An Efficient Algorithm to Find the Height of a Text Line and Overcome Overlapped and Broken Line Problem during Segmentation

Sanjibani Sudha Pattanayak[1], Sateesh Kumar Pradhan[2]
Department of Computer Science and Application
Utkal University, Bhubaneswar, India

Ramesh Chandra Mallik[3]
P.G. Department of Odia Language and Literature,
Utkal University, Bhubaneswar, India

***Abstract*—Line segmentation is a critical phase of the Optical Character Recognition (OCR) which separates the individual lines from the image documents. The accuracy rate of the OCR tool is directly proportional to the line segmentation accuracy followed by the word/character segmentation. In this context, an algorithm, named height_based_segmentation is proposed for the text line segmentation of printed Odia documents. The proposed algorithm finds the average height of a text line and it helps to minimize the overlapped text line cases. The algorithm also includes post-processing steps to combine the modifier zone with the base zone. The performance of the algorithm is evaluated through the ground truth and also by comparing it with the existing segmentation approaches.**

***Keywords*—Document image analysis; line segmentation; word segmentation; database creation; printed Odia document**

I. Introduction

The Segmentation phase is one of the important phases of the character recognition process which separates the individual lines or words or characters from the image documents. There are several challenges associated with this segmentation process which are intended to discuss taking into consideration of the printed Odia documents.

The Projection profile-based method is a traditional method of segmentation. It has been noticed that a few problems may appear if the projection-profile-bases method will be adopted for segmentation. Therefore, the issues and challenges involved in the process of segmentation are discussed.

Odia language has a rich set of symbols. Approximately, 300 (three hundred) symbols comprising of the consonants, vowels, conjuncts, modifiers, digits, special symbols, etc. The modifiers for a symbol can be placed in different positions; top, bottom, left or right. Odia symbols do not maintain any headlines like other north Indian scripts (Devanagari, Bangla...). Below shown is an example of an Odia line (Fig. 1). The present example shows the different zones, such as base zone, upper modifier zone, and lower modifier zone.

It is not necessary that every text line would contain a lower or upper modifier zone. It may contain only base zone, base zone with any one of the upper/lower modifier zone or base zone with both the modifier zones.

A. Challenges Faced During Text Line Segmentation of Printed Odia Document

During the segmentation, the contents of two text lines might be overlapped or touched in different places making the segmentation process challenging. This is shown here in Fig. 2.

Furthermore, there are cases where during segmentation, the upper and lower modifier zone get separated from the base zone resulting in split or broken lines (Fig. 3).

The skewness of the image document cannot be avoided as manual interpretation for scanning is involved (shown in Fig. 4). There is a problem associated with document base-line detection. Without correcting the skewness, the segmentation becomes difficult.

A document may contain fonts of different sizes e.g. large font for heading or large initial letter (Fig. 5). This is another challenge in front of the text line segmentation.

There might be more than one column in the same document which makes the segmentation process projecting more challenges (Fig. 6).

In the proposed algorithm resolves issues like overlapped text lines, splitted lines and documents with different font sizes. It has been evaluated on different types of documents like good quality, low text density, degraded images, etc.

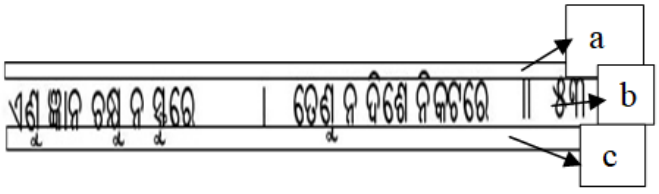

Fig. 1. Different Zones of an Odia Text Line (a) Upper Modifier Zone (b) Base Zone (c) Lower Modifier Zone.

Fig. 2. Example of Overlapped Lines.

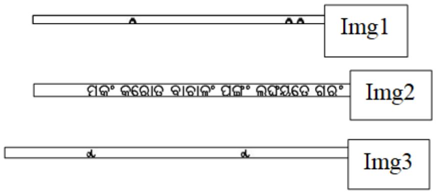

Fig. 3. Demonstration of Text Line Broken into 3 different Images; (Img1:) upper Modifier Zone, (Img2:) base Zone, (Img3:) Lower Modifier Zone of a Single Line.

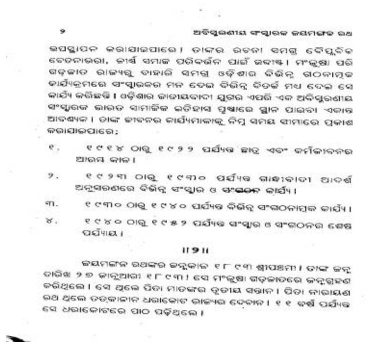

Fig. 4. Example of a Skewed Document.

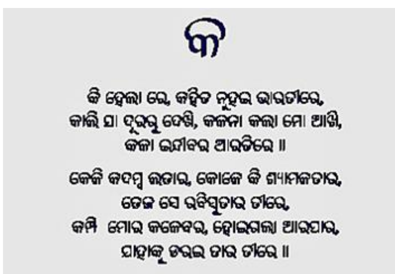

Fig. 5. Document with Large Font for Heading.

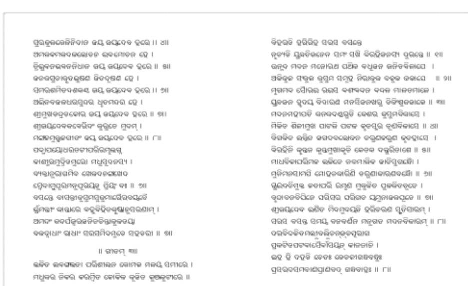

Fig. 6. Document having more than One Columns.

II. Review of Literature

The segmentation algorithms which are successfully applied for the English language, cannot be used for Odia documents due to the inherent nature of Odia scripts. The structure of Roman script and Odia script are different in nature. Different algorithms have been proposed by the researchers for line segmentation earlier. Here few segmentation strategies that have been adopted for Telugu documents are cited as Odia script has lots of similarity with Telugu script [8].

A projection profile is a well-known method for segmentation. In the plotted projection profile, the peaks and valleys of the image are shown. The zero-valued valleys indicate a white line or gap between two lines. When the lines are well separated, the projection profile gives a satisfactory result, whereas in many cases, it results splitting and overlapping. In detail, it has been discussed and demonstrated in the result section.

Kopullu et al. [1] generated connected components from document images for segmentation. The pixels that are connected are labeled with the same blob and then the blob is extracted from the image. They have tested their method on 465 different Telugu documents. It shows good results for high-quality documents but the output degraded with quality. For Odia documents, it will separate the untouched modifiers from the base, thus recognition complexity will increase. Swamy et al. [2] combines the projection profile and connected component for segmenting lines, words as well as characters. It shows good results for high-quality documents but gives segmentation error for touching lines and broken characters. A method using a modified histogram obtained from run-length smearing is proposed by N Priyanka [3]. This method has been verified for different Indic language documents like Telugu, Bengali, Devanagari, Kannada and also for multilingual documents. Kopullu and Negi [4] developed a robust method for text line segmentation using the Fringe map where a fringe map is generated for the input binary image; between text line, peak fringe number is located to construct region between adjacent text lines. Then the segmented path is generated by joining peak fringe number. This method has been tested on 234 images and resulted in 97% accuracy. Tripathy and Pal [5] have proposed a water reservoir based segmentation approach for extracting text lines of unconstrained Odia handwritten documents. Then to extract words, vertical projection profile and structural features of Odia characters are taken into account. Then using structural, topological and water reservoir based features, characters are extracted. Senapati et al. [6] have worked on text line segmentation of printed Odia documents where they have considered the white space between two consecutive lines. For word segmentation, the image is transposed to invert row and column and the same line extraction algorithm is repeated. This method works only in a case where lines are well separated. Segmentation for Odia documents is yet to achieve some milestone.

III. Motivation

There is no substantial work done in the segmentation area in Odia language which is the major motivating factor to work in the same area.

The study of the projection profile output which is shown in the following figure (Fig. 7) motivates height based segmentation. The text line which is not splitted or overlapped is considered as a normal line. The study of the output generated by implementing the traditional projection profile method on the Odia documents, reveals that most of the text lines segmented are normal lines.

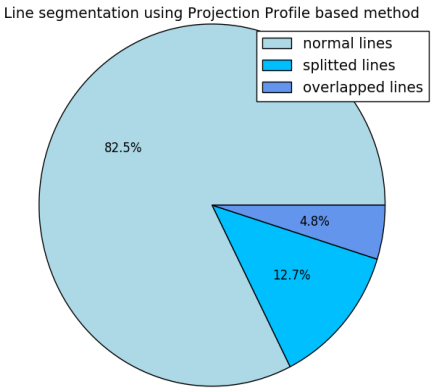

Fig. 7. Overall Line Segmentation Result for all the Documents using the Projection Profile-based Method.

In this case, more than 80% segmented lines are normal lines whereas around 12% segmented lines are broken and around 5% lines are overlapped. Also, document-wise line segmentation results in more normal lines in comparison to splitted or overlapped lines in most of the cases. So, the average height of a normal line can be found easily which will help verify if a line is broken or not. Also, it will help to find the overlapped region between two lines.

IV. Proposed Method

The proposed algorithm named as height_based_ segmentation segments text lines from printed Odia documents having a single column. A variety of documents; good and poor quality, have been collected from different sources. These documents are then converted to image form (jpg, bmp, png, tiff, etc.). The no. of lines present in these documents varies from 5 to 35. The algorithm considers an image document as the input, segments individual text lines and stores these as output. The output images are in jpg format. The segmentation is done based on the approximate height of a text line. So the height of the text line plays a very important role here.

As a part of pre-processing, input image is binarized applying Otsu's method. Here the background of the image document is white whereas the texts are written in black.

Consecutive nonzero rows (The row having at least one black pixel) in the image document form a text line. So counting the consecutive rows gives the height of the text line. These consecutive text lines may include a normal text line, overlapped lines or a broken line that contains the modifier zone. Finding the approximate height of a text line can help us to solve the problem of overlapped lines as well as a broken line.

The text line which is not broken or overlapped is considered as a normal line. To find the approximate height, the frequency of the height of the text lines is considered. If more normal lines are present in a document, the highest frequency is considered as the approximate height of the text line. To confirm the above, the second highest frequency of height is found. If the ratio of the second-highest frequency of height and highest frequency of height is more than 3, then there are more broken lines in the document. (The height of the upper or lower modifier which is separated from the base zone of the text line usually occupies one-third of the height of the base zone.). Here the second highest frequency of height is considered as the approximate height of the normal text line.

When the ratio of the highest frequency of height and second highest frequency of height is more than 2, then there are more overlapped lines in the document. So the second highest frequency of height is considered as the approximate height of the normal text line.

Now for each identified text line, collect the first nonzero row no. Find if the text line is overlapped, broken line or normal line from its height. If it is a normal line, extract the sub-image from the document image within the range (first non-zero row and first non-zero row+ height of the text line).

If the text line is found to be overlapped, the quotient value obtained from the height of the text line and approximate height of the line will give the no. of text lines overlapped. To extract the individual text lines from the overlapped text line, the following code snippet is used.

i) **threshold**=8 *// threshold value 8 gives the best result for this experiment*

ii) **rem**=height of text line% **Approx._height**

q=height of text line/ **Approx._height** *//maximum q no. of text lines is overlapped*

r= **Approx._height** /3 *//r is the approx. height of the upper or lower modifier.*

iii) if (**Approx._height** - rem <= r):

/(all those lines whose height is greater than **Approx._height**, doesn't contain an overlapped line. e.g.: calculated approx. height is 18. But, the height of a text line may be slightly more or less than 18.) */*

iv) **starting_point**=first_nonzero_row

v) for all the rows in the range (**starting_point**+**Approx_height**-**r**, **starting_point** + **Approx_height** + **r**):

vi) if a row has less pixel than **threshold**:
mark this row **op** as an overlapping row.

vii) Extract the sub-image from the document between **starting_point** and **op**.

viii) **starting_point**=**op**+1. */* The next text line starts from the next row after op/**

viii) *repeat step (iv) to (viii) for q no. of times. To extract q text lines.*

After extracting images of all the text lines from the document image, post-processing is applied to solve the broken line problem. All the extracted images are saved sequentially in a folder. The height of each text line is computed. If the ratio of the approximate height and the height of the text line is found to be around 3, then it is identified as

an upper or lower modifier zone of the text line. It should be merged with the base zone of the text line. Using the algorithm which is mentioned in paper [7], it is identified either as a lower or upper modifier zone image. If the image is found to be an upper modifier zone, then it is merged with the next image in the folder which is supposed to be the image of the base zone. If the image is identified as the lower modifier zone, then it is merged with the previous image in the folder which is supposed to be the image of the base zone.

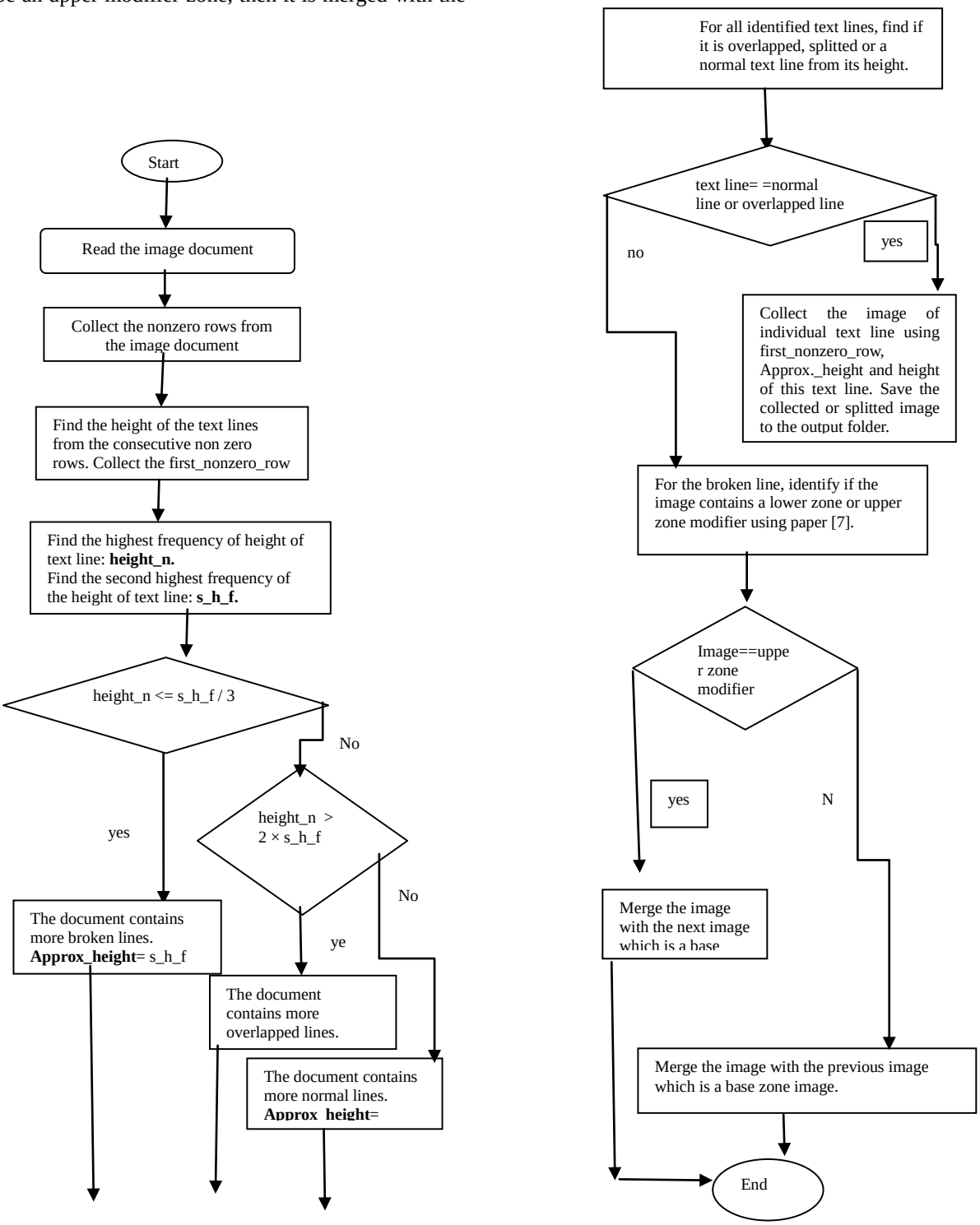

Fig. 8. Flow chart for the height_based_segmentation algorithm:

V. Experimental Result

The Experiment is carried out on a variety of printed Odia books having different font sizes and font types. Few documents are of very high quality and well-spaced whereas many books are of degraded printing quality. There are cases in which the font of the chapter heading is much larger than the body.

The pdf formats of the books are collected from different sources. To maintain heterogeneity, books published in different time period have been chosen. Pages that are considered for the experiment are free from all sorts of graphical contents; contain only textual data. The books contain writing only in a single column.

Then the pdf forms of the books are converted to image documents. These image documents are the input for our segmentation system. Here, a total of 265 documents are considered for the experiment. No. of text lines present in these documents varies from 5 to 40.

The proposed algorithm height_based_segmentation is implemented in Python numpy and scipy package. The performance of the algorithm is evaluated with the ground truth manually as well as with the popular projection profile-based method.

From the result (which is demonstrated in above graph Fig. 8 and Table I), it can be observed that using height_based_segmentation algorithm, broken lines have been reduced by 96% whereas overlapped line cases have been reduced by 71%. Training more varieties of modifiers may reduce more broken lines.

Using a horizontal projection profile-based algorithm, around 26000 nos. of words are segmented from the segmented lines.

With these collected words and lines, two separate databases have been built. The line database contains nearly 6000 normal lines, 300 overlapped lines, and 800 broken lines. Similarly, the word database contains approximately 26000 words. The database contains data of varied font type and size. These databases will be available to the researchers who work in the related field by contacting the authors.

TABLE. I. Comparison of Projection Profile-based Method and our Algorithm

Segmentation method	No. of documents	No. of normal lines	No. of broken lines	No. of overlapped lines
Projection profile based method	265	5450	841	319
Our Algorithm	265	5993	26	61

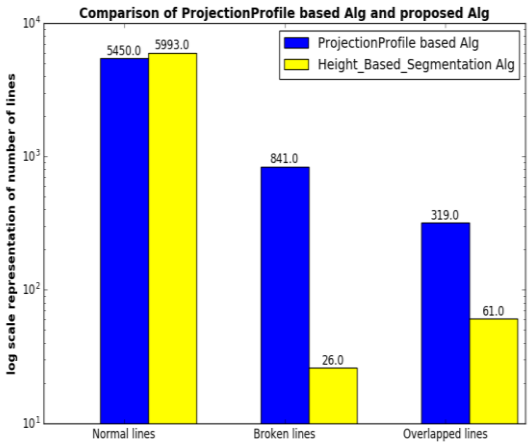

Fig. 9. Comparison of Projection Profile-based Algorithm and Proposed Algorithm.

VI. Conclusion

Line segmentation is a very important phase of the optical character recognition method. The accuracy of the whole system is largely dependent on line-segmentation accuracy. Here, an attempt has been made to experiment line segmentation based on the height of the normal lines as it is observed that comparatively few lines in a document suffer from broken-line or overlapped issues. Here, our new algorithm gives a ray of hope. The accuracy of our algorithm can be increased even more by correcting the skewness of the document. More variations of modifier symbols may be collected and trained to reduce more broken line problem.

References

[1] V. K. Koppula, N. Atul and U. Garain, "Robust Text Line, Word And Character Extraction from Telugu Document Image," Second International Conference on Emerging Trends in Engineering and Technology, Nagpur, 2009, pp. 269-272.

[2] Das, M. Swamy, and Dr. CRK Reddy. “Segmentation of Overlapping Text Lines , Characters in Printed Telugu Text Document Images.” 2010.

[3] Nallapareddy Priyanka, Srikanta Pal and Ranju Manda. Article:Line and Word Segmentation Approach for Printed Documents. IJCA,Special Issue on RTIPPR (1):30–36, 2010.

[4] V. K. Koppula and A. Negi, "Fringe Map Based Text Line Segmentation of Printed Telugu Document Images," 2011 International Conference on Document Analysis and Recognition, Beijing, 2011, pp. 1294-1298.

[5] N. Tripathy and U. Pal, "Handwriting segmentation of unconstrained Oriya text," Ninth International Workshop on Frontiers in Handwriting Recognition, Kokubunji, Tokyo, Japan, 2004, pp. 306-311.

[6] D. Senapati, S. Rout and M. Nayak, "A novel approach to text line and word segmentation on Odia printed documents," Third International Conference on Computing, Communication and Networking Technologies (ICCCNT'12), Coimbatore, 2012, pp. 1-6, 2012.

[7] S S Pattanaik, S K Pradhan, R C Malik, Printed Odia Symbols for character Recognition: a Database study, In the proceeding of 3rd Intl. conference on Advanced Computing and Intelligent Engineering,2018, Bhubaneswar, India.

[8] P Mohanty, On script Complexity and the Odia script, Dedicated to Gabriel Altmann on the Occasion of his 75th Birthday,2007.