Automatic Localization of Page Segmentation Errors

Dheeraj Mundhra[*]
IIT Kharagpur
Kharagpur, India
09MA2009@iitkgp.ac.in

Anand Mishra
IIIT Hyderabad
Hyderabad, India
anand.mishra@research.iiit.ac.in

C. V. Jawahar
IIIT Hyderabad
Hyderabad, India
jawahar@iiit.ac.in

ABSTRACT

Page segmentation is a basic step in any character recognition system. Its failure is one of the major causes for deteriorating overall accuracy of the current Indian language OCR engines. Many segmentation algorithms are proposed in literature. Often these algorithms fail to adapt dynamically to a given page and thus tend to yield poor segmentation for some specific regions or some specific pages. Given the ground truth, locating page segmentation errors is a straight foreword problem and merely useful for comparing segmentation algorithms. In this work, we locate segmentation errors without directly using the ground truth. Such automatic localization of page segmentation errors can be considered a major step towards improving page segmentation errors. In this work, we focus on localizing line level segmentation errors. We perform experiments on more than 18000 scanned pages of 109 books belonging to four prominent south Indian languages.

General Terms

Experimentation.

Keywords

OCR, document image segmentation, under-segmentation, over-segmentation, false alarm.

1. INTRODUCTION

The success of page segmentation algorithm critically affects the performance of OCR. Page segmentation algorithms are one of the widely studied topics in document image analysis literature (see [6], [14]). Most of these segmentation algorithms perform satisfactorily well but tend to fail in some specific region or for some specific pages. The main reason for such failures is that these algorithms are heavily dependent on parameters and thus fail to adapt to a given page dynamically.

Document image segmentation is a widely studied topic in literature. Consistent appearance of page segmentation work (see [2], [3], [4]) in competitions at ICDAR shows the interest of the community in this area. Kise *et al.* [8] had proposed a powerful Voronoi diagram based segmentation algorithm a decade ago. Nevertheless, the complexity and variations in the document images make the task of segmenting the given document page into lines still challenging. There is also interest in designing a hybrid segmentation algorithm [1] or designing segmentation algorithm by learning the page features [10].

Set theoretic approach of analysing segmentation algorithms was presented in [11]. In [14] performance of six most popular page segmentation algorithm were analysed. Sesh Kumar *et al.* [9] had done similar analysis on segmentation algorithm for Indian languages. All these works compared document image segmentation algorithms assuming the availability of the ground truth. Although such works do fair comparison of segmentation algorithms, but have several disadvantages: (1) They do not directly lead towards improvement in the segmentation algorithm (2) They do not specify why the errors arise? (3) They are not applicable for large scale evaluation of OCRs [15] due to unavailability of the ground truth.

Recently, in computer vision community researchers have shown interest in unsupervised evaluation of image segmentation algorithms [17]. Here for a given image, availability of the ground truth is not assumed. Rather a set of features are computed from the segmented image and based on these features performance of image segmentation algorithms are measured. A survey of unsupervised methods for evaluating segmentation algorithms is given in [17]. We are highly inspired by such methods. However, we do not do any evaluation of page segmentation algorithms in this work. Rather we go one step further and try to find out where and what type of page segmentation errors are present at line level for a given page. Such segmentation error localization can be considered a major step towards improvement in segmentation output. Once segmentation errors are localized automatically, one can either use human intervention or alternate segmentation algorithms for error correction. However, improving the segmentation accuracy is beyond the scope of this work. The primary objective of this work is to automatically locate segmentation errors with very high accuracy.

[*]This work was carried out when Dheeraj Mundhra was visiting IIIT Hyderabad.

Similar to [14], we focus on *line level* document image segmentation, where a segmentation algorithm partitions the text block into lines. Often these segmentation algorithms fail to segment lines correctly. In other words, each segmented line is either correctly segmented, over-segmented, under-segmented, false alarm or missing dangling modifier. The objective of this work is to locate these errors without the help of ground truth. (We use ground truth only for evaluation of proposed error localization scheme). We formulate the problem of locating page segmentation errors as a multi-class classification problem where each segmented line is classified into one of the five classes i.e., correct, under-segmented, over-segmented, false alarm or missing dangling modifier. For this we compute a set of features for few segmented lines assuming the availability of the ground truth for them. We then, compute the same set of features for the rest of the segmented lines to locate the segmentation errors in a classification framework.

We have shown segmentation error localization performance on a specific segmentation algorithm. However, proposed method is independent of segmentation algorithm. We used 109 books in four prominent south Indian languages for our experiments, where randomly selected half of the pages were used for training whereas rest half was used for testing. To evaluate the performance of proposed method, we used ground truth based comparison. The proposed scheme localizes the segmentation errors with more than 78% accuracy, which means that we are able to localize more than seventy eight percentage of the total page segmentation errors automatically.

The reminder of the paper is organised as follows. In Section 2 various types of page segmentation errors are described. In Section 3, we formulate the problem of locating page segmentation errors as a classification problem. Here we describe features used to learn various types of page segmentation errors. Section 4 describes experiments and results. We finally conclude our work in Section 5.

2. PAGE SEGMENTATION ERRORS

There are large number of document segmentation algorithms available in literature. The popular ones are Recursive XY cut [12], White-space analysis [5], Docstrum [13], Voronoi diagram based [8], and RLSA [16]. Description of these algorithms is not in the scope of this paper. However, readers are encouraged to see [14] for the description of these segmentation algorithms. Most of these segmentation algorithms suffer from some or other page segmentation errors. These errors can be defined in a set theocratic notion as in [14]. We summarize these definitions here.

Let S and G be the set of lines denoting segmentation output and ground truth respectively. Then we can define segmentation errors as follows:

Correct.

A Line $B \in S$ is said to be correct if there exists a unique line $A \in G$ such that $A \cap B$ is significant.

Over-segmented.

A line $B \in S$ is said to be over-segmented if there exist at-least one more line $B^{'} \in S$ and $A \in G$ such that both $A \cap B$ and $A \cap B^{'}$ are significant.

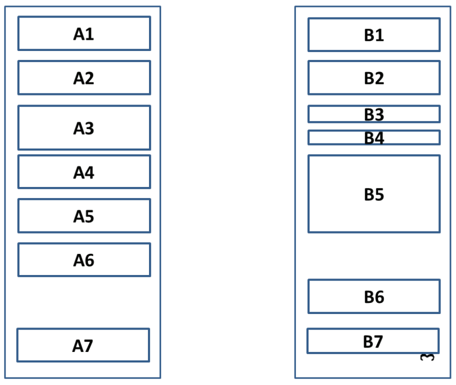


Figure 1: Ground truth and segmented page.

Under-segmented.

A line $B \in S$ is said be under-segmented if there exist multiple lines A's in G such that $A \cap B$ is significant.

Missing component.

A line $B \in S$ is said to be missing component if there exists a unique line $A \in G$ such that $A \cap B$ is not significant. In other words, by calling line A as missing component we mean that line A has missed some dangling modifier either above or below the line. This error is very common in Indian language document image segmentation. (Note that this error is not defined in [14]).

False alarm.

A line $B \in S$ is said to be a false alarm if there does not exist any line $A \in G$ such that $A \cap B \neq \phi$.

Missed line.

A line $A \in G$ is said to be a missed line if there does not exist any line $B \in S$ such that $A \cap B \neq \phi$.

We demonstrate typical examples of page segmentation errors in Figure 1. Here lines B_1 and B_2 are correctly segmented. B_3 and B_4 are over-segmented. Line B_5 is under-segmented. Line B_6 is a false alarm whereas A_6 is a missed line. Line B_7 is missing component as it is missing a small dangling modifier which is just below the line. In this work, for a given segmented page we compute certain features (which we describe in next section) and locate first five segmentation errors in a given page in multi-class classification framework. For learning the features we assume the availability of the ground truth for training images, whereas for a test image we compute same set of features for every line. We then locate the errors by classifying each line as either correct, over-segmented, under-segmented, false alarm or missing component.

3. THE PROBLEM OF LOCATING PAGE SEGMENTATION ERRORS

More often the existing page segmentation algorithms tend to fail for some specific pages or some specific regions of the page. Figure 2 shows some typical examples of failure at line level segmentation. Given a segmented page, our goal is to locate page segmentation errors. Once segmentation errors are localized, one can use human intervention or alternate

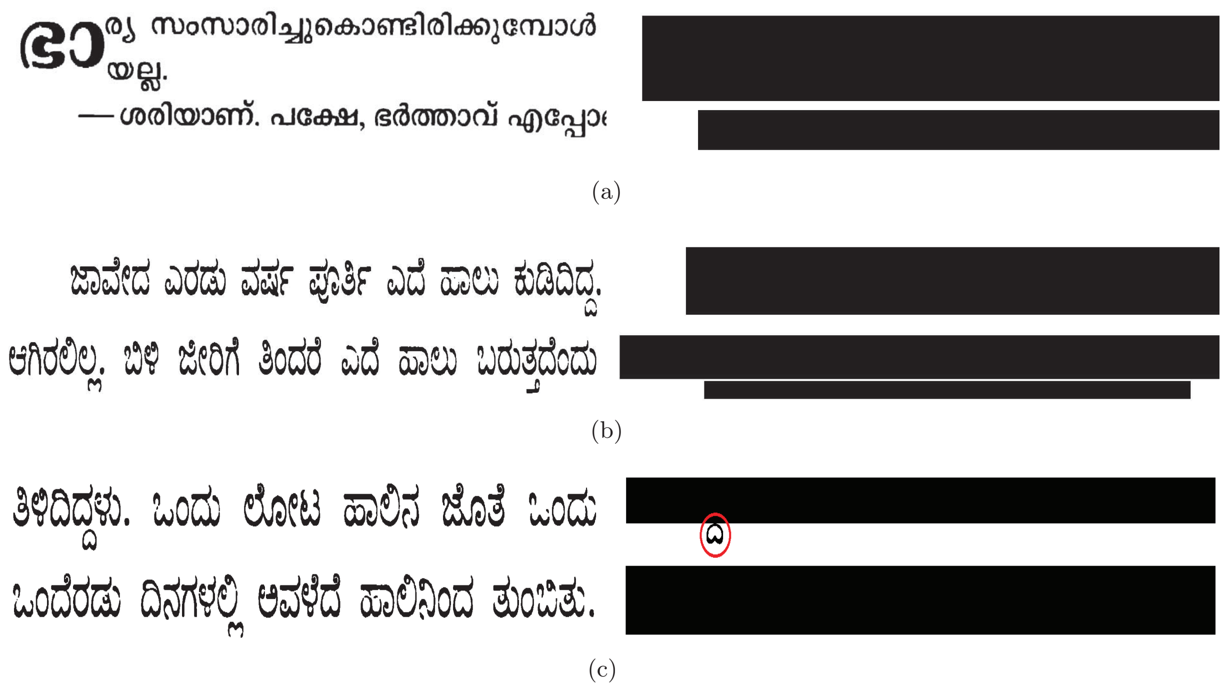


Figure 2: Typical segmentation errors: left and right columns show part of a sample page and corresponding segmented output respectively. (a) Two lines are merged into one line (under-segmentation) (b) One line is spilt into two lines (over segmentation) (c) A dangling modifier shown in a small red circle is missed (missing component).

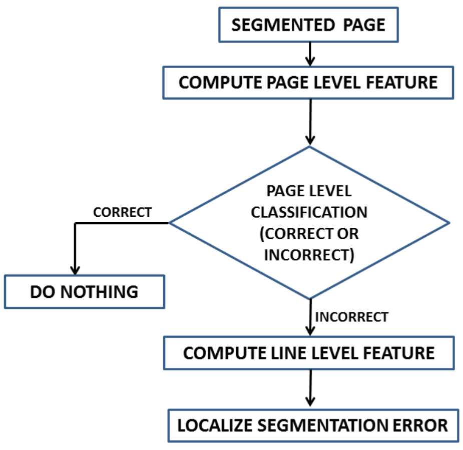

SEGMENTED PAGE
COMPUTE PAGE LEVEL FEATURE
PAGE LEVEL CLASSIFICATION (CORRECT OR INCORRECT)
CORRECT
DO NOTHING
INCORRECT
COMPUTE LINE LEVEL FEATURE
LOCALIZE SEGMENTATION ERROR

Figure 3: Process of locating segmentation errors.

algorithm with tuned parameters for error correction. However, improving the segmentation accuracy is beyond the scope of this paper. We rather work on segmentation error localization. We do it in two stages. Figure 3 demonstrates the process of error localization. In stage-1 we compute some page level features and classify each page as correct or erroneous (over, under, false alarm, or missing component). In erroneous pages we compute line level features and classify each line either as correct, over-segmented, under-segmented, false alarm or missing component. Note that we learn both line level and page level features in supervised learning framework i.e. we assume availability of ground truth for the training images so that we can learn line/page level features and corresponding error. However, for the test images we locate the errors using nearest neighbour based classification.

The training and testing phase of locating page segmentation errors can be summarized as follows:

Page level.

Learning phase:

1. Compute page level features for each page of training document image.
2. Assign correct or incorrect label to each page using ground truth. (Note that by correct label of a page we mean all the lines in that page are correctly segmented)

Testing phase:

1. Compute page level features for each page of test document image.
2. Classify each page into correct or incorrect using k-nearest neighbour based classification.

Line level.

Learning phase:

1. Compute line level features for each line of training document image.
2. Assign correct or error labels to each line using ground truth.

Testing phase:
1. Compute line level features for each line of a test document image.
2. Classify each line into correct, over-segmented, under-segmented, false alarm or missing component using k-nearest neighbour based classification.

The motivation behind two stages classification is that we want to avoid extra computation for correct pages. We achieve this by using a set of simple features in stage-1 where we classify correct and incorrect pages, and in the stage-2 we compute computationally more expensive line level features only for the pages which are classified as incorrect by stage-1.

To evaluate the performance of our system we first locate all the errors using ground truth as in [14]. We then, compare our results of error localization with it and report percentage of errors which we are able to localize automatically.

3.1 Features

We observe that (1) most of the characters in a page are of same size, font and style, (2) line spacing within the documents are mostly same, (3) page is formatted uniformly within a book, (4) two nearby lines in a document is mostly of same height. We use these concepts to design a set of features at page level. These features are powerful enough to decide whether a page is correctly segmented or not by looking into the segmented page. The features we use for classifying segmented page as correct or incorrect i.e., stage-1 classification are as follows:

f1: Maximum line height.
Given a segmented page with set of lines $= \{L_1, L_2, ..., L_n\}$ we define feature f1 as:

$$f1 = max\{LH_1, LH_2, ..., LH_n\},$$

where LH_i is the height of i^{th} line L_i

f2: Minimum line height.
Given a segmented page with set of lines $= \{L_1, L_2, ..., L_n\}$ we define feature f2 as:

$$f2 = min\{LH_1, LH_2, ..., LH_n\},$$

where LH_i is the height of i^{th} line L_i

f3: Difference of maximum and average line height.
Feature f3 is defined as:

$$f3 = f1 - \frac{1}{n}\sum_{i=1}^{n} LH_i.$$

f4: Difference in average and minimum line gap.
Let $LG_{i-1,i}$ be the line gap between lines L_i and L_{i-1}, then we define f4 as:

$$f4 = \frac{1}{n-1}\sum_{i=2}^{i=n} LG_{i-1,i} - min\{LG_{1,2}, LG_{2,3}, ..., LG_{n-1,n}\}.$$

f5: Maximum of difference in line heights and line gap.
Let LH_{i-1}, LH_i and LH_{i+1} be the line heights of line i-1, i and i+1 respectively. Further, suppose $LG_{i-1,i}$ and $LG_{i,i+1}$ are the gaps between lines i-1 and i, and i and i+1 respectively. Then we define feature f5 as:

$$f5 = max\{|LH_{i-1} - LH_i| - LG_{i-1,i}\} \ \forall i \in \{2, 3, ..., n\}.$$

f6: Maximum area of connected component between lines.
We find out all connected components (CCs) between lines which are not part of any line. We compute the area of all CCs and take the maximum area as a feature.

f7: Minimum area of connected component between lines.
Similarly we take the minimum area of CCs as a feature.

f8: Maximum white width in vertical profile of the page.
We compute the vertical profile for the page and in that vertical profile find out longest run of zero. We use this as a feature f8. This helps us to identify if two multi-column lines are merged.

In stage-2, we wish to locate page segmentation errors, i.e. for each segmented line in a segmented page we have to classify it as correct or erroneous. This is a tougher task than stage-1, thus we need powerful and error specific features for these stage. However, this will not affect the overall computation of the error localization, as we locate error only for those pages which are classified as incorrect by stage-1. The set of features which we compute for each line L_i in order to classify it as correct, over-segmented, under-segmented, false alarm or missing component are as follows:

$F1$: Difference in line heights and line gap.
We define this feature as follows:
$F1 = max\{|LH_{i-1} - LH_i| - LG_{i-1,i}, |LH_i - LH_{i+1}| - LG_{i,i+1}\}$,
where LH_i is the height of line i and $LG_{i-1,i}$ is the gap between lines $i-1$ and i. The intuition behind this feature is two closest line should be of similar height.

$F2$: Difference in line height and maximum height of connected component.
We use difference of line height and maximum height of connected component in a line as a feature. This helps us to locate under segmentation where line height is far greater than size of maximum connected component.

$F3$: Maximum area of CCs closest to line.
To define this feature for a line L_i we find out the CCs which are not part of any line and is the closest to line L_i compared to its above or below lines i.e., lines L_{i-1} and L_{i+1}. We compute the area of all such CCs and take maximum area as a feature F3 for line L_i

$F4$: Maximum word gap in line.
In case of multicolumn some segmentation algorithms merge two horizontal lines. To identify such case we compute maximum word gap in line and use this as feature F4

Language	tot.lines	overseg	underseg	m.c.	f.a.
Telugu	123493	6.47	0.78	4.8	0.89
Tamil	144215	2.74	3.3	1.1	1.43
Malayalam	181951	0.41	0.43	1.21	0.72
Kannada	114468	4.42	11.49	3.6	4.95
Total	**564127**	**3.14**	**3.48**	**2.45**	**1.8**

Table 2: Percentage of lines having error estimated using ground truth. This shows that around 11% of the total segmented lines have some segmentation error.

F5: Maximum area of connected component in a line.
In every line we compute the maximum area of connected component and use it as a feature. Very high and low value of this feature correspond to false alarm.

F6: Minimum of upper and lower line gaps.
We define feature F6 for the line L_i as:

$$F6 = min\{LG_{i-1,i}, LG_{i,i+1}\}.$$

This feature helps us to locate over-segmentation where one of the gap $LG_{i-1,i}$ or $LG_{i-1,i}$ is very low.

4. EXPERIMENTS AND RESULTS

4.1 About Dataset

We use a dataset [7] of 109 books in four prominent south Indian languages for all our experiments. Table 4 gives the details of the dataset. This dataset contains pages scanned in 600 dpi. Some sample pages of this dataset can be seen in Figure 4. We also have line level annotation in form of XML for this dataset produced using a semi-automatic tool.

Segmentation of Indian language document pages is a challenging task, mainly due to (1) Presence of dangling modifiers (2) The relative position of the neighbouring characters are not fixed etc. In [9], authors experimentally show that many well-known segmentation algorithm perform poorer in case of Indian languages compared to English, which also makes automatic error localization an important task.

For performance evaluation of our system, we need to compare our error localization with one obtained using ground truth. Moreover, we also use ground truth based evaluation in learning phase of our system. Thus we do our experiment in two phase. In phase-1, we localize the segmentation errors with the help of ground truth. In phase-2, we learn the ground truth based error localization for the training images. However, for test images we automatically localize errors using nearest neighbour based classification.

4.2 Error localization using ground truth

We first run the segmentation algorithm on all the pages. With the help of ground truth we locate all the segmentation errors and store line co-ordinates along with corresponding error type in the database. Further, if all the lines in a page are correctly segmented we tag that page as correct. Similarly, a page having line level segmentation error, is tagged as incorrect page. Table 2 summarizes segmentation errors at line level caused by segmentation algorithm. We see that on average around 11% of the segmented lines are not correct. Which is actually a very large quantity considering the huge size of the dataset. Our aim is to localize these errors automatically i.e. without using ground truth.

Language	ρ_{cc}	ρ_{ii}	ρ_p
Telugu	76.30	94.52	91.92
Tamil	87.48	91.12	89.90
Malayalam	85.66	84.55	85.06
Kannada	89.49	93.84	92.76
Average	**84.42**	**90.73**	**89.66**

Table 3: Accuracy of classifying pages as correct/incorrect. ρ_{cc}, ρ_{ii} and ρ_p denotes how accurately we classify correct as correct, incorrect as incorrect and overall accuracy at page level.

Language	overseg	underseg	m.c.	f.a.	ρ_l
Telugu	89.67	64.48	80.64	52.82	82.68
Tamil	84.65	91.65	44.44	79.83	83.84
Malayalam	79.30	99.83	52.24	76.62	71.68
Kannada	59.11	88.07	62.46	77.50	78.55
Average	**79.63**	**88.09**	**62.46**	**70.62**	**78.51**

Table 4: Percentage of segmentation errors we automatically detect (Proposed Scheme). Here ρ_l denotes the overall error localization performance.

4.3 Automatic error localization

4.3.1 Page level

In this stage, for a given segmented page, we say whether the page is correctly segmented or not. For this we use half of the pages as training and rest half as testing. For training images we compute page level features as described in Section 3 and learn the correctness of the page from ground truth, and for testing images we use k-NN based classifier to decide whether the page is correct or not. (Note that by correct page we mean those pages which are perfectly segmented at line level).

This step is important to avoid unnecessary computation for the correctly segmented pages, as we can work on only incorrect pages for line level error localization.

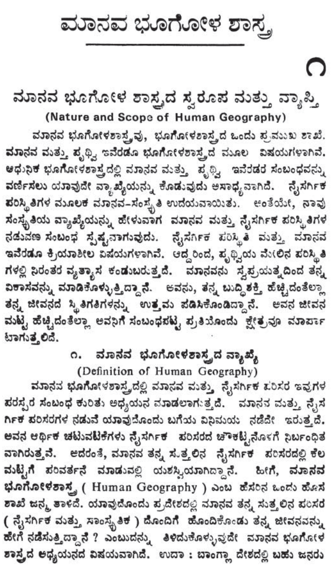
The task of tagging pages as correct/incorrect is performed in classification framework. Thus we define a set of performance measures using confusion matrix. Let C be a confusion matrix where row 0 and 1 corresponds to correct and incorrect classification respectively. (Recall that the entry C_{ij} corresponds to total number of lines classified as class j but are actually belonging to class i). We define correct as correct(ρ_{cc}), incorrect as incorrect (ρ_{ii}) and overall classification accuracy (ρ_p) as follows:

$$\rho_{cc} = \frac{C_{00} \times 100}{C_{00} + C_{01}}$$
$$\rho_{ii} = \frac{C_{11} \times 100}{C_{10} + C_{11}}$$
$$\rho_p = \frac{(C_{00} + C_{11}) \times 100}{C_{00} + C_{01} + C_{10} + C_{11}}$$

Table 3 describes how accurately we say correct pages as correct and incorrect pages as incorrect based on above measures. We see that we are able to classify correct page as correct and incorrect page as incorrect with more than 89% accuracy.

Language	Number of books	Number of pages	Number of lines in ground truth
Telugu	23	4872	131362
Tamil	28	4722	146504
Malayalam	31	5020	183606
Kannada	27	3689	124105
Total	**109**	**18303**	**585577**

Table 1: Details of dataset used for the experiment.


(a)

(b)

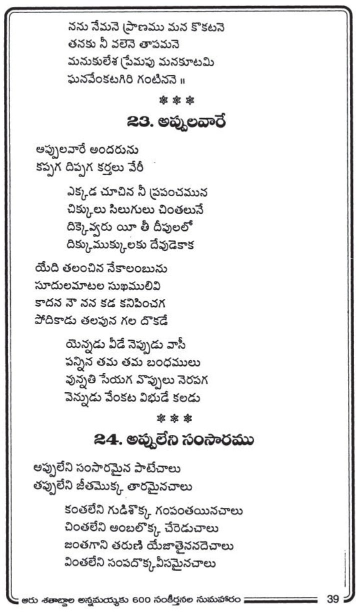

(c)

Figure 4: Sample images from the dataset in three different languages: (a) Kannada (b) Malayalam (c) Telugu.

4.3.2 Line level

At line level error localization, we classify each segmented page as correct or one of the segmentation error. We use 50% of the lines randomly as a training set and report error localization on the rest of the lines. The k-NN based classification with k=5 is used in our experiments. We use three random splits of the data and report the average accuracy of error localization. We do these experiments language wise. To measure the performance of line level error localization we define a performance metric using confusion matrix. Let C be a confusion matrix showing confusion among correct, over-segmentation, under-segmentation, missing component and false alarm. Further, suppose rows $\{0, 1, 2, 3, 4\}$ of C correspond to correct, over-segmentation, under-segmentation, missing component and false alarm respectively. Then we define overall error localization accuracy at line level as follows:

$$\rho_l = \frac{\sum_{i=1}^{4} C_{ii} \times 100}{\sum_{i=1}^{4}\sum_{j=0}^{4} C_{ij}}$$

This measure gives percentage of segmentation errors which we are able to detect automatically. Here $j = 0$ to 4 signifies that we also consider confusion among erroneous and correct lines while computing overall error localization accuracy. Table 4 describes the percentage of error we correctly locate. We observe that on average we are able to locate more than 78% of the page segmentation errors correctly. Which is a significant result, as our error localization is automatic and thus one can use manual intervention or alternate segmentation algorithms to correct the segmentation errors.

Further, since we localize the errors in a classification framework, where each line is classified as correct or any of the segmentation error. It is also important to describe what percentage of correct lines we classify as incorrect. Our experimental results show that on average only 0.19%, 0.30%, 0.10% and 0.07% of correctly segmented lines are located as under-segmented, over-segmented, missed component and false alarm respectively. In other words, proposed scheme also locates the correct lines as correct with very high accuracy.

We also demonstrate qualitative results of the proposed method in Figure 5. We observe that we are able to locate all type segmentation errors very accurately, except for

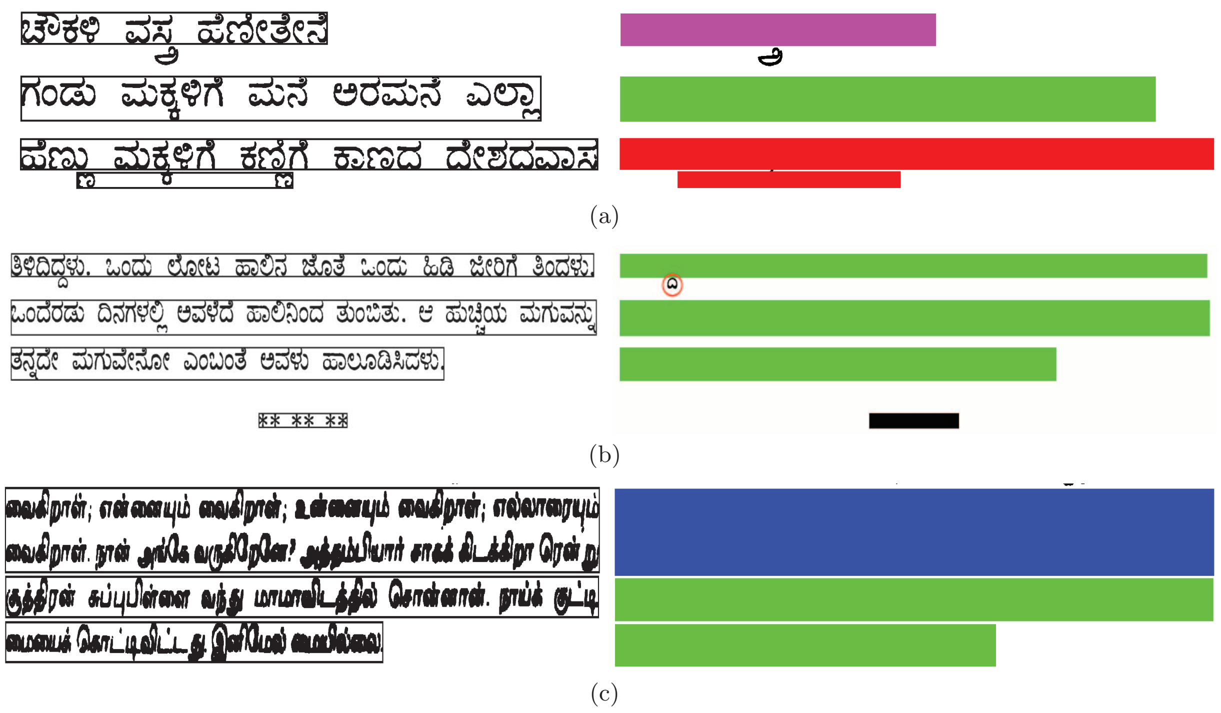


Figure 5: Error localization based on proposed scheme. Left and right columns represent segmentation output and our error localization results respectively. Green, blue, red, pink and black colours represent automatic localization as correct, under-segmented, over-segmented, missing component and false alarm (best viewed in colour). (Note - In Figure (b) the first line is incorrectly located as correct. Although it is actually missing a small dangling modifier shown in a red circle just below the line).

the Figure 5(b), where a line which is missing a component (shown in tiny red circle) is localized as correct.

In summary, we compute a set of features at line level for the training data (where we assume availability of ground truth). For test data we compute the same set of features for every line and apply k-NN based classification to locate segmentation errors. In this framework, every line is either classified as correct, over-segmented, under-segmented, missing component or false alarm. Thus, we are able to localize the segmentation errors without ground truth with a very high accuracy. (Note that we use ground truth only for the performance evaluation of the proposed method).

5. CONCLUSIONS

In this work, we address the problem of localizing page segmentation errors. The proposed scheme is able to locate segmentation errors without ground truth with high accuracy. Such error localization is very important for segmentation error correction which can be done either by manual intervention or running alternate segmentation algorithms in the error localized part. Further, the proposed error localization scheme is independent of segmentation algorithms. Future direction of this work is to design segmentation post-processor to automatically correct page segmentation errors. Finally, we conclude our work by stating that *we are able to automatically locate more than 78% of the page segmentation errors without using ground truth.*

6. ACKNOWLEDGEMENTS

This work is supported by Ministry of Communication and Information Technology, Government of India, New Delhi.

7. REFERENCES

[1] M. Agrawal and D. S. Doermann. Voronoi++: A dynamic page segmentation approach based on voronoi and docstrum features. In *ICDAR*, pages 1011–1015, 2009.

[2] A. Antonacopoulos, D. Bridson, and B. Gatos. Page segmentation competition. In *ICDAR*, pages 75–79, 2005.

[3] A. Antonacopoulos, B. Gatos, and D. Bridson. Page segmentation competition. In *ICDAR*, pages 1279–1283, 2007.

[4] A. Antonacopoulos, S. Pletschacher, D. Bridson, and C. Papadopoulos. Icdar 2009 page segmentation competition. In *ICDAR*, pages 1370–1374, 2009.

[5] H. S. Baird, S. E. jones, and S. J. Fortune. Image segmentation by shape-directed covers. In *ICPR*, 1990.

[6] V. Govindaraju and S. Setlur. *Guide to OCR for Indic Scripts*. Springer, 2009.

[7] C. V. Jawahar and A. Kumar. Content-level annotation of large collection of printed document images. In *ICDAR*, pages 799–803, 2007.

[8] K. Kise, A. Sato, and M. Iwata. Segmentation of page images using the area voronoi diagram. *Computer Vision and Image Understanding*, 70:370–382, 1998.

[9] K. S. S. Kumar, S. Kumar, and C. V. Jawahar. On segmentation of documents in complex scripts. In *ICDAR*, pages 1243–1247, 2007.

[10] K. S. S. Kumar, A. M. Namboodiri, and C. V. Jawahar. Learning to segment document images. In *PReMI*, pages 471–476, 2005.

[11] S. Mao and T. Kanungo. Empirical performance evaluation methodology and its application to page segmentation algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(3):242–256, 2001.

[12] G. Nagy, S. C. Seth, and M. Viswanathan. A prototype document image analysis system for technical journals. *IEEE Computer*, 25(7):10–22, 1992.

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

[14] F. Shafait, D. Keysers, and T. M. Breuel. Performance evaluation and benchmarking of six-page segmentation algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(6):941–954, 2008.

[15] L. Vincent. Google book search: Document understanding on a massive scale. In *ICDAR*, pages 819–823, 2007.

[16] K. Y. Wong, R. G. Casey, and F. M. Wahl. Document analysis system. *IBM Journal of research and development*, 1982.

[17] H. Zhang, J. E. Fritts, and S. A. Goldman. Image segmentation evaluation: A survey of unsupervised methods. *Computer Vision and Image Understanding*, 110(2):260–280, 2008.