Hybrid Page Segmentation with Efficient Whitespace Rectangles Extraction and Grouping

Kai Chen, Fei Yin, Cheng-Lin Liu
National Laboratory of Pattern Recognition (NLPR)
Institute of Automation of Chinese Academy of Sciences
95 Zhongguancun East Road, Beijing 100190, P.R. China
Email: {kchen, fyin, liucl}@nlpr.ia.ac.cn

***Abstract*—Page segmentation is still a challenging problem due to the large variety of document layouts. Methods examining both foreground and background regions are among the most effective to solve this problem. However, their performance is influenced by the implementation of two key steps: the extraction and selection of background regions, and the grouping of background regions into separators. This paper proposes an efficient hybrid method for page segmentation. The method extracts whitespace rectangles based on connected component analysis, and filters whitespace rectangles progressively incorporating foreground and background information such that the remaining rectangles are likely to form column separators. Experimental results on the ICDAR2009 page segmentation competition test set demonstrate the effectiveness and superiority of the proposed method.**

***Keywords*—Page segmentation, whitespace rectangles extraction, whitespace rectangles grouping.**

I. Introduction

Document image understanding systems find wide applications in document conversion and information retrieval. Page segmentation occupies an important position in those systems. The correctness of segmenting page into blocks greatly affects the overall recognition performance. After text lines identification and reading order determination, the text line images are fed to an OCR engine to derive the output text.

Many methods of page segmentation have been reported in the literature. They can be categorized into foreground analysis, background analysis and hybrid ones from the viewpoint of objects to be analyzed.

Foreground analysis methods collect black pixels to form document components (connected components, words, text lines and so on) recursively. The representatives of this category include *Docstrum* [1], *Minimal Spanning Tree* [2] and *Document Representation* [3]. Background analysis methods use straight or curved lines of white pixels to segment the page. Sometimes regions of white pixels are used. These methods include *X-Y Cut* [4], *Maximal Empty Rectangles* [5], *Whitespace Thinning* [6] and *Voronoi Diagram* [7]. Hybrid methods analyze both foreground and background regions. Pavlidis and Zhou [8] introduced a method which identifies column gaps and group them into column separators after horizontal smearing of black pixels. They target documents with some constraints. The Jouve method submitted to ICDAR2011 Layout Analysis Competition [9] first extracts connected components and group them into words. Whitespace rectangles aligned at the end of words are then aggregated into separators. In this method, the number of whitespace rectangles is large and how to group them is not specified in the brief description.

Hybrid methods overcome the error accumulation from low-level components by foreground analysis and the under-segmentation of background analysis due to its ignorance of foreground components homogeneity. However, it is not trivial to extract proper background regions and to group them into separators accurately. There is not a feasible general approach to achieve this goal.

This paper proposes a hybrid page segmentation method that extracts proper whitespace rectangles based on connected component (CC) analysis, and filters whitespace rectangles progressively incorporating foreground and background information, such that the remaining rectangles can be grouped into column separators using simple rules. The effectiveness of the method was demonstrated on the ICDAR2009 page segmentation competition test set.

II. Hybrid Segmentation Method

The proposed page segmentation method is based on whitespace rectangles extraction and grouping. It consists of three major steps: whitespace rectangles extraction, filtering and grouping. The block diagram of our method is shown in Fig. 1. First the foreground CCs are extracted and linked into chains according to their horizontal adjacency relationship. Whitespace rectangles are extracted from the gap between horizontally adjacent CCs, and are progressively filtered according to the comparison of rectangle width and adjacency relationship with text lines. The remaining rectangles are grouped into separators, and non-viable separators are filtered out heuristically. Large CCs are then merged with text lines. Last, text blocks are formed by grouping text lines and ordered.

For illustrating the steps subsequently, we define vertical overlap and horizontal overlap as shown in Fig. 2. Two components are vertically overlapping if their vertical spans overlap. They are horizontally overlapping if their horizontal spans overlap.

A. Whitespace rectangles extraction

A column separator has CCs on both left and right sides. Therefore, we extract all the whitespace rectangles between horizontally adjacent CCs as candidates, which are then filtered progressively.

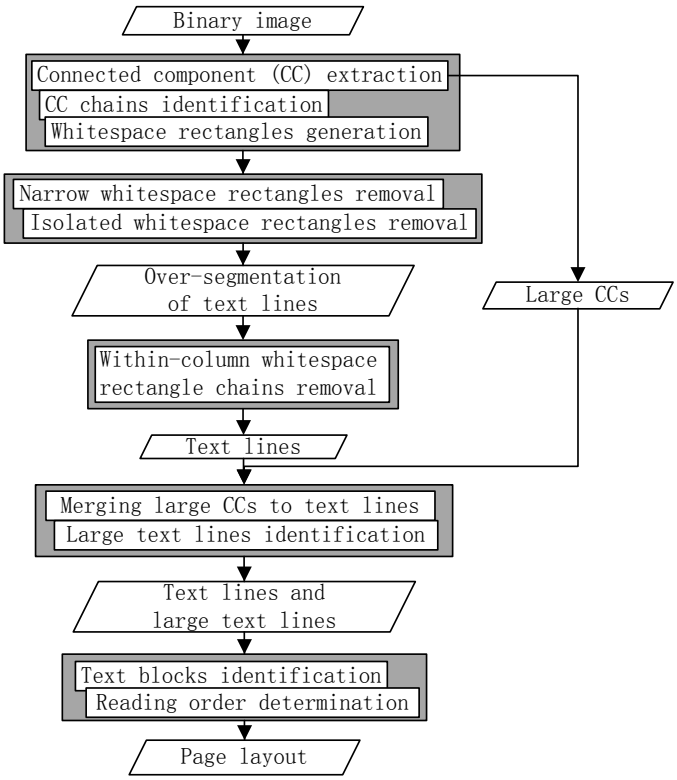
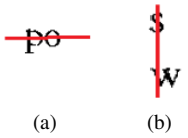


Fig. 1. Block diagram of the proposed algorithm.



(a) (b)

Fig. 2. Examples of vertically overlapping components (a) and horizontally overlapping components (b).

We use a linear-time algorithm [10] to extract CCs and store the coordinates (x_min, x_max, y_min, y_max) of the bounding boxes of the CCs. A CC may constitute large titles or graphics if its long side is greater than one tenth of the long side of the page. We call such a CC a large one and put it aside to be processed later.

To facilitate whitespace rectangles filtering, we link the CCs into horizontal chains considering that the CCs in a text line are mostly horizontally aligned (vertically overlapping). To do this, we find the right nearest neighbor (RNN) for each CC. A CC and its RNN overlap vertically (Fig. 3), and are

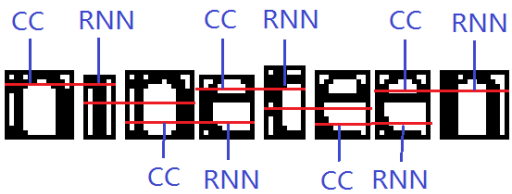


Fig. 3. Examples of CCs and their RNNs.

linked into the same chain. To find RNNs of all the CCs straightforwardly requires $O(N^2)$ computation. This can be reduced substantially by first sorting all the CCs in ascending order of their y_min. Then for each CC C, we compare it with the subsequent CCs. Once C encounters a CC S that does not vertically overlap with C, the CCs after S cannot be RNN of C because they do not overlap with C vertically. After linking every CC to its RNN, the CC with no left neighbor becomes the start of a chain, which ends at a CC that has no RNN. The link path from a start CC to an end forms a CC chain. The length of a CC chain is defined as the span from the left bound of the start CC to the right bound of the end CC. After all CC chains are identified (Fig 6(a)), whitespace rectangles are generated between every CC and its RNN inside each CC chain (Fig 6(b)).

B. *Whitespace rectangles filtering*

A CC chain may contain both within-column and between-column rectangles (Fig. 4). Most within-column rectangles can

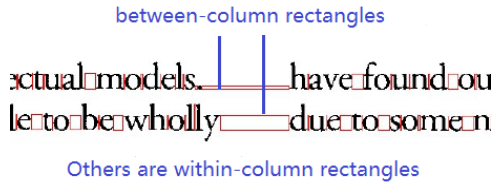


Fig. 4. Between-column and within-column whitespace rectangles.

be eliminated based on heuristics that they usually have small width or have no vertical neighbors.

For each CC chain, we estimate the number of between-column rectangles N based on the chain width and only retain the N widest rectangles, because between-column rectangles are wider than within-column ones. There is hardly a document that has more than eight columns. Therefore, N of a chain with a length of L is defined as

$$N = max(1, 8 * L/page\_width). \quad (1)$$

N is set at least one to handle situations when a chain has only one between-column rectangle and no within-column ones. As is shown in Fig. 6(c), rectangles of smaller width are removed. An over-segmentation of text lines are acquired by linking horizontally adjacent CCs which have no rectangles in between.

A rectangle can also be eliminated if it is surrounded by text lines and isolated from other rectangles. Obviously, such a rectangle cannot be between-column. Positive and negative examples are both shown in Fig. 5. Therefore, more rectangles

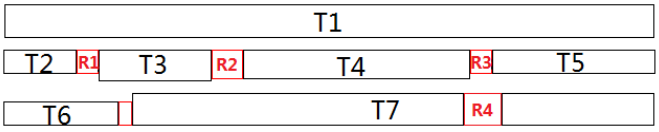


Fig. 5. Rectangle R1 is surrounded by text lines T1, T2, T3 and T6. R2 is surrounded by T1, T3, T4 and T7. Therefore, R1 and R2 are isolated rectangles. R3 and R4 are not isolated by text lines.

are removed as shown in Fig. 6(d). By doing this, the over-segmentation of text lines is lessened with the linking of more horizontally adjacent CCs.

Fig. 6. (a) CC chains; (b) Whitespace rectangles; (c) Retaining N widest rectangles in each chain; (d) Removal of isolated rectangles; (e) Over-segmentation of text lines.

C. Whitespace rectangles grouping

We analyze the remaining rectangles in groups, which are vertical rectangle chains formed by linking vertically adjacent rectangles (Fig. 8(a)).

Up to now most within-column whitespace rectangles have been removed and their adjacent CCs have been linked into text lines. Out of all the rectangles in a text line, the one with the largest width defines the maximum gap of the text line. It is highly probable that a rectangle causes over-segmentation if it is not wider than the maximum gap of its horizontally adjacent text lines. Such a rectangle is labelled as a within-column candidate. Examples are shown in Fig. 7. Vertical rectangle chains containing only within-column candidates are eliminated (Fig. 8(b)). Following the removal of such rectangles, their adjacent text lines are merged and maximum gaps are updated.

A within-column rectangle will not be labelled as a candi-

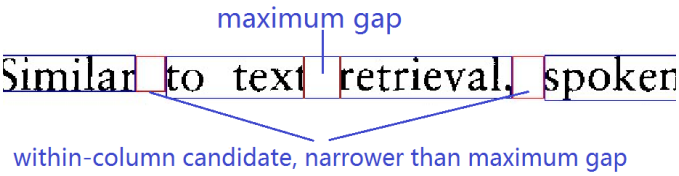


Fig. 7. Examples of a maximum gap and within-column candidates.

date in the above process if its adjacent text lines on left and right sides both have only one word. Obviously, the maximum gaps of both text lines are narrower than the within-column rectangle. However, text lines of one word were all acquired at the step of removing rectangles of smaller width in the *whitespace rectangles filtering* section and since then they never change. Therefore, we also label a rectangle as a within-column candidate if it has two one-word adjacent text lines which were acquired at that step. Along with the within-column candidates labelled by comparing with maximum gaps, we repeat the process of removing within-column candidate chains once again. Although a between-column rectangle may also have two adjacent text lines of one word, it is very unlikely that a between-column rectangle chain consists of all this kind of between-column rectangles.

After removing the vertical rectangle chains of all within-column candidates, in each of the remaining chains, we check the top and bottom rectangle to see whether they are within-column candidates or not. A within-column candidate at the top or bottom of a chain is probably a real within-column one adjacent to a real column separator. Hence, such candidate rectangles are also removed. With more adjacent CCs being linked after removing within-column rectangles, we get text lines which barely have over-segmentation (Fig. 8(c)).

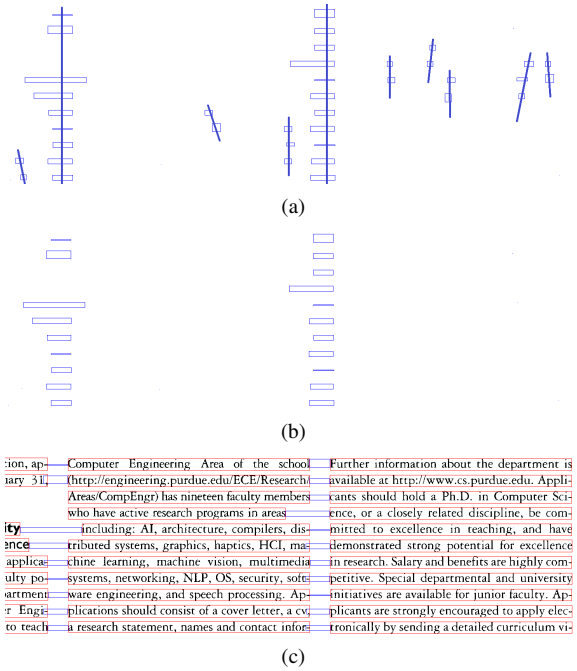


Fig. 8. (a) Whitespace rectangle chains; (b) Removal of within-column rectangle candidate chains; (c) Text lines after removing within-column rectangle chains.

D. Processing of large connected components

Large connected components (CCs) postponed to this stage are classified into text CCs and non-text CCs. The large CCs which contain text lines are treated as graphics and are eliminated. Large CCs forming long horizontal or vertical straight lines are also eliminated.

The remaining large CCs are treated as text CCs to be merged with detected text lines or form new text lines. A large CC will be merged into its horizontally adjacent text line if the gap between them is not wider than the maximum gap of the text line, and the height of the CC is not greater than twice the text line height. The remaining large CCs are linked to form new text lines in a way similar to linking normal CCs except that the maximum number of between-column rectangles is reduced to four. During the linking, if the enclosing box of two large CCs contain a detected text line, they are treated as graphics and are removed. Large text lines with no more than two CCs are also removed.

E. Text blocks identification

Text blocks are identified by linking vertically adjacent text lines considering that the text lines in a block are vertically aligned (horizontally overlapping). To do this, we find the below nearest neighbors (BNNs) for each text line. A trick similar to the one used to find RNNs of CCs is also used here. First we sort the text lines in ascending order of their y_min. Then for each text line T, we compare it with the subsequent text lines. When T first encounters a text line A and A overlaps horizontally with T, then A is a BNN of T. Subsequent text lines need to overlap vertically with A to be a BNN of T. Therefore the process of finding BNNs of T can be terminated when a subsequent text line S does not vertically overlap with A. This is illustrated in Fig. 9, where a rectangle denotes a text line.

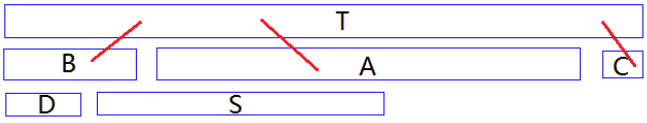


Fig. 9. The order of the text lines is T, A, B, C, S, D after sorting them according to y_min. T finds A first as a BNN. B and C overlap vertically with A, and so, are the BNNs of T. D and S do not overlap vertically with A, and so, cannot be BNNs of T.

To ensure there is only one text line at a certain vertical position of a text block, text lines are linked into chains sequentially. After the closure of the previous chain, the first text line of the remaining ones is used as the start of a new chain, which is enlarged by adding BNN of the previous member until it adds a text line which has no BNN or has more than one BNN. If a text line to be added is the BNN of more than one text line, the enlargement of the chain is terminated before the addition. New chains are generated continually until there are no text lines left. The members of a text line chain are grouped to form a text block, as exemplified in Fig. 10.

At last, the reading order of text blocks is determined using the method in [11] which considers the geometric relationship between blocks to determine the partial order and finally extends to global order.

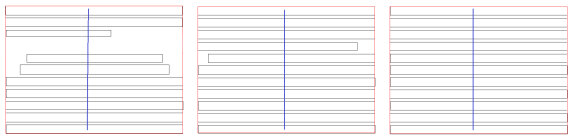


Fig. 10. Text line chains and text blocks.

III. Experimental Results

The evaluation of page segmentation performance is difficult because the correspondence between detected text blocks and ground-truth is complicated. The ICDAR page segmentation competitions [9] [12] provide an evaluation methodology which has been used successfully to compare the entrants of the competitions. The methodology combines different types of segmentation errors (split, merge, miss, false detection, misclassification and reading order) using adjustable weights accounting for different application scenarios. The details can be found in [12] and [13]. Using the methodology, we evaluated the performance of our method on the test set of ICDAR2009 competition and compare with the published results of the competition. The test set contains 55 document images of various layouts. The results for different scenarios are shown in Fig. 11, Fig. 12 and Fig. 13, respectively.

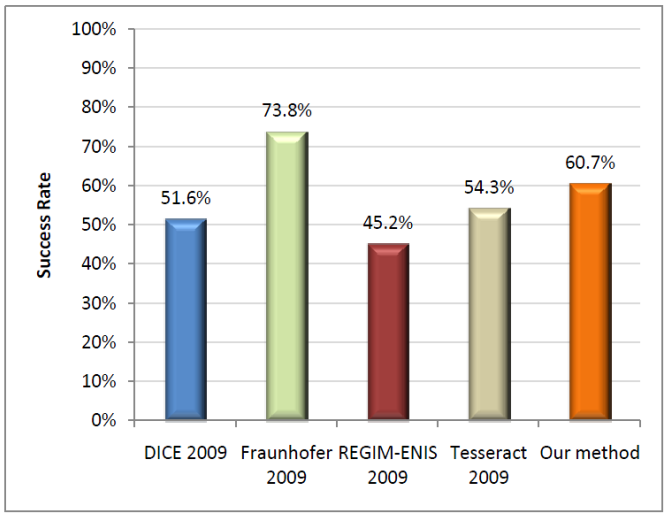


Fig. 11. Results using the segmentation-scenario evaluation profile.

The evaluation mechanism gives different comparison results depending on the evaluation scenarios related to real world applications. In some applications, all types of regions are of interest. While some other applications only concern text regions. The segmentation scenario takes segmentation success rate for all types of regions into account. The OCR scenario considers segmentation success rate of all types of regions and labelling success rate of text regions. Our algorithm is inferior to the best system of the competition when evaluated in segmentation and OCR scenarios which consider all types of regions, because we focus only on text regions and the other regions are removed. The Fraunhofer system outperforms our method in the first two scenarios largely because 29 out of 55 documents in the test set contain large graphics regions. The text scenario favors our method because it only considers

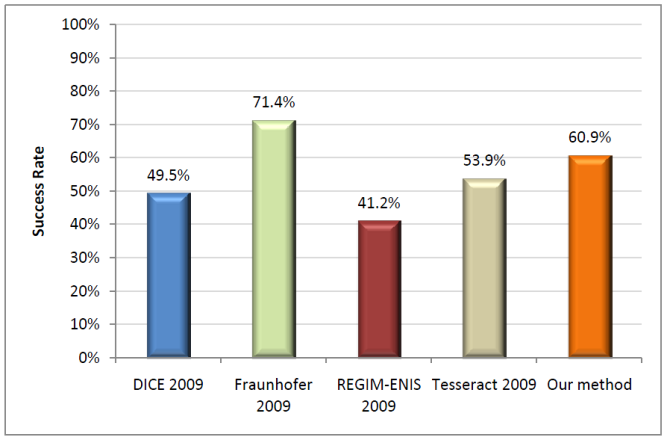


Fig. 12. Results using the OCR-scenario evaluation profile.

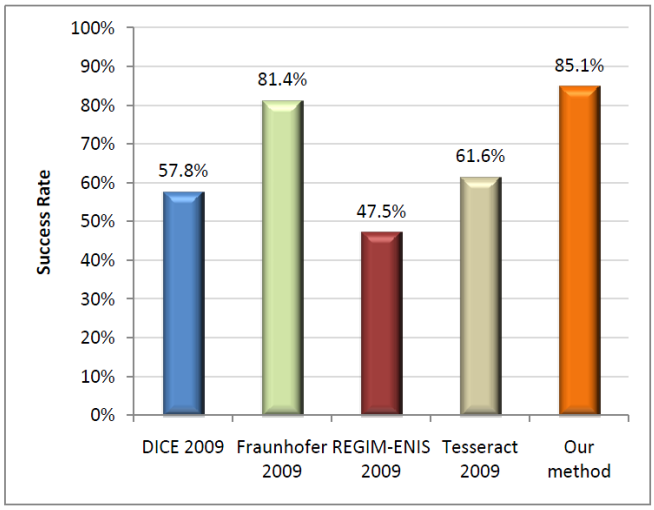


Fig. 13. Results using the text-scenario evaluation profile.

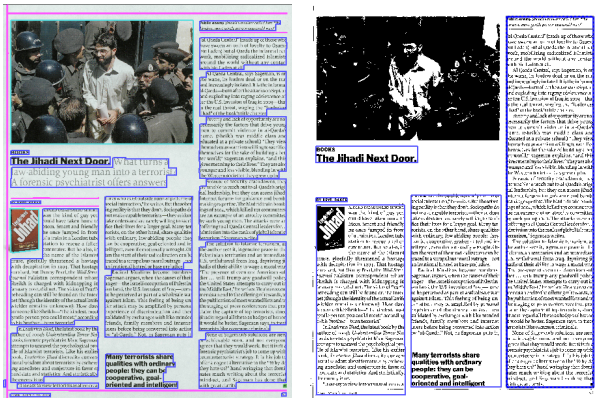
segmentation and labelling success rate of text regions. In this scenario, our method gives the best performance. Fig. 14 shows the segmentation result of our method compared with the ground-truth, where our method detects text regions correctly but ignores graphics and does not split text regions into paragraphs.

IV. Conclusion

We proposed an efficient hybrid page segmentation method based on whitespace analysis. Based on connected component analysis, whitespace rectangles are extracted and filtered progressively incorporating both foreground and background information. Experimental results on the ICDAR2009 page segmentation competition dataset demonstrate the superiority of our method in respect of text recognition. Future improvements are to reduce artificial parameters and refine the splitting of text regions into paragraphs.

Acknowledgment

The author would like to thank Apostolos Antonocopoulos for providing document image data and the evaluation results of our method. This work is supported in part by the National Basic Research Program of China (973 Program) Grant 2012CB316302, the National Natural Science Foundation of China (NSFC) grants 61175021 and 61273269.



(a) (b)

Fig. 14. Document image ground truth (a) and segmentation result (b).

References

[1] L. O'Gorman, The document spectrum for page layout analysis, *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, no. 11, pp. 1162-1173, 1993.

[2] A. Simon, J. Pret, A. Johnson, A fast algorithm for bottom-up document layout analysis, *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 3, pp. 273-276, 1997.

[3] A. Jain, B. Yu, Document representation and its application to page decomposition, *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 3, pp. 294-308, 1998.

[4] G. Nagy, S. Seth, Hierarchical representation of optically scanned documents, *Proc. 7th ICPR,* pp. 347-349, 1984.

[5] T.M. Breuel, Two geometric algorithms for layout analysis, *Document Analysis Systems,* pp. 188-199, Aug 2002.

[6] K. Kise, O. Yanagida, S. Takamatsu, Page segmentation based on thinning of background, *Proc. 13th ICPR,* pp. 788-792, 1996.

[7] K. Kise, A. Sato, M. Iwata, Segmentation of page images using the area voronoi diagram, *Computer Vision and Image Understanding,* vol. 70, no. 3, pp. 370-382, June 1998.

[8] T. Pavlidis, J. Zhou, Page segmentation and classification, *CVGIP: Graphical Models and Image Processing,* vol. 54, pp. 484-496, 1992.

[9] A. Antonacopoulos, C. Clausner, C. Papadopoulos, S. Pletschacher, Historical document layout analysis competition, *Proc. 11th ICDAR,* pp. 1516-1520, 2011.

[10] F. Chang, C.-J. Chen, C.-J. Lu, A linear-time component-labeling algorithm using contour tracing technique, *Computer Vision and Image Understanding,* vol. 93, pp. 206-220, 2004.

[11] T.M. Breuel, High performance document layout analysis, *Proc. Symposium on Document Image Understanding Technology,* April 2003.

[12] A. Antonacopoulos, S. Pletschacher, D. Bridson, C. Papadopoulos, ICDAR2009 page segmentation competition, *Proc. 10th ICDAR,* pp. 1370-1374, 2009.

[13] C. Clausner, S. Pletschacher A. Antonacopoulos, Scenario driven in-depth performance evaluation of document layout analysis methods, *Proc. 11th ICDAR,* pp. 1516-1520, 2011.