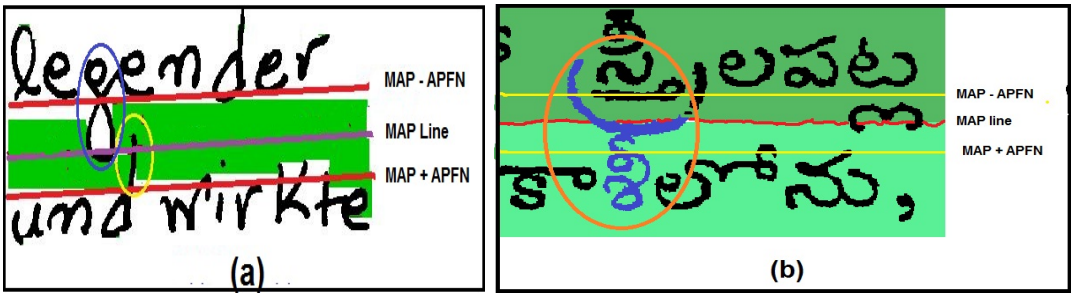


Figure 4.11: Overlapping and Touching CC assignment. Blue circled CC is assigned to upper text line in (a), yellow circled CC is assigned to lower text line in (a) and (b) is touching component.

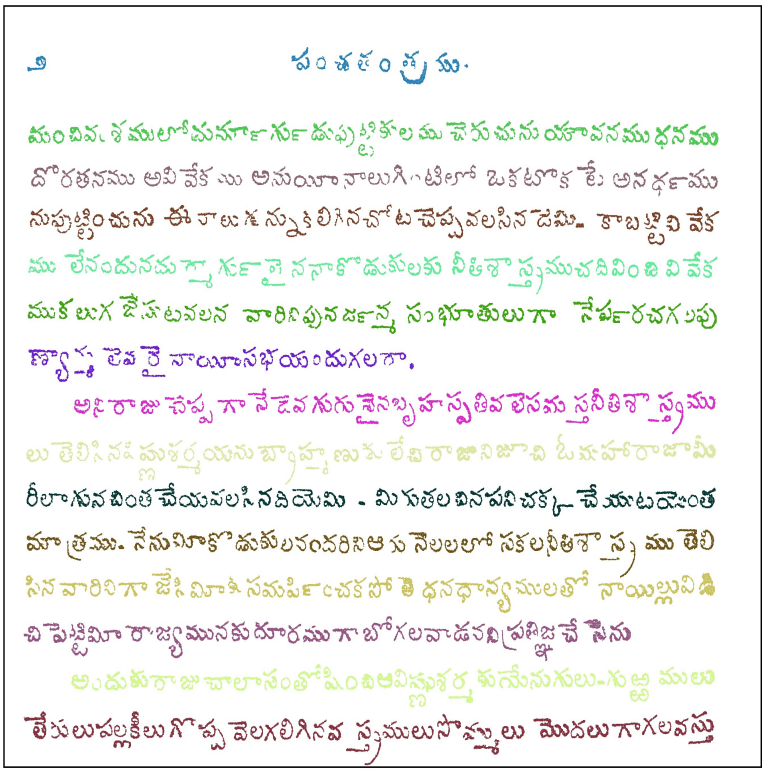


Figure 4.12: Successful text line segmentation of Text and Space affinity based algorithm using PFNs.

4.3 Evaluation of Segmentation methods

Traditionally in machine printed document analysis, the text line segmentation results are represented as rectangular bounding boxes. Therefore, the evaluation is often based on the four coordinates of a bounding box. However, some overlaps inevitably occur among bounding boxes in printed Telugu text documents. Non overlapping closed curves are better in the sense of representing printed text lines. By this representation, the evaluation can be done at the pixel level, which is more accurate than that done at the bounding box level [62].

The method used to evaluate the performance of the submitted algorithms is based on counting the number of matches between the entities detected by the algorithm and the entities in the ground truth [48]. For the detection of matches, we used a

Algorithm 2: TLSP:Text line segments and parameters algorithm

Input : Fringe Map, CCs with PFNs and IPFNs

Output: Text Line segment boundaries and parameters

`/* TLS`$_i$ `is ith text line segment, T is the temp, Q is queue and CC`$_i$`, `$1 \leq i \leq n$`, n is number of connected components. */`

Initialize k=1,Assign[1:n]=0, Add CC_k to T, Add CC_k to Q and Assign[1]=1

repeat

  repeat

    CC_j = delete CC from Q

    for *i = 1 to n* **do**

      if *($i \neq j$) and (Assign[i] =0)* **then**

        if *($PFN_i \subset TLROI(PFN_j)$ or $PFN_j \subset TLROI(PFN_i)$) **and** $VOV(CC_i, CC_j)$* **then**

          Add CC_i to T, Add CC_i to Q and Assign[i]=1

  until *Q is empty*

  if *T contains more than one CC* **then**

    TLS_k =T, empty T, Increment k

  for *p = 1 to n* **do**

    if *(Assign[p] =0)* **then**

      Add CC_p to T, Add CC_p to Q, Assign[p]=1 and Break

until *Q is empty*

decrement k `/* k number of Text line segments */`

for *i = 1 to k* **do**

  Left(TLS_i) = Left(leftmost CC in TLS_i)

  Right(TLS_i) = Right(rightmost CC in TLS_i)

  for *j = Left(TLS_i) to Right(TLS_i)* **do**

    MAP(j)= Average PFNs row value of TLS_i in a window `/* window is from` $j - 2 \times AH$ `to` $j + 2 \times AH$ `*/`

    CG(j)= Average IPFNs row value of TLS_i in a window

    AD=AD+MAP(j)- CG(j); APFN=APFN+MAP(j)

  AD(TLS_i)=AD ÷ (Right(TLS_i)- Left(TLS_i))

  APFN(TLS_i)=APFN ÷(Right(TLS_i)-Left(TLS_i))

  Bottom(TLS_i)=MAP(TLS_i); Top(TLS_i)=MAP(TLS_i)- 2 × AD

Algorithm 3: CC-Assign:Assignment of CCs to TLSs algorithm

Input : CCs and TLS and parameters

Output: Components are assigned to text line and labelled

`/* The connected components are assigned to the Text lines and label with line number. */`

```
1  for i = 1 to n do
2      for j= 1 to k do
3          if CC_i ⊂ TLS_j then
4              label CC_i with j
5          else
6              if CC_i intersect MAP(TLS_j) then
7                  if Bottom(CC_i) ≤ (MAP(TLS_j) + APFN(TLS_j)) then
8                      label CC_i with j
9                  else
10                     if Top(CC_i) > (MAP(TLS_j) − APFN(TLS_j)) and
                       Bottom(CC_i) > (MAP(TLS_j)+ APFN(TLS_j)) then
11                         then label CC_i with j+1
12                     else
13                         if Top(CC_i) < (MAP(TLS_j) − APFN(TLS_j)) and
                           Bottom(CC_i) >(MAP(TLS_j)+ APFN(TLS_j)) then
14                             cut the component CC_i at the intersection of
                               MAP(TLS_j) and label the component from
                               TOP(CC_i) to intersection point of CC_i with j.
                               Then check the remaining component similarly
                               with TLS_{j+1}.
```

Algorithm 4: Affinity based Text line segmentation algorithm using PFN

Input : A binary Image

Output: Text Line Image: Components of a text line are unique labelled

1 Generate fringe map F for the input binary image I

2 Compute PFNs in vertical direction in the fringe map

3 Filter the PFNs inside the components

4 Find CCs and label with unique number. Then Mark Over height (OH) and Under height (UH) CCs

5 Associate PFNs to each CC.

6 **forall the** *CC* **do**

7 `find PFNs within the TLR(CC)`

8 `If PFN does not exist then mark the fringe numbers located at (TOP(CC) + AH + P) in TLR(CC) as PFNs of CC.`

9 Call TLSP algorithm to find text line segments and parameters

10 Merge overlapping Text line segments. Let TLS_i and TLS_j be the two TLSs. Then two text line segments are joined if
$(\text{Top}(\text{TLS}_j) \leq CG(TLS_i) \leq Bottom(TLS_j)$ and
$\text{Top}(\text{TLS}_i) \leq CG(TLS_j) \leq Bottom(TLS_i))$

11 TLSs are extended horizontally up to Image bounds by extending the MAP(TLS). Then each TLS is assumed as text line.

12 Call CC-Assign algorithm to assign CCs to text lines

MatchScore table whose values are calculated according to the intersection of the ON pixel sets of the result and the ground truth.
Let I be the set of all image points, G_j the set of all points inside the j ground truth region, R_i the set of all points inside the i result region, T(s) a function that counts the points of set s. Table MatchScore(i,j) represents the matching results of the j ground truth region and the i result region:

$$MatchScore(i,j) = \frac{T(G_j \cap R_i \cap I)}{T((G_j \cup R_i) \cap I)} \tag{4.1}$$

A region pair is considered as a one-to-one match only if the matching score is equal to or above the evaluator's acceptance threshold T_a. Let N be the count of ground-truth elements, M be the count of result elements, and o2o be the number of one-to-one matches, the detection rate (DR) and recognition accuracy (RA) are defined as follows:

$$DR = \frac{o2o}{N}, RA = \frac{o2o}{M} \tag{4.2}$$

A final performance metric F-Measure (FM) can be extracted if we combine the values of detection rate (DR) and recognition accuracy (RA):

$$FM = \frac{2(DR)(RA)}{DR + RA} \tag{4.3}$$

The performance evaluation method is robust since it has been used in the contests (ICDAR 2007, ICDAR 2009, ICDAR 2013 and ICFHR 2010 Handwritten Segmentation) [62] and it depends only on the selection of the acceptance threshold T_a.

4.4 Experimentation & Performance Analysis

We evaluated the performance of projection profile(PP), Adaptive RLSA (ARLSA) [40], Piece-wise Projection (PAL) [45] and Proposed PFN Affinity based algorithms 4 for text line segmentation using equations 4.1, 4.2 and 4.3. The test datasets are collected from DLI and University of Hyderabad OCR lab under the grant No. 14(6)/2006-HCC(TDIL), Ministry for Communications and Information Technology (MCIT), New Delhi, Government of India.

Generation of Pixel Level Ground Truth

Despite there existing sources for images and a Telugu text image corpus no ground truth exists for these images. Therefore for the purpose of evaluation the pixel level ground truth was required to be created. Before this work no existing ground truth data was present for Telugu printed script images.

Total 10 books with 1025 page images. The number of text lines for all 1025 document images was 29875. These documents contain overlapping components, Touching components, variation in space, Text skew, document skew different font size and poetry documents. We experimented with different thresholds Ta=100%, 95%, 90% and 80% for acceptance of segmented text line. 100% accuracy mean the pixels detected by the algorithm is fully matched against ground truth. Performance of different methods on Telugu documents mentioned is shown in Tables 4.1, 4.2, 4.3 and 4.4. The overall performance of methods is shown in Figure 4.13

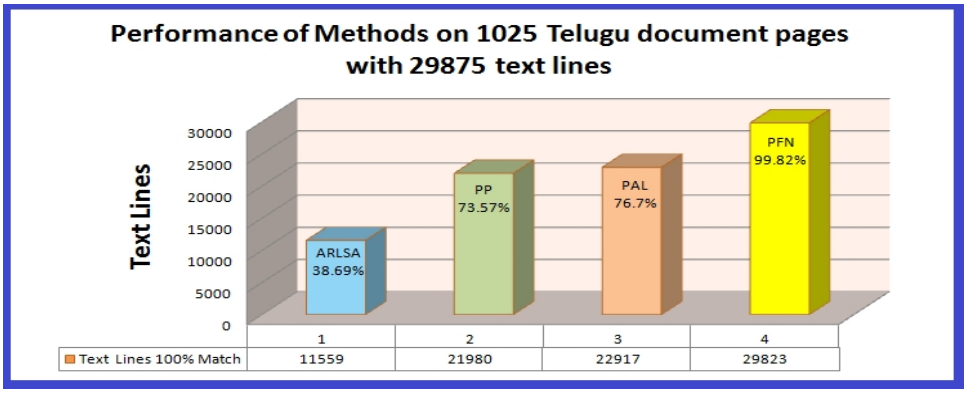


Figure 4.13: Overall performance of Methods on 1025 Telugu document pages.

Table 4.1: Text Line Segmentation performance of different methods on Telugu documents with Threshold $Ta = 100\%$ (100% accuracy against Ground Truth)

$Ta = 100\%$								
Book Name	**Number of Pages**	**Actual Lines**	**Method**	**Detected Lines**	**one2one**	**Detected rate**	**Recognition accuracy**	**Pixel level hit ratio (FM)**
Aashya-padham	122	3601	ARLSA	5524	1328	0.368	0.24	0.291
			PP	4238	2713	0.753	0.64	0.692
			PAL	3907	2916	0.81	0.746	0.776
			PFN	**3601**	**3601**	**1**	**1**	**1**
Rambabu Diary Part-l	73	2068	ARLSA	3510	811	0.392	0.231	0.29
			PP	2212	1892	0.915	0.855	0.884
			PAL	2166	1936	0.936	0.893	0.914
			PFN	**2068**	**2063**	**0.998**	**0.998**	**0.998**
Rekkalu	69	2267	ARLSA	3310	388	0.171	0.12	0.139
			PP	2586	1919	0.846	0.742	0.79
			PAL	2497	2011	0.887	0.805	0.844
			PFN	**2267**	**2241**	**0.989**	**0.989**	**0.989**
PelliDani Puttupoor votharalu Raja	76	1937	ARLSA	3154	832	0.429	0.263	0.326
			PP	1205	548	0.282	0.454	0.348
			PAL	1398	678	0.35	0.485	0.406
			PFN	**1937**	**1937**	**1**	**1**	**1**
Raja sekhara Charitra	39	1141	ARLSA	1922	365	0.319	0.189	0.238
			PP	1191	1082	0.948	0.908	0.927
			PAL	1223	1101	0.965	0.9	0.931
			PFN	**1141**	**1139**	**0.998**	**0.998**	**0.998**
Gurajada Rachanalu Kathanikalu	31	922	ARLSA	1433	276	0.299	0.192	0.234
			PP	994	832	0.902	0.837	0.868
			PAL	986	843	0.914	0.854	0.883
			PFN	**922**	**915**	**0.993**	**0.993**	**0.993**
Ganga jaatara	45	1452	ARLSA	2265	453	0.311	0.201	0.243
			PP	1573	1262	0.869	0.802	0.834
			PAL	1601	1296	0.892	0.809	0.849
			PFN	**1452**	**1446**	**0.995**	**0.995**	**0.995**
Panduranga Mahathyam	168	4812	ARLSA	7722	2035	0.423	0.263	0.324
			PP	5717	3816	0.793	0.667	0.724
			PAL	5819	3916	0.813	0.673	0.736
			PFN	**4812**	**4808**	**0.999**	**0.999**	**0.999**
Gurajada Rachanalu Kavithala Samputam	121	3339	ARLSA	3300	1406	0.421	0.426	0.423
			PP	5253	2023	0.605	0.385	0.471
			PAL	4356	2124	0.636	0.487	0.552
			PFN	**3339**	**3339**	**1**	**1**	**1**
Srungara Nyshadam	281	8336	ARLSA	11554	3665	0.439	0.317	0.368
			PP	11009	5893	0.706	0.535	0.609
			PAL	10865	6096	0.731	0.561	0.635
			PFN	**8336**	**8334**	**0.999**	**0.999**	**0.999**

Table 4.2: Text Line Segmentation performance of different methods on Telugu documents with Threshold $Ta = 95\%$ (95% accuracy against Ground Truth)

$Ta = 95\%$								
Book Name	**Number of Pages**	**Actual Lines**	**Method**	**Detected Lines**	**one2one**	**Detected rate**	**Recognition accuracy**	**Pixel level hit ratio (FM)**
Aashya-padham	122	3601	ARLSA	5524	2972	0.825	0.538	0.651
			PP	4238	3364	0.934	0.793	0.858
			PAL	3907	3405	0.945	0.871	0.907
			PFN	**3601**	**3601**	**1**	**1**	**1**
Rambabu Diary Part-l	73	2068	ARLSA	3510	1829	0.884	0.521	0.655
			PP	2212	2004	0.969	0.905	0.936
			PAL	2166	2046	0.989	0.944	0.966
			PFN	**2068**	**2063**	**0.998**	**0.998**	**0.998**
Rekkalu	69	2267	ARLSA	3310	1405	0.619	0.424	0.503
			PP	2586	2160	0.952	0.835	0.89
			PAL	2497	2178	0.96	0.872	0.914
			PFN	**2267**	**2241**	**0.989**	**0.989**	**0.989**
PelliDani Puttupoor votharalu Raja	76	1937	ARLSA	3154	1583	0.817	0.502	0.621
			PP	1205	598	0.308	0.496	0.38
			PAL	1398	896	0.462	0.64	0.537
			PFN	**1937**	**1937**	**1**	**1**	**1**
Raja sekhara Charitra	39	1141	ARLSA	1922	917	0.803	0.477	0.5981
			PP	1191	1128	0.988	0.947	0.967
			PAL	1223	1132	0.992	0.925	0.957
			PFN	**1141**	**1139**	**0.998**	**0.998**	**0.998**
Gurajada Rachanalu Kathanikalu	31	922	ARLSA	1433	719	0.779	0.501	0.61
			PP	994	873	0.946	0.878	0.911
			PAL	986	879	0.953	0.891	0.921
			PFN	**922**	**915**	**0.992**	**0.992**	**0.992**
Ganga jaatara	45	1452	ARLSA	2265	1145	0.788	0.505	0.616
			PP	1573	1408	0.969	0.895	0.931
			PAL	1601	1414	0.973	0.883	0.926
			PFN	**1452**	**1446**	**0.996**	**0.996**	**0.996**
Panduranga Mahathyam	168	4812	ARLSA	7722	3691	0.767	0.478	0.588
			PP	5717	4697	0.976	0.821	0.892
			PAL	5819	4703	0.977	0.808	0.884
			PFN	**4812**	**4808**	**0.999**	**0.999**	**0.999**
Gurajada Rachanalu Kavithala Samputam	121	3339	ARLSA	3300	2227	0.667	0.675	0.670
			PP	5253	3011	0.901	0.573	0.701
			PAL	4356	3099	0.928	0.711	0.805
			PFN	**3339**	**3339**	**1**	**1**	**1**
Srungara Nyshadam	281	8336	ARLSA	11554	6317	0.757	0.546	0.635
			PP	11009	8070	0.968	0.733	0.834
			PAL	10865	8112	0.973	0.746	0.845
			PFN	**8336**	**8334**	**0.999**	**0.999**	**0.999**

Table 4.3: Text Line Segmentation performance of different methods on Telugu documents with Threshold $Ta = 90\%$ (90% accuracy against Ground Truth)

$Ta = 90\%$								
Book Name	**Number of Pages**	**Actual Lines**	**Method**	**Detected Lines**	**one2one**	**Detected rate**	**Recognition accuracy**	**Pixel level hit ratio (FM)**
Aashya-padham	122	3601	ARLSA	5524	3093	0.858	0.559	0.678
			PP	4238	3380	0.938	0.797	0.862
			PAL	3907	3422	0.95	0.875	0.911
			PFN	**3601**	**3601**	**1**	**1**	**1**
Rambabu Diary Part-l	73	2068	ARLSA	3510	1894	0.915	0.539	0.679
			PP	2212	2043	0.988	0.923	0.954
			PAL	2166	2046	0.989	0.944	0.966
			PFN	**2068**	**2063**	**0.998**	**0.998**	**0.998**
Rekkalu	69	2267	ARLSA	3310	1526	0.673	0.461	0.547
			PP	2586	2205	0.972	0.852	0.908
			PAL	2497	2212	0.975	0.885	0.928
			PFN	**2267**	**2241**	**0.989**	**0.989**	**0.989**
PelliDani Puttupoor votharalu Raja	76	1937	ARLSA	3154	1702	0.878	0.539	0.668
			PP	1205	614	0.317	0.509	0.39
			PAL	1398	1024	0.528	0.731	0.613
			PFN	**1937**	**1937**	**1**	**1**	**1**
Raja sekhara Charitra	39	1141	ARLSA	1922	944	0.827	0.491	0.616
			PP	1191	1134	0.993	0.952	0.972
			PAL	1223	1135	0.994	0.928	0.96
			PFN	**1141**	*1139*	**0.998**	**0.998**	**0.998**
Gurajada Rachanalu Kathanikalu	31	922	ARLSA	1433	744	0.806	0.519	0.632
			PP	994	879	0.953	0.884	0.917
			PAL	986	879	0.953	0.891	0.921
			PFN	**922**	**915**	**0.993**	**0.993**	**0.993**
Ganga jaatara	45	1452	ARLSA	2265	1201	0.827	0.53	0.646
			PP	1573	1425	0.981	0.905	0.942
			PAL	1601	1436	0.988	0.896	0.94
			PFN	**1452**	**1446**	**0.995**	**0.995**	**0.995**
Panduranga Mahathyam	168	4812	ARLSA	7722	4155	0.863	0.538	0.663
			PP	5717	4755	0.988	0.831	0.903
			PAL	5819	4770	0.991	0.819	0.897
			PFN	**4812**	**4808**	**0.999**	**0.999**	**0.999**
Gurajada Rachanalu Kavithala Samputam	121	3339	ARLSA	3300	2389	0.715	0.723	0.719
			PP	5253	3134	0.938	0.596	0.729
			PAL	4356	3156	0.945	0.724	0.82
			PFN	**3339**	**3339**	**1**	**1**	**1**
Srungara Nyshadam	281	8336	ARLSA	11554	7034	0.843	0.608	0.707
			PP	11009	8122	0.974	0.737	0.839
			PAL	10865	8134	0.975	0.748	0.847
			PFN	**8336**	**8334**	**0.999**	**0.999**	**0.999**

Table 4.4: Text Line Segmentation performance of different methods on Telugu documents with Threshold $Ta = 80\%$ (80% accuracy against Ground Truth)

$Ta = 80\%$								
Book Name	**Number of Pages**	**Actual Lines**	**Method**	**Detected Lines**	**one2one**	**Detected rate**	**Recognition accuracy**	**Pixel level hit ratio (FM)**
Aashya-padham	122	3601	ARLSA	5524	3210	0.891	0.581	0.703
			PP	4238	3387	0.94	0.799	0.864
			PAL	3907	3488	0.968	0.892	0.929
			PFN	**3601**	**3601**	**1**	**1**	**1**
Rambabu Diary Part-l	73	2068	ARLSA	3510	1923	0.929	0.547	0.689
			PP	2212	2054	0.993	0.928	0.959
			PAL	2166	2056	0.994	0.949	0.971
			PFN	**2068**	**2063**	**0.998**	**0.998**	**0.998**
Rekkalu	69	2267	ARLSA	3310	1597	0.704	0.482	0.572
			PP	2586	2217	0.977	0.857	0.913
			PAL	2497	2221	0.979	0.889	0.932
			PFN	**2267**	**2241**	**0.989**	**0.989**	**0.989**
PelliDani Puttupoor votharalu Raja	76	1937	ARLSA	3154	1709	0.882	0.541	0.671
			PP	1205	618	0.319	0.512	0.393
			PAL	1398	1303	0.672	0.932	0.781
			PFN	**1937**	**1937**	**1**	**1**	**1**
Raja sekhara Charitra	39	1141	ARLSA	1922	965	0.845	0.502	0.63
			PP	1191	1135	0.994	0.953	0.973
			PAL	1223	1135	0.994	0.928	0.96
			PFN	**1141**	**1139**	**0.998**	**0.998**	**0.998**
Gurajada Rachanalu Kathanikalu	31	922	ARLSA	1433	759	0.823	0.529	0.644
			PP	994	887	0.962	0.892	0.925
			PAL	986	891	0.966	0.903	0.933
			PFN	**922**	**915**	**0.993**	**0.993**	**0.993**
Ganga jaatara	45	1452	ARLSA	2265	1230	0.847	0.543	0.661
			PP	1573	1430	0.984	0.909	0.945
			PAL	1601	1446	0.995	0.903	0.947
			PFN	**1452**	**1446**	**0.995**	**0.995**	**0.995**
Panduranga Mahathyam	168	4812	ARLSA	7722	4542	0.943	0.588	0.724
			PP	5717	4759	0.989	0.832	0.903
			PAL	5819	4781	0.993	0.821	0.899
			PFN	**4812**	**4808**	**0.999**	**0.999**	**0.999**
Gurajada Rachanalu Kavithala Samputam	121	3339	ARLSA	3300	2444	0.731	0.74	0.736
			PP	5253	3176	0.951	0.604	0.739
			PAL	4356	3214	0.962	0.737	0.835
			PFN	**3339**	**3339**	**1**	**1**	**1**
Srungara Nyshadam	281	8336	ARLSA	11554	7600	0.911	0.657	0.764
			PP	11009	8142	0.976	0.739	0.841
			PAL	10865	8221	0.986	0.756	0.856
			PFN	**8336**	**8334**	**0.999**	**0.999**	**0.999**

4.5 Conclusion

The PFN based approaches were refined successively for more accuracy and robustness. First was the static window approach. It was improved by the region of influence approach. Affinity concepts were quite useful in formulating the region of influence approach. The approaches which determines affinity are seen to do better. PFN methods are proving to be better than other approaches. These methods are script independent in nature.

Chapter 5

Expanding the Scope of Fringe based Text Segmentation

Previously we established the utility of PFNs for text line segmentation. Further algorithms were presented that were based on affinity with PFN as tool. Here we show that these concepts are generic and extensible. We show here the extensions to word segmentation, baseline detection and character segmentation.

5.1 Fringe based Approach for Word segmentation

Word Segmentation is a process of dividing text line image into word. In another way aggregation of character components of a word image. Most of the word segmentation approaches use the gap between the components for segmentation. The main assumption is that the gaps between words are greater than the gaps between characters. U.V Marti [34] label the CC and computed their convex hull. Then the Euclidean distances between the convex hulls are calculated for the classification of inter word and intra word gaps. C.V. Lakshmi [26] and Priyanka et al [49] are used vertical projection profile for computing gaps. Then they used height of the text line for differentiating the word and character gap. Koppula et al [22] also computed gaps using vertical projection profiles. Threshold for separating gaps is estimated using clustering technique.

Here we proposed a method using PFNs. It contains the following steps:

1. Generate Fringe map for the input binary image.
2. Compute PFNs in horizontal direction.
3. Filter PFNs inside CCs.
4. Estimate the gaps between characters and aggregate the connected components of a word.

The word segmentation procedure of the proposed method is applied independently to each text line detected from the text line segmentation algorithm. However the gap statistics are collected from the page image of that text line. We generate fringe map for the input binary and compute PFNs horizontally as described in the Chapter 3. To filter the PFNs inside the CCs. First label the CCs in page image then the PFNs between the pixels of same label are removed. After filtering the remaining PFNs are between the CCs. We compute the weight average of PFN values Avg and height peak P of histogram of PFNs. Then we consider the threshold $T = (Avg + P)/2$ as the gap between characters. To aggregate the components of a word the white pixels surrounded by the components are smeared whose fringe value is less than or equal to T. Results of the proposed word segmentation method is shown in figure 5.1, 5.2, 5.3 & 5.4. It must be noted that these examples shown vary in script and the second example is a difficult one with varying skew in a handwritten document.

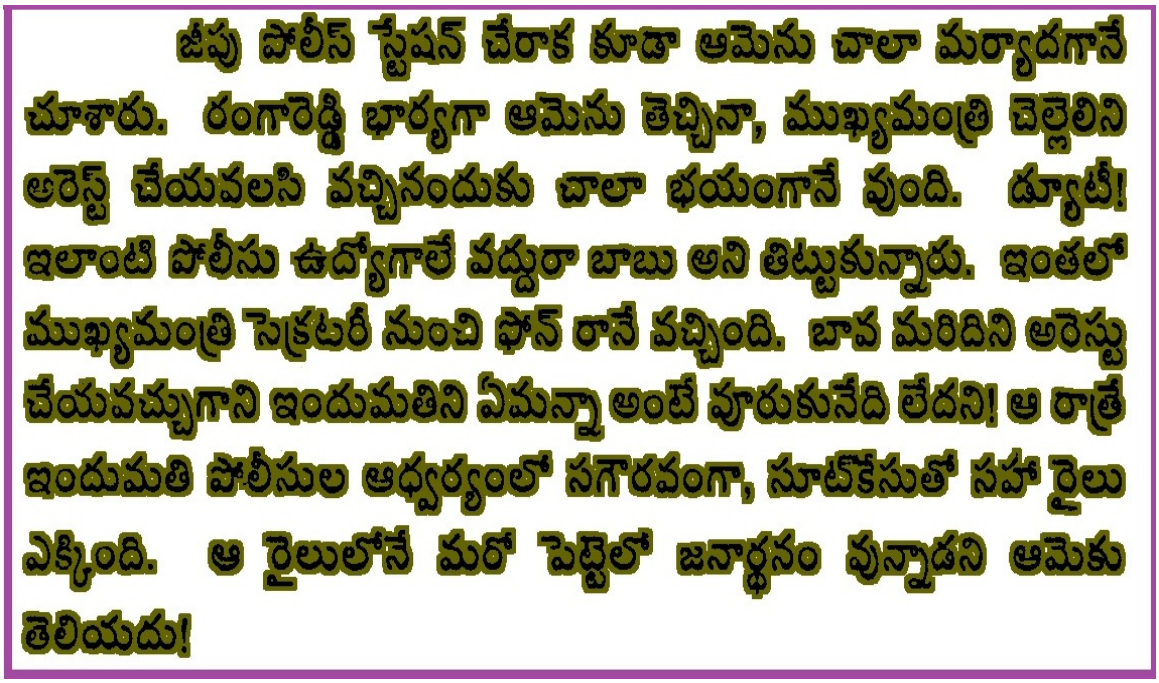


Figure 5.1: Word segmentation results of PFN based method on Telugu script.

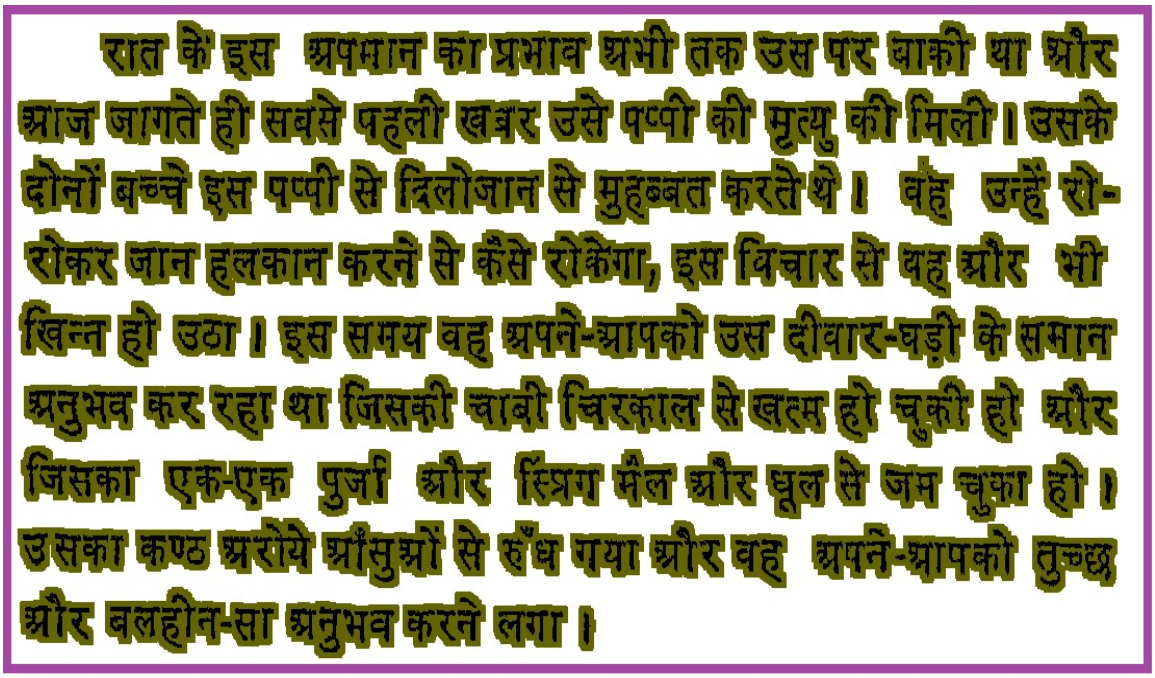


Figure 5.2: Word segmentation results of PFN based method on Devanagari script.

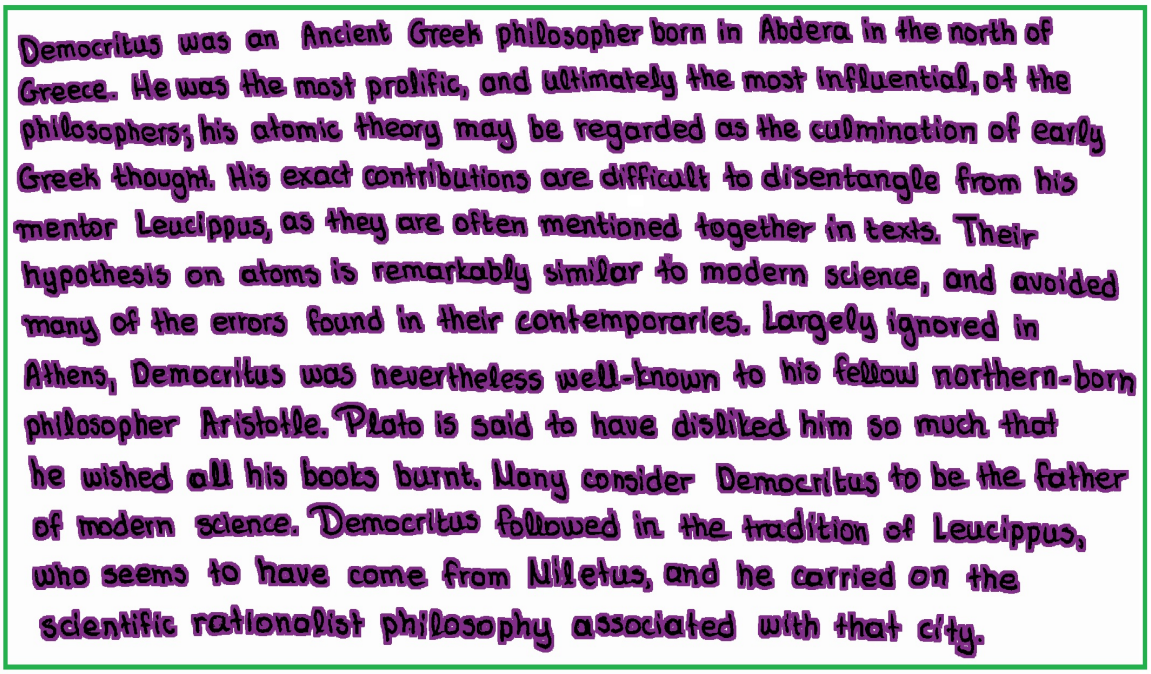


Figure 5.3: Word segmentation results of PFN based method on Roman script.

Figure 5.4: Word segmentation results of PFN based method on handwritten Bangla skewed document.

5.2 Baseline Detection using Fringe Maps

Detecting baseline is one of the useful process in OCR system stage. The baseline can be used in either skew estimation, or for segmenting the words or characters from text line. It can also be used to extract dependent features. Using baseline, the characters and shape are classified into three groups Ascenders, descenders and special marks called diacritics [53]. The baseline is the line upon which most letters "sit". Several baseline detection methods are proposed in the literature. The horizontal projection based approach counts the elements on horizontal line and assumes that maximum number of elements on horizontal line is the baseline. Although it is robust and very easy to implement but it needs long straight line of text. The histogram projection mostly failed in estimating the correct baseline for ligatures having greater number of ascender and descender. This method has the defect to be very sensitive to the skew [8]. The Hough transform method transposition in the polar coordinates space of the tracing points makes it possible to detect agglomerations of point images intersection defining the angle of writing lines skew [30]. In [12,13,17,22,23] researches detected baseline using horizontal projection based approach for zoning the Indic scripts and for separating base character (BC) and consonant modifiers (CM).
Here we propose a new method using PFNs for detecting baseline. In this method, we assumes the connection of bottom point of base characters gives baseline. The method is largely derived for Telugu text but may be applied to other scripts. This method contains the following steps:

1. Generate Fringe map for the input binary text line image.
2. Compute PFNs in horizontal direction.
3. Label the CCs and find the PFNs inside the CCs, Filter the remaining PFNs.
4. Find base character by computing Moving average line using PFNs Y position, this line pass through the base characters.
5. Compute Base line using the bottom bound of base characters MBR.

Here we generate fringe map for input text line image. Then the PFNs along the horizontal direction are computed. To separate the PFNs inside and between the CCs, first label the CCs, then the PFNs between the CCs (a PFN between the pixels of neighboring CCs) are filtered out. We may have more than one PFN in a column, in such case we take average Y position of PFNs in that column. Now the base characters are find by generating a line which pass through the base character. We take a window of size width $3 \times AH$ and compute moving average of PFNs Y position. This gives a line which pass through the base characters. The bottom bound of MBR of base characters is taken as the points for generating base line. As we computed moving average of PFNs, similarly we consider base character bottom points for generating baseline. Fig. 5.5 shows results of the method. Note that the baseline is following the undulations of the text despite uneven skew.

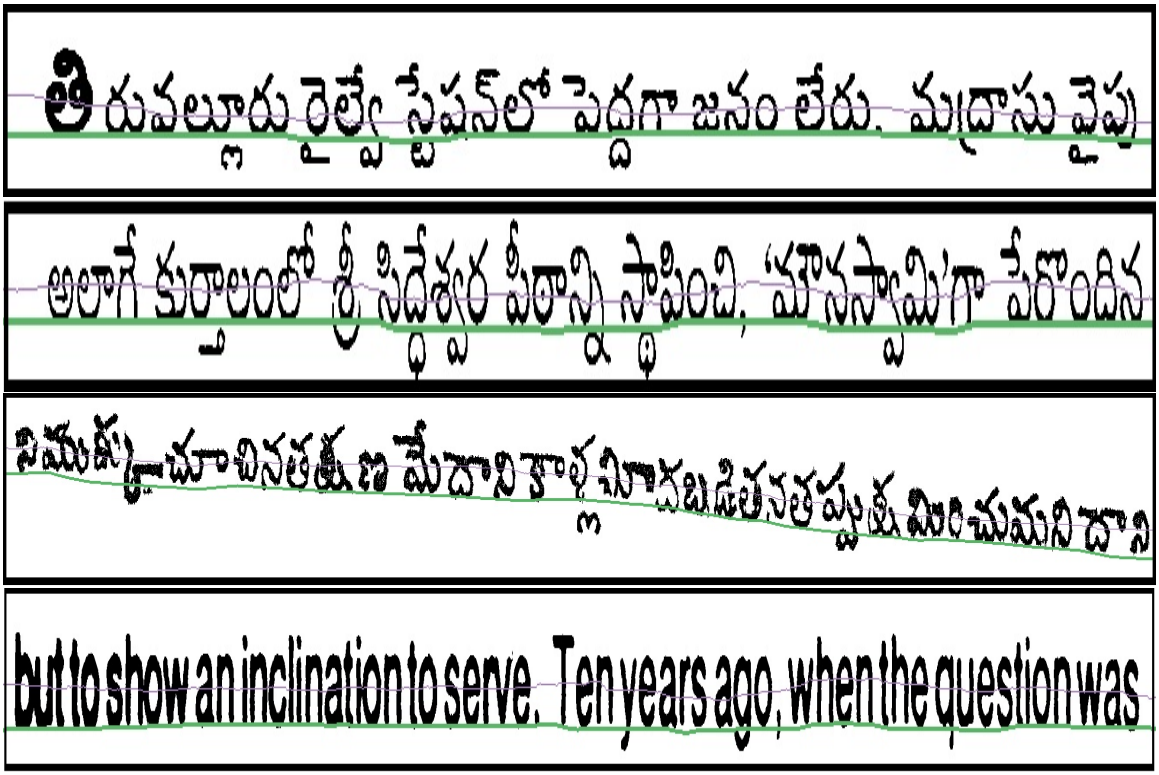


Figure 5.5: Baseline detection method results, Green color line is baseline and Violet color line is a line pass through the base characters.

5.3 Character Segmentation and Isolation of Region for Potential Recognition Units using Fringe Maps

Character segmentation is an operation that seeks to decompose an image of a sequence of characters into subimages of individual characters. In many machine print applications involving limited font sets each character occupies a block of fixed width. The pitch or number of characters per unit of horizontal distance, provides a basis for estimating segmentation points. The sequence of segmentation points obtained for a given line of print should be approximately equally spaced at the distance corresponding to the pitch. This provides a global basis for segmentation, since separation points are not independent. In Telugu documents, we find lot of space variation between characters due to consonant modifier and primitive typesetting as shown in Fig. 1.12.

In the literature [10], Projection profile based and Connected component based are used for character segmentation. The vertical projection of a text line consists of a simple running count of the black pixels in each column. It can serve for detection of white space between successive letters. This space is used for segmenting characters. A vertical projection is less satisfactory for the slanted characters commonly occurring in handwritten. We can't apply the projection profile methods on Indic scripts directly. We may not find linear space between characters as shown in Fig. 1.12. The Connected component based method is simple for segmenting the character of scripts like English language. But in Telugu and its oriented scripts character(Akshara) is a group of connected components. Grouping of connected components of a character is a problem. In [39] Negi et al first extracted connected components from the word image then after recognized each component and grouped as character.

Here we proposed a method for segmenting characters. In this method we segment character (Akashara) and label each component (glyph) of character. We extracted some characteristics from Telugu script.

1. Most of the base characters bottom bound lies on baseline.

2. Consonant modifiers have a MBR mid point that is near or below the baseline.
3. Consonant modifiers of a character are vertically staged with base character.

These characteristics are used to segregate and regroup the connected components of a character. The Character segmentation algorithm is given below.

1. Detect baseline for the input line or word image.
2. The *Bottom* bound of the CC is below the baseline are marked as consonant modifiers
3. Filter out the consonant modifiers from the image.
4. Extract base character and find character boundaries
5. Assign Consonant modifiers and vowel modifiers to the appropriate base character

We detect baseline for the input line or word image as we discussed in the section 5.2. Then the CCs whose *bottom* bound of bounding box is below the baseline are filtered and marked as consonant modifiers. The resultant image contains vowel modifiers and base characters. The base characters are identified as discussed in section 5.2 and remaining are vowel modifiers. The Character boundaries are computed by taking the base character *Left* bound as starting point and *Right* bound as ending point. Consonant modifiers and vowel modifiers are assigned to the base character with the following heuristics and label the CCs as vowel modifier if the *bottom* bound is above baseline, remaining are consonant modifiers.
Let CC_i be a connected component and its Left and Right bounds are CC_{iL} and CC_{iR}. Any pair (CC_i, CC_j), $CC_i \in BC$ and $CC_j \notin BC$

1. Overlap CC with one base character. If $CC_{iL} \leq CC_{jL} \leq CC_{jR} \leq CC_{iR}$ or $CC_{jL} \leq CC_{iL} \leq CC_{iR} \leq CC_{jR}$ or $CC_{jL} \leq CC_{iL} \leq CC_{jR}$ or $CC_{iL} \leq CC_{jL} \leq CC_{iR}$ then assign CC_j to the CC_i

2. Overlap CC with two base characters. In this case we assigned the CC to the higher overlapping base character. The overlapping with two base characters is check with $CC_{jL} \leq CC_{iR}$ and $CC_jR > CC_{(i+1)L}$.

Fig. 5.6 shows the result of akshara segmentation and components of askhara.

Figure 5.6: Telugu askshara segmentation and components of akshara.

5.4 Conclusion

In this chapter we extended the fringe map for segmenting word, characters and baseline detection. The proposed word segmentation is tested on Telugu and other scripts. This method is generic in nature because it script independent method. It is tested on handwritten skewed document, result shows good accuracy. We proposed baseline detection method, this is also tested on different scripts. The robustness of baseline detection method is detection baseline even the document is skewed. We proposed character segmentation method for Telugu scripts, this method segmenting characters and label glyphs of the character.

Chapter 6

Fringe Based Text Line Segmentation Method Applied to Indic Scripts and Handwritten Images

In this chapter we show another extension of this approach. Now show this same techniques are extended to the other categories of document images. That is printed documents Indic scripts and Handwritten documents of Latin, English and Bangla from ICDAR 2013 handwriting segmentation contest.

6.1 Introduction

A lot of research and development in OCR systems has led to the availability of a significant number of commercial OCRs in printed Roman and also in other Oriental scripts. However such OCR commercial products for printed Indic scripts is still a rarity. Efforts towards developing OCR systems are being made around the country and else where.

Research work at Institutes like ISI Kolkata published on Devnagari and Bangla, Tamil is published by IISC Bangalore, Telugu by University of Hyderabad, Hindi,

Marathi and Malayalam by CDAC, Gumukhi by Lehal et al and Oriya by Utkal University Bhubaneswar.

Pal and Chaudhuri [44] published a printed OCR system for Devanagari. Bansal [5] published two-pass mechanism to segment a uniform text zone into constituent text lines. In the first pass, lines are separated assuming that no line fusion or break exists. This segmentation is based on horizontal histograms of the document. Heights of all text lines obtained in first pass are divided into three bins. The number of text lines in each bin is counted. The bin which contains maximum text lines becomes the representative bin for properly segmented text lines. The average height of a text line in this bin is used as threshold height. A text line whose height is more than the threshold height is suspected to be the fusion of more than one text line. The suspected fused lines are further segmented in the second pass. Second pass makes an attempt to break a suspected fused line into constituent lines using threshold height. A vertical histogram of a text line is made and every gap of two or more pixels in the histogram is taken to be the word delimiter.

Chaudhuri and Pal [12] reported a complete printed Bangla OCR system. They segmented text blocks into lines by using the valleys of projection profiles. Words are segmented using vertical projection profiles. Pal and Datta [45] published a method using Piece-wise projection profiles for segmenting unconstrained handwritten text. Lehal and Singh presented an OCR system for printed Gurumukhi script [28] and Jindal et al [17] discussed segmentation problems in Gurumukhi script. Both used projection profile for segmenting lines and words. Aparna and Ramakrishnan [2] detailed a complete OCR system for Tamil script. Horizontal and vertical projection profiles are used for line and word segmentation. Kumar and Ramakrishnan [23] presented recognition system for printed Kannada text. In this, projection profiles are used for line segmentation. The character spacing in Kannada script is non-uniform due to the presence of consonant conjuncts. The spacing between the base characters in the middle zone becomes comparable to word spacing. Hence, morphological dilation is used to connect all the characters in a word, before performing word segmentation. Rahiman and Rajasree [51] published OCR for Malayalam script. They also projection profile based method for segmenting lines and words. Chaudhuri et

al [13] and Mohanty and Behera [35] used projection profiles for line and word segmentation in the OCR system for Oriya script. Tripathy and Pal [63] used piece-wise projection profile for segmenting Oriya script.
In most of the OCR System for Indic script used projection profile based methods for segmenting lines, words and characters. Here we apply our methods based on fringe maps for segmenting lines and words on various Indic scripts.

6.2 Text Line Segmentation Methods on Printed Documents of Indic Scripts

We present an experimental results of our method on different Indic script. The test data set was collected from the digital library of India. Test data set contains printed documents of Hindi, Bangla, Tamil, Kannada and Oriya scripts. Total 150 text document pages with single column are selected for experiments. A typical image size is approximately 3000×5000 pixels. Fig. 6.1, Fig. 6.2, Fig. 6.3 and Fig. 6.4 shows results of our line segmentation method on different Indic scripts. We quantitatively experimented different methods on Indic scripts. Performance of these are tabulate in Table 6.1.

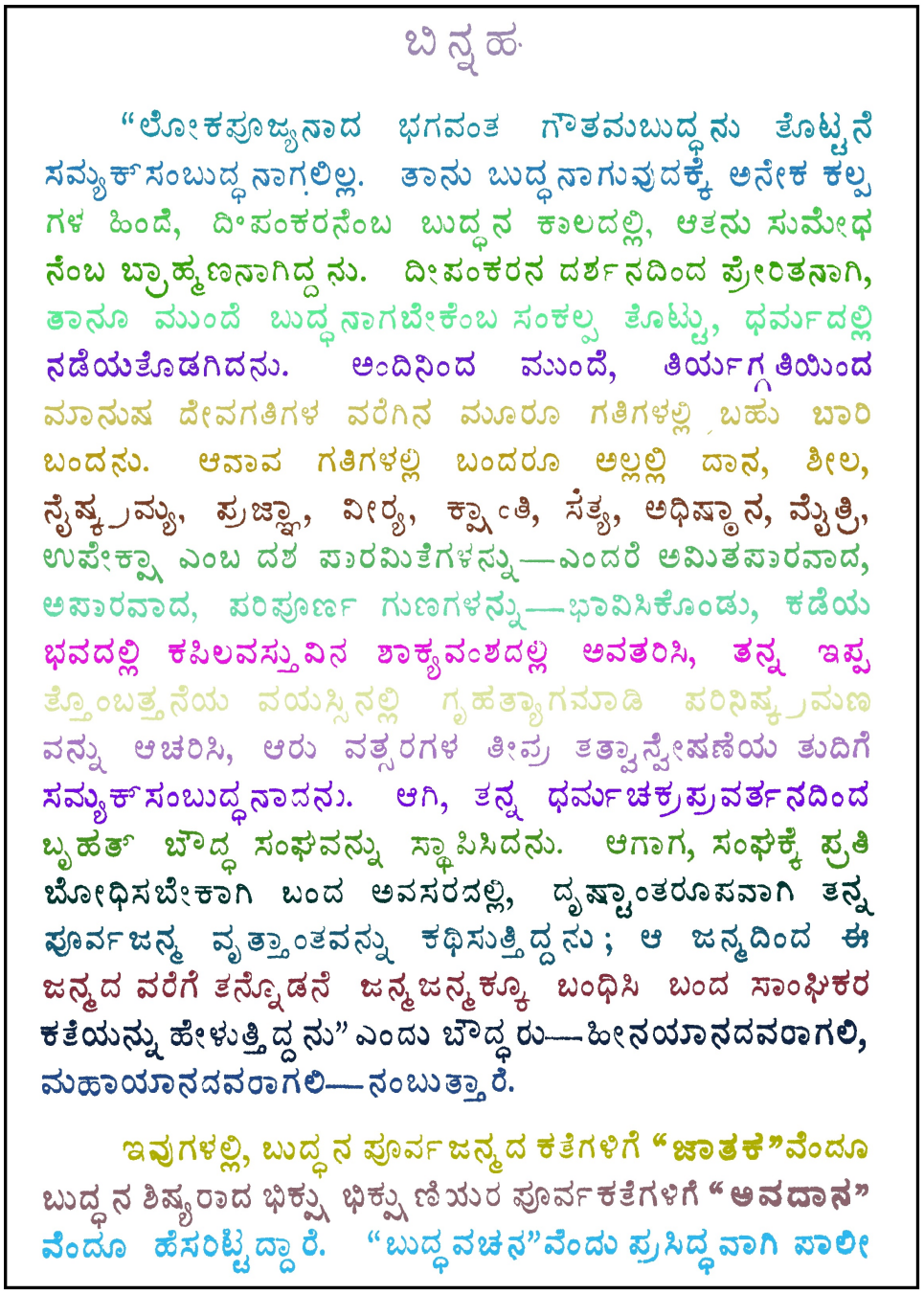
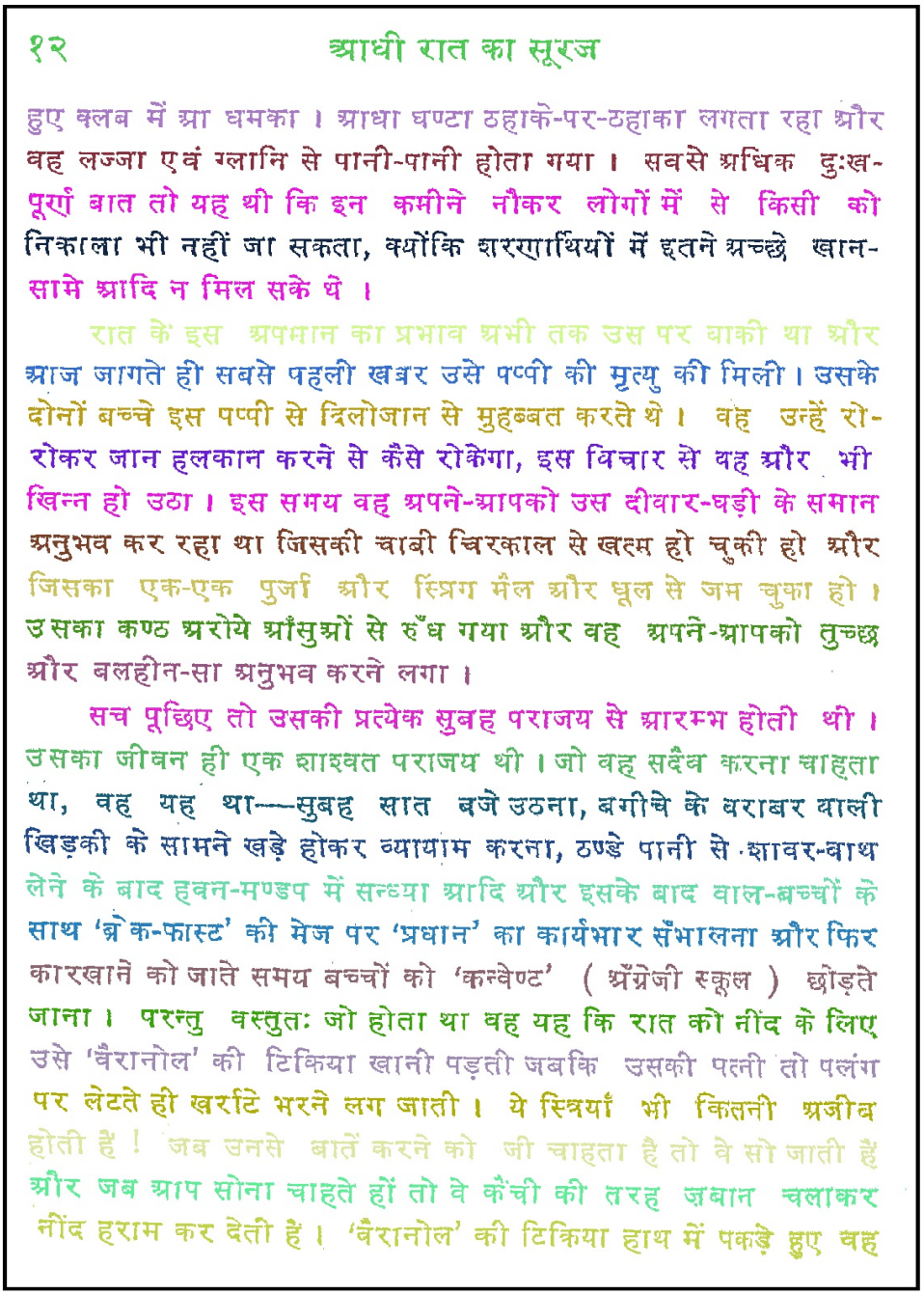


Figure 6.1: Line segmentation result of our method on Kannada script.



१२ आधी रात का सूरज

हुए क्लब में आ धमका। आधा घण्टा ठहाके-पर-ठहाका लगता रहा और वह लज्जा एवं ग्लानि से पानी-पानी होता गया। सबसे अधिक दुःख-पूर्ण बात तो यह थी कि इन कमीने नौकर लोगों में से किसी को निकाला भी नहीं जा सकता, क्योंकि शरणार्थियों में इतने अच्छे खान-सामे आदि न मिल सके थे।

रात के इस अपमान का प्रभाव अभी तक उस पर बाकी था और आज जागते ही सबसे पहली खबर उसे पप्पी की मृत्यु की मिली। उसके दोनों बच्चे इस पप्पी से दिलोजान से मुहब्बत करते थे। वह उन्हें रो-रोकर जान हलकान करने से कैसे रोकेगा, इस विचार से वह और भी खिन्न हो उठा। इस समय वह अपने-आपको उस दीवार-घड़ी के समान अनुभव कर रहा था जिसकी चाबी चिरकाल से खत्म हो चुकी हो और जिसका एक-एक पुर्जा और स्प्रिंग मैल और धूल से जम चुका हो। उसका कण्ठ अरोये आँसुओं से रुँध गया और वह अपने-आपको तुच्छ और बलहीन-सा अनुभव करने लगा।

सच पूछिए तो उसकी प्रत्येक सुबह पराजय से आरम्भ होती थी। उसका जीवन ही एक शाश्वत पराजय थी। जो वह सदैव करना चाहता था, वह यह था—सुबह सात बजे उठना, बगीचे के बराबर वाली खिड़की के सामने खड़े होकर व्यायाम करना, ठण्डे पानी से शावर-बाथ लेने के बाद हवन-मण्डप में सन्ध्या आदि और इसके बाद बाल-बच्चों के साथ 'ब्रेक-फास्ट' की मेज पर 'प्रधान' का कार्यभार सँभालना और फिर कारखाने को जाते समय बच्चों को 'कन्वेण्ट' (अँग्रेजी स्कूल) छोड़ते जाना। परन्तु वस्तुतः जो होता था वह यह कि रात को नींद के लिए उसे 'वैरानोल' की टिकिया खानी पड़ती जबकि उसकी पत्नी तो पलंग पर लेटते ही खर्राटे भरने लग जाती। ये स्त्रियाँ भी कितनी अजीब होती हैं! जब उनसे बातें करने को जी चाहता है तो वे सो जाती हैं और जब आप सोना चाहते हों तो वे कैंची की तरह ज़बान चलाकर नींद हराम कर देती हैं। 'वैरानोल' की टिकिया हाथ में पकड़े हुए वह

Figure 6.2: Line segmentation result of our method on Devanagari script.

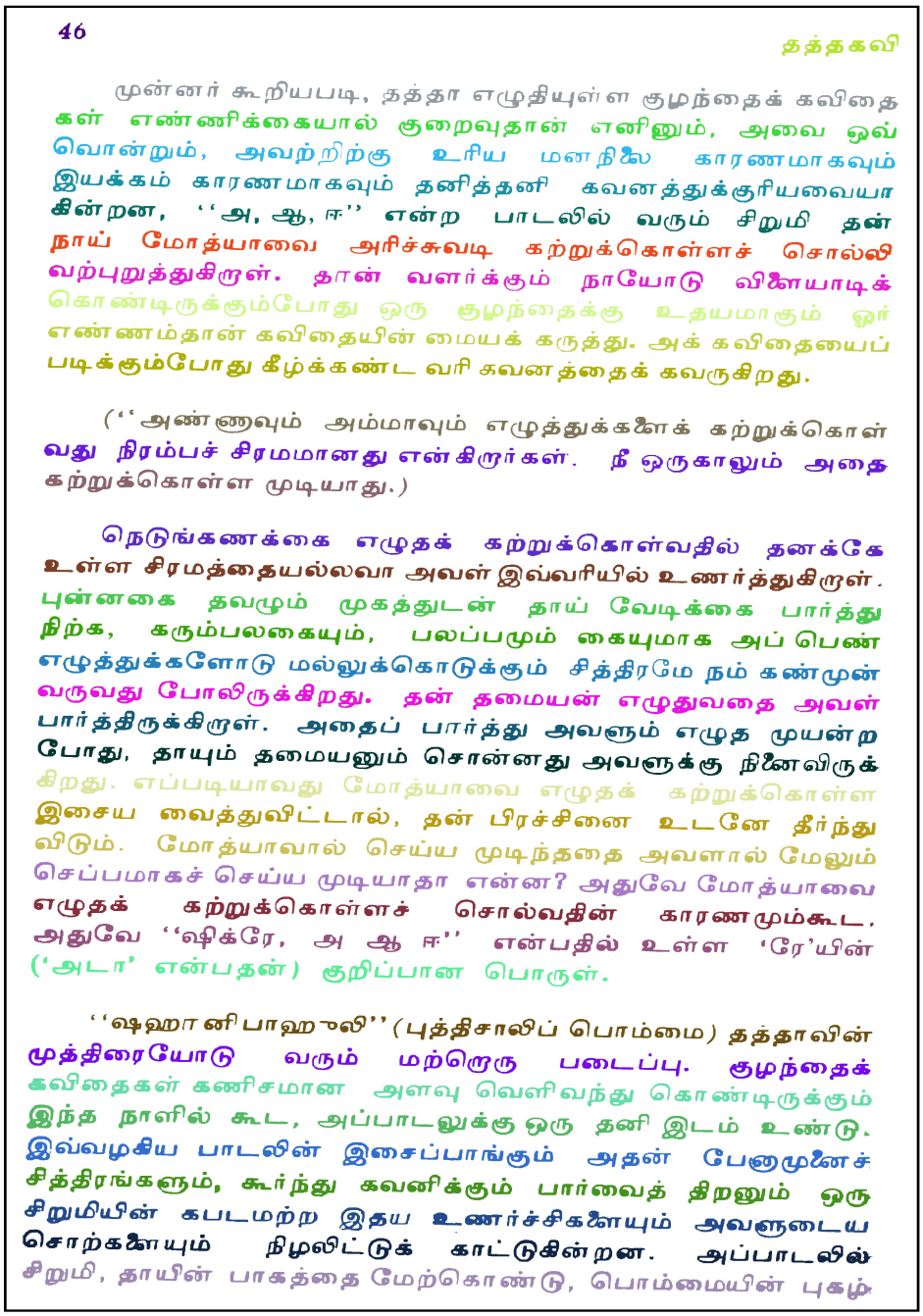


Figure 6.3: Line segmentation result of our method on Tamil script.

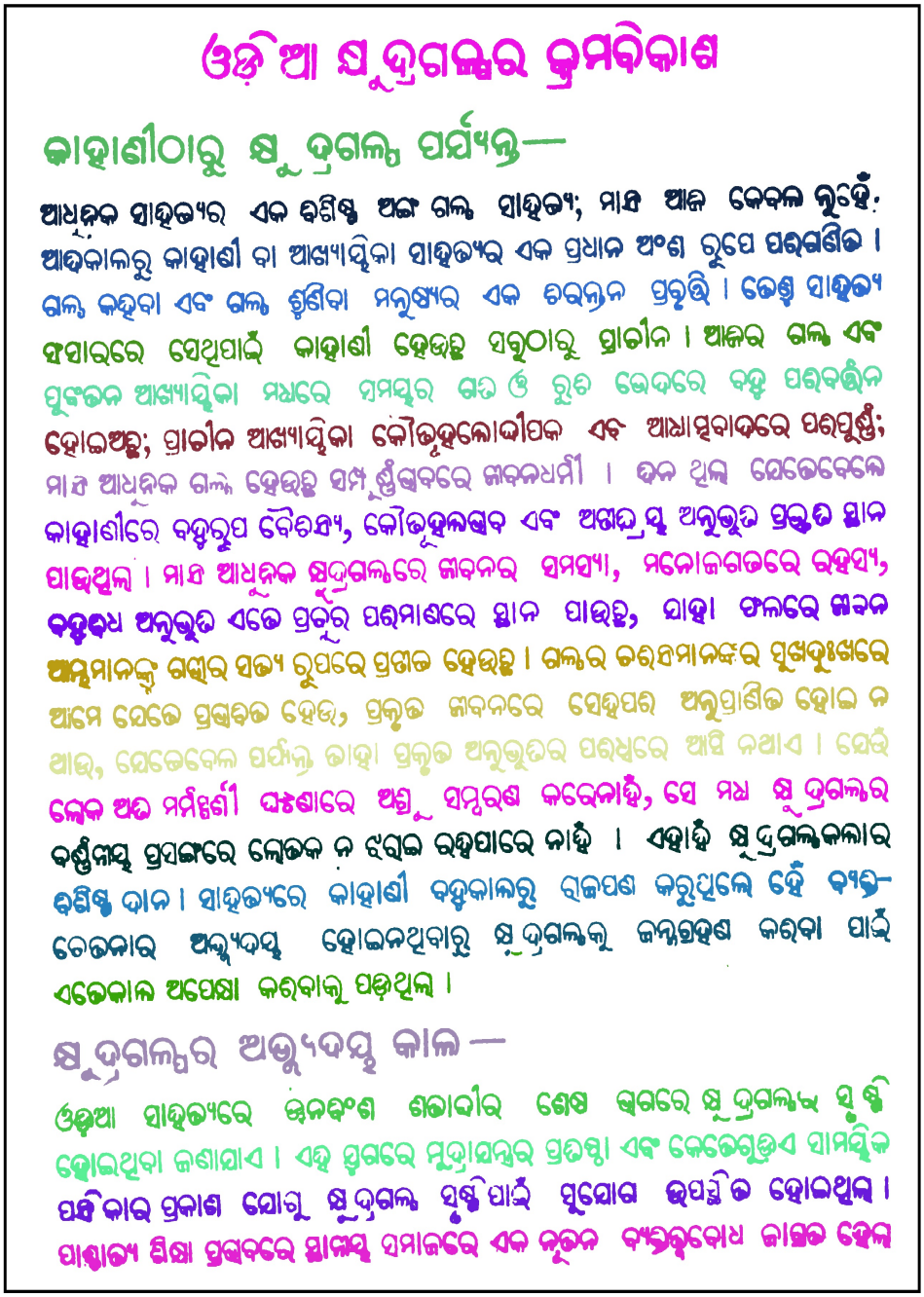


Figure 6.4: Line segmentation result of our method on Oriya script.

Table 6.1: Text Line Segmentation performance of different methods on Indic Scripts

$Ta = 95\%$								
Book Name	**Number of Pages**	**Actual Lines**	**Method**	**Detected Lines**	**one2one**	**Detected rate**	**Recognition accuracy**	**Pixel level hit ratio (FM)**
28 Kate 14 Chitra (Kannada)	30	739	ARLSA	856	568	0.769	0.664	0.712
			PP	457	298	0.403	0.652	0.498
			PAL	832	634	0.858	0.762	0.807
			PFN	**731**	**731**	**0.989**	**1**	**0.995**
Aadhi Raat Suraj (Hindi)	30	872	ARLSA	914	798	0.915	0.873	0.894
			PP	659	578	0.663	0.877	0.755
			PAL	848	832	0.954	0.981	0.967
			PFN	**866**	**864**	**0.991**	**0.998**	**0.994**
Alor Swakhar (Bengali)	30	713	ARLSA	789	613	0.856	0.776	0.816
			PP	489	336	0.471	0.687	0.559
			PAL	725	699	0.980	0.964	0.972
			PFN	**704**	**704**	**0.987**	**1**	**0.993**
Dattaakavi (Tamil)	30	1146	ARLSA	1306	893	0.779	0.684	0.728
			PP	874	658	0.574	0.753	0.651
			PAL	1024	792	0.691	0.773	0.73
			PFN	**1142**	**1132**	**0.987**	**0.991**	**0.989**
Oriya Galpamala (Oriya)	30	764	ARLSA	859	608	0.796	0.708	0.749
			PP	488	294	0.385	0.602	0.47
			PAL	827	612	0.801	0.74	0.769
			PFN	**752**	**748**	**0.979**	**0.995**	**0.987**

6.3 Text Line Segmentation Methods on Handwritten Documents

Segmentation of handwritten text into lines is a complex and challenging task due to variety of different situations. In particular difficulty arises with the situations:

1. Large variation of spaces between text lines,
2. Components of adjacent text lines touching each other
3. Variation of skew angle between text lines

4. Text with curvilinear lines

5. Characters with different sizes and variable intra-word gaps, etc.

All these problems seriously affect the segmentation and, consequently, the recognition accuracy. We collected benchmarking data set from the ICDAR 2013 Handwriting Segmentation contest. In the previous contest the data set are from English and Greek but in the contest ICDAR 2013 they included the Indian script Bangla. We experimented 42 Roman documents and 26 Bengali documents with a total of 1251 text lines. Fig. 6.5 and Fig. 6.6 shows the result of our method on sample images of ICDAR 2013 Handwriting segmentation contest. Table 6.2 shows the performance of the methods. Our method shows better performance than other methods. In ICDAR 2013 Contest the acceptance threshold used was Ta=95% for text line segmentation. INMC method shows 98.66% performance for text line segmentation. Our method resulted 91.1% for Ta=95%.

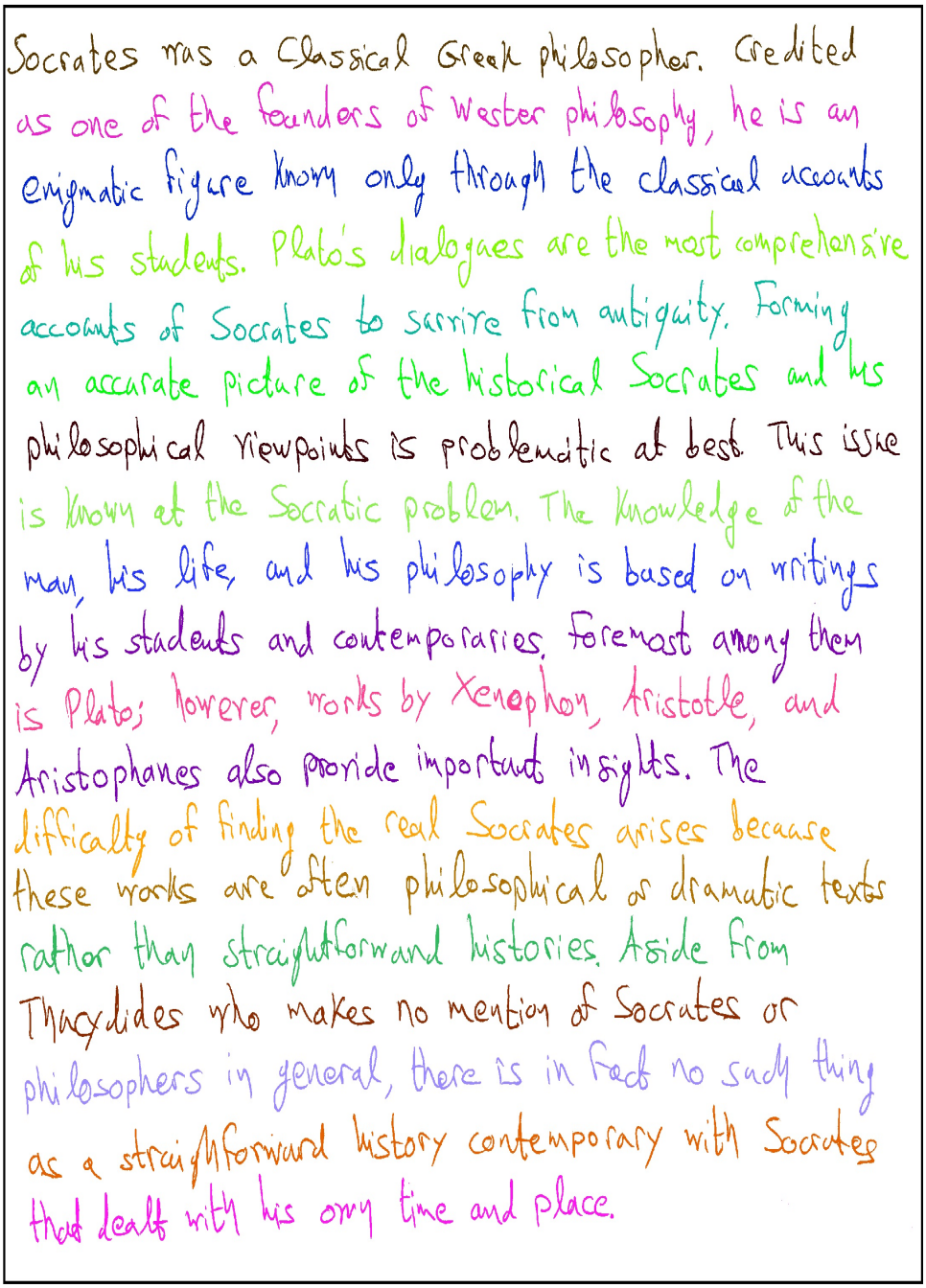


Figure 6.5: Line segmentation result of our method on Roman Handwritten documents.

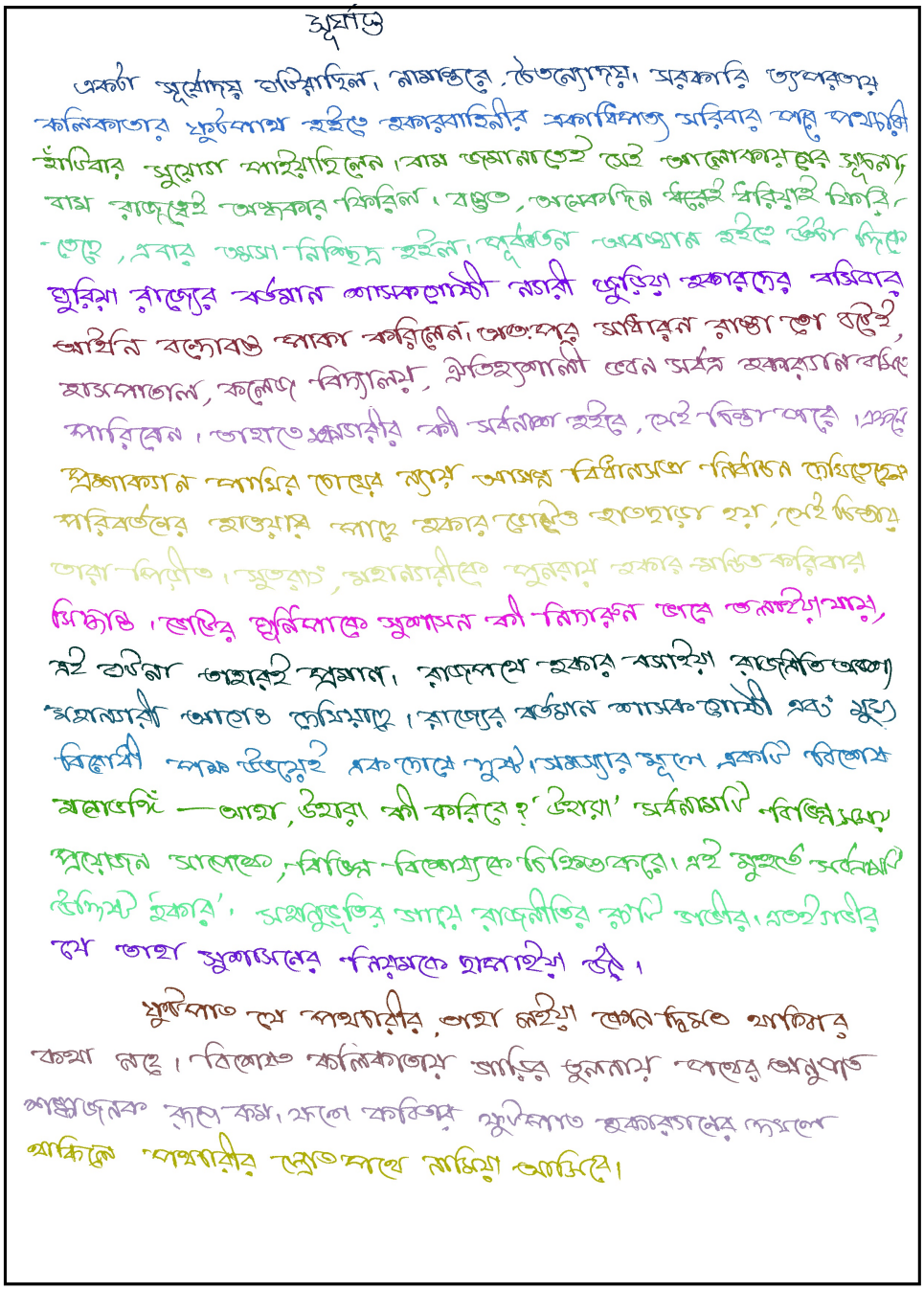


Figure 6.6: Line segmentation result of our method on Bengali Handwritten documents.

Table 6.2: Text Line Segmentation performance of different methods on Handwritten documents (ICDAR 2013 DATA SET).

Name of script	**Number of Pages**	**Actual Lines**	**Method**	**Detected Lines**	**one2one**	**Detected rate**	**Recognition accuracy**	**Pixel level hit ratio (FM)**
$Ta = 100\%$								
BENGALI	26	445	ARLSA	536	91	0.204	0.169	0.185
			PP	138	74	0.166	0.536	0.254
			PAL	332	98	0.220	0.229	0.252
			PFN	**496**	**167**	**0.375**	**0.336**	**0.354**
ROMAN	42	806	ARLSA	827	173	0.215	0.209	0.212
			PP	311	119	0.147	0.382	0.213
			PAL	773	183	0.227	0.237	0.232
			PFN	**825**	**368**	**0.456**	**0.446**	**0.451**
$Ta = 95\%$								
BENGALI	26	445	ARLSA	536	232	0.521	0.433	0.473
			PP	138	79	0.177	0.572	0.271
			PAL	332	225	0.506	0.678	0.579
			PFN	**496**	**369**	**0.829**	**0.743**	**0.784**
ROMAN	42	806	ARLSA	827	479	0.594	0.579	0.567
			PP	311	172	0.213	0.553	0.308
			PAL	773	468	0.581	0.605	0.593
			PFN	**825**	**743**	**0.921**	**0.9**	**0.911**
$Ta = 90\%$								
BENGALI	26	445	ARLSA	536	255	0.573	0.476	0.520
			PP	138	80	0.179	0.579	0.274
			PAL	332	267	0.6	0.804	0.687
			PFN	**496**	**411**	**0.923**	**0.828**	**0.873**
ROMAN	42	806	ARLSA	827	497	0.616	0.601	0.608
			PP	311	172	0.213	0.553	0.308
			PAL	773	499	0.619	0.646	0.632
			PFN	**825**	**775**	**0.961**	**0.939**	**0.950**
$Ta = 80\%$								
BENGALI	26	445	ARLSA	536	268	0.602	0.5	0.546
			PP	138	80	0.179	0.579	0.274
			PAL	332	284	0.638	0.855	0.731
			PFN	**496**	**434**	**0.975**	**0.875**	**0.922**
ROMAN	42	806	ARLSA	827	509	0.631	0.615	0.623
			PP	311	172	0.213	0.553	0.307
			PAL	773	521	0.646	0.674	0.66
			PFN	**825**	**794**	**0.985**	**0.962**	**0.973**

6.4 Conclusion

In this chapter we tested the performance of text line segmentation method (proposed in chapter 4, Text and Space affinity using fringe map) on different Indic scripts and Handwritten documents. Indic scripts (Hindi, Bengali, Oriya, Tamil and Kannada) each of 50 printed document pages are collected from digital library of India. The test performance our method shows 99.4% on Hindi, 99.3% on Bengali, 98.7% on Oriya, 98.9% on Tamil and 99.5% on Kannada. This method is tested on data set of ICDAR 2013 Handwritten Segmentation Contest. The performance of the method shows 91.1% on Roman and 78.4% on Bengali scripts for Threshold Ta of 95% (95% pixel level accuracy against the ground truth). In the ICDAR 2013 contest INMC method shows 98.66% performance. Our proposed method for Telugu script text line segmentation is tested without any design changes. By modifying the design of overlapping components assignment and touching components cutting can improve the performance of the method.

Chapter 7

Concluding Remarks

In this work a comprehensive study on segmentation of Telugu text is presented. Previous attempts to segment Telugu text used classical methods like projection profiles, RLSA and CC based methods. However these methods are unable to produce a sufficiently high accuracy of segmentation. We extracted the characteristics of text segmentation approaches in the literature and derived the concepts of Text affinity, Space affinity and their inter-play with other text attributes for designing accurate segmentation methods. Text affinity is the term we use to categorize methods that attempt to group together related text into words and lines. Space affinity segregate text based on background information. We design here a high accuracy method of segmentation using both text and space affinity. We presented a fringe map based approach to bring out a holistic balance between the Text and Space affinity. Fringe maps were used in OCR previously only for character recognition. We proposed novel operations on fringe maps which can extend their use for text line segmentation. The proposed most important operation on fringe maps that was proposed here is the PFN approach to analyse the fringe values. PFN formalism was crucial to the development of the proposed text segmentation methods. We proposed text line segmentation method for Telugu text using fringe maps. It is tested upon the corpus of 1025 printed Telugu document images collected from Robust OCR Project, University of Hyderabad and Digital Library of India. Our method shows 99.82% performance, which is superior than other methods. This is the most comprehen-

sive study on line segmentation of Telugu text. Word and character segmentation and baseline detection methods for Telugu text are also proposed. It was shown in experimentation that text line, word and baseline detection methods are generic and script independent. We tested these methods on different scripts and document images with handwritting. Complex cases such as touching, overlapping and skew are also successfully resolved by proposed approach. We tested proposed text line segmentation method on 68 page images of ICDAR 2013 Handwritten Segmentation Contest data set. Our method shows performance of 91.1% on Roman scripts and 78.4% on Bangala for 0.95 pixel level hit ratio. The performance of the text line segmentation algorithm here is reduced due to hard cutting of touching components and misclassification of over height components. The creation of a pixel level ground truth data set for the purpose of evaluation of the text line segmentation approaches is one of the contributions of this work. In our future work we shall do more extensive tests on Telugu and also various Indic scripts of printed and handwritten documents. Machine learning approach may be attempted to make it more robust.

Bibliography

[1] A Antonacopoulos and RT Ritchings. Flexible page segmentation using the background. In *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*, volume 2, pages 339–344. IEEE, 1994.

[2] KG Aparna and AG Ramakrishnan. A complete Tamil optical character recognition system. In *Document Analysis Systems V*, pages 53–57. Springer, 2002.

[3] Henry S Baird. Global-to-Local layout analysis. *IAPR Workshop on Synatactic and Structural Pattern Recognition*, pages 136–147, Sep. 1988.

[4] Henry S Baird. Background structure in document images. *International Journal of Pattern Recognition and Artificial Intelligence*, 8(05):1013–1030, 1994.

[5] Veena Bansal and RMK Sinha. Integrating knowledge sources in Devanagari text recognition system. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 30(4):500–505, 2000.

[6] Chakravarthy Bhagvati, Tanuku Ravi, S. Mahesh Kumar, and Atul Negi. On developing high accuracy OCR systems for Telugu and other Indian scripts. In *Language Engineering Conference*, volume 13-15, pages 18–23, Hyderabad, India, December 2002. IEEE Computer Society.

[7] Gunilla Borgefors. Distance transformations in digital images. *Comput. Vision Graph. Image Process.*, 34(3):344–371, June 1986.

[8] Houcine Boubaker, Monji Kherallah, and Adel M Alimi. New algorithm of straight or curved baseline detection for short Arabic handwritten writing. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 778–782. IEEE, 2009.

[9] R.L. Brown. The fringe distance measure: an easily calculated image distance measure with recognition results comparable to Gaussian blurring. *Systems, Man and Cybernetics, IEEE Transactions on*, 24(1):111–115, Jan. 1994.

[10] Richard G Casey and Eric Lecolinet. A survey of methods and strategies in character segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(7):690–706, 1996.

[11] R Cattoni, T Coianiz, S Messelodi, and CM Modena. Geometric layout analysis techniques for document image understanding: a review. *ITC-irst Technical Report*, 9703(09), 1998.

[12] BB Chaudhuri and U Pal. A complete printed Bangla OCR system. *Pattern recognition*, 31(5):531–549, 1998.

[13] BB Chaudhuri, U Pal, and Mandar Mitra. Automatic recognition of printed Oriya script. *Sadhana*, 27(1):23–34, 2002.

[14] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Distance transforms of sampled functions. *Theory of Computing*, 8(1):415–428, 2012.

[15] Jaekyu Ha, Robert M Haralick, and Ihsin T Phillips. Document page decomposition by the bounding-box project. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 2, pages 1119–1122. IEEE, 1995.

[16] CV Jawahar, MNSSK Pavan Kumar, and SS Ravi Kiran. A bilingual OCR for Hindi-Telugu documents and its applications. In *null*, page 408. IEEE, 2003.

[17] M. K. Jindal, G. S. Lehal, and R. K. Sharma. Segmentation Problems and Solutions in Printed Degraded Gurmukhi Script. *Signal Processing*, 2006.

[18] Rangachar Kasturi, Lawrence OGorman, and Venu Govindaraju. Document image analysis: A primer. *Sadhana*, 27(1):3–22, 2002.

[19] K Kise, J Sugiyama, N Babaguchi, and Y Tezuka. Layout model based analysis of document structure. *Trans. IEICE*, 72:1029–1039, 1989.

[20] Koichi Kise, Akinori Sato, and Motoi Iwata. Segmentation of page images using the area Voronoi diagram. *Computer Vision and Image Understanding*, 70(3):370–382, 1998.

[21] Koichi Kise, O Yanagida, and Shinobu Takamatsu. Page segmentation based on thinning of background. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 3, pages 788–792. IEEE, 1996.

[22] Vijaya Kumar Koppula, Atul Negi, and Utpal Garain. Robust text line, word and character extraction from Telugu document image. In *ICETET*, pages 269–272, 2009.

[23] B Vijay Kumar and AG Ramakrishnan. Machine recognition of printed Kannada text. In *Document Analysis Systems V*, pages 37–48. Springer, 2002.

[24] KS Sesh Kumar, Anoop M Namboodiri, and CV Jawahar. Learning segmentation of documents with complex scripts. In *Computer Vision, Graphics and Image Processing*, pages 749–760. Springer, 2006.

[25] P.P. Kumar, C. Bhagvati, A Negi, A Agarwal, and B. L. Deekshatulu. Towards improving the accuracy of Telugu OCR systems. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 910–914, Sept. 2011.

[26] Vasantha Lakshmi and C. Patvardhan. An optical character recognition system for printed Telugu text. *Pattern Anal. Appl.*, 7(2):190–204, July 2004.

[27] G. S. Lehal and Chandan Singh. A complete OCR system for Gurmukhi script. In *Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR In-*

ternational Workshops SSPR 2002 and SPR 2002, Windsor, Ontario, Canada, August 6-9, 2002, Proceedings, pages 358–367, 2002.

[28] GS Lehal and Chandan Singh. A Gurmukhi script recognition system. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, pages 557–560. IEEE, 2000.

[29] Yi Li, Yefeng Zheng, David Doermann, Stefan Jaeger, and Yi Li. Script-Independent text line segmentation in freestyle handwritten documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1313–1329, 2008.

[30] Laurence Likforman-Sulem, Anahid Hanimyan, and Claudie Faure. A Hough based algorithm for extracting text lines in handwritten documents. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 2, pages 774–777. IEEE, 1995.

[31] Laurence Likforman-Sulem, Abderrazak Zahour, and Bruno Taconet. Text line segmentation of historical documents: A survey. *IJDAR*, 9(2-4):123–138, 2007.

[32] G. Louloudis, B. Gatos, and C. Halatsis. Text line detection in unconstrained handwritten documents using a Block-Based Hough Transform approach. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02*, ICDAR '07, pages 599–603, Washington, DC, USA, 2007. IEEE Computer Society.

[33] R. Manmatha and Jamie L. Rothfeder. A scale space approach for automatically segmenting words from historical handwritten documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1212–1225, 2005.

[34] U. V. Marti and H. Bunke. Text line segmentation and word recognition in a system for general writer independent handwriting recognition. In *In Sixth International Conference on Document Analysis and Recognition*, pages 159–163. IEEE Computer Society, 2001.

[35] Sanghamitra Mohanty and Hemanta Kumar Behera. A complete OCR development system for Oriya script. *Proceedings of SIMPLE*, 4, 2004.

[36] George Nagy. Twenty years of document image analysis in PAMI. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1):38–62, Jan. 2000.

[37] George Nagy, Sharad C Seth, and Spotswood D Stoddard. Document analysis with an expert system. In *Pattern recognition in practice II*, pages 149–155. New York: Elsevier Science, 1986.

[38] George Nagy, Sharad C. Seth, and Mahesh Viswanathan. A prototype document image analysis system for technical journals. *IEEE Computer*, 25(7):10–22, 1992.

[39] Atul Negi, Chakravarthy Bhagvati, and B. Krishna. An OCR System for Telugu. In *6th International Conference on Document Analysis and Recognition (ICDAR 2001)*, pages 1110–1114, Seattle, WA, USA, September 2001. IEEE Computer Society.

[40] Nikos Nikolaou, Michael Makridis, Basilis Gatos, Nikolaos Stamatopoulos, and Nikos Papamarkos. Segmentation of historical machine-printed documents using Adaptive Run Length Smoothing and skeleton segmentation paths. *Image Vision Comput.*, 28(4):590–604, April 2010.

[41] Nicolas Normand and Christian Viard-Gaudin. A background based adaptive page segmentation algorithm. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 138–141. IEEE, 1995.

[42] L. O'Gorman. The document spectrum for page layout analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(11):1162–1173, November 1993.

[43] Lawrence O'Gorman and Rangachar Kasturi. Document image analysis. *IEEE*, page 125, 1995.

[44] U Pal and BB Chaudhuri. Printed Devnagari script OCR system. *VIVEK-BOMBAY-*, 10:12–24, 1997.

[45] U. Pal and Sagarika Datta. Segmentation of Bangla unconstrained handwritten text. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*, ICDAR '03, pages 1128–1132, Washington, DC, USA, 2003. IEEE Computer Society.

[46] Vassilis Papavassiliou, Themos Stafylakis, Vassilis Katsouros, and George Carayannis. Handwritten document image segmentation into text lines and words. *Pattern Recognition*, 43(1):369–377, 2010.

[47] Theo Pavlidis and Jiangyin Zhou. Page segmentation by white streams. In *Proc. 1st Int. Conf. Document Analysis and Recognition (ICDAR), Int. Assoc. Pattern Recognition*, pages 945–953, 1991.

[48] Ihsin T Phillips and Atul K Chhabra. Empirical performance evaluation of graphics recognition systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(9):849–870, 1999.

[49] Nallapareddy Priyanka, Srikanta Pal, and Ranju Mandal. Line and word segmentation approach for printed documents. *IJCA Special Issue on Recent Trends in Image Processing and Pattern Recognition-RTIPPR*, pages 30–36, 2010.

[50] Arun K Pujari, C Dhanunjaya Naidu, M Sreenivasa Rao, and BC Jinaga. An intelligent character recognizer for Telugu scripts using multiresolution analysis and associative memory. *Image and Vision Computing*, 22(14):1221–1227, 2004.

[51] M Abdul Rahiman and MS Rajasree. OCR for Malayalam script using neural networks. In *Ultra Modern Telecommunications & Workshops, 2009. ICUMT'09. International Conference on*, pages 1–6. IEEE, 2009.

[52] Zaidi Razak, Khansa Zulkiflee, Mohd Yamani Idna Idris, Emran Mohd Tamil, Mohd Noorzaily Mohamed Noor, Rosli Salleh, Mohd Yaakob, Zulkifli Mohd Yusof, and Mashkuri Yaacob. Off-line handwriting text line segmentation: A review. *International journal of computer science and network security*, 8(7):12–20, 2008.

[53] Muhammad Imran Razzak, Muhammad Sher, and SA Hussain. Locally baseline detection for online Arabic script based languages character recognition. *International Journal of the Physical Sciences*, 5(7):955–959, 2010.

[54] A. Rosenfeld and J.L. Pfaltz. Distance functions on digital pictures. *Pattern Recognition*, 1(1):33 – 61, 1968.

[55] Azriel Rosenfeld and John L. Pfaltz. Sequential operations in digital picture processing. *J. ACM*, 13(4):471–494, Oct. 1966.

[56] Mehmet Sezgin and Blent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *J. Electronic Imaging*, 13(1):146–168, 2004.

[57] Faisal Shafait, Daniel Keysers, and Thomas M. Breuel. Performance comparison of six algorithms for page segmentation. In *7th IAPR Workshop on Document Analysis Systems*, pages 368–379. Springer, 2006.

[58] A Lawrence Spitz. Recognition processing for multilingual documents. In *Procee dings of the 1990 Intl Conf. on Electro nic Publish ing, Docu ment Manip ulation and Typograph y, Gaithe rsburg, Maryland*, pages 193–2, 1990.

[59] S. N. Srihari and V. GovindarajuKumar. Analysis of textual images using the Hough Transform. In *Machine Vision and Applications*, pages 141–153, 1989.

[60] Sargur N Srihari. Document image understanding. In *Proceedings of 1986 ACM Fall joint computer conference*, pages 87–96. IEEE Computer Society Press, 1986.

[61] B Anuradha Srinivas, Arun Agarwal, and C Raghavendra Rao. An overview of OCR research in Indian scripts. *IJCSES*, 2(2):141–153, 2008.

[62] Nikolaos Stamatopoulos, Basilis Gatos, Georgios Louloudis, Umapada Pal, and Alireza Alaei. ICDAR 2013 handwriting segmentation contest. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 1402–1406. IEEE, 2013.

[63] N. Tripathy and U. Pal. Handwriting segmentation of unconstrained Oriya text. In *Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition*, IWFHR '04, pages 306–311, Washington, DC, USA, 2004. IEEE Computer Society.

[64] Kwan Y. Wong, Richard G. Casey, and Friedrich M. Wahl. Document analysis system. *IBM journal of research and development*, 26(6):647–656, 1982.