Performance Comparison of Six Algorithms for Page Segmentation

Faisal Shafait, Daniel Keysers, and Thomas M. Breuel

Image Understanding and Pattern Recognition (IUPR) research group,
German Research Center for Artificial Intelligence (DFKI)
and Technical University of Kaiserslautern,
D-67663 Kaiserslautern, Germany
{faisal, keysers, tmb}@iupr.net

Abstract. This paper presents a quantitative comparison of six algorithms for page segmentation: X-Y cut, smearing, whitespace analysis, constrained text-line finding, Docstrum, and Voronoi-diagram-based. The evaluation is performed using a subset of the UW-III collection commonly used for evaluation, with a separate training set for parameter optimization. We compare the results using both default parameters and optimized parameters. In the course of the evaluation, the strengths and weaknesses of each algorithm are analyzed, and it is shown that no single algorithm outperforms all other algorithms. However, we observe that the three best-performing algorithms are those based on constrained text-line finding, Docstrum, and the Voronoi-diagram.

1 Introduction

Document image layout analysis is a crucial step in many applications related to document images, like text extraction using optical character recognition (OCR), reflowing documents, and layout-based document retrieval. Layout analysis is the process of identifying layout structures by analyzing page images. Layout structures can be physical (text, graphics, pictures, ...) or logical (titles, paragraphs, captions, headings, ...). The identification of physical layout structures is called physical or geometric layout analysis, while assigning different logical roles to the detected regions is termed as logical layout analysis [1]. In this paper we are concerned with geometric layout analysis. The task of a geometric layout analysis system is to segment the document image into homogeneous zones, each consisting of only one physical layout structure, and to identify their spatial relationship (e.g. reading order). Therefore, the performance of layout analysis methods depends heavily on the page segmentation algorithm used. Over the last two decades, several page segmentation algorithms have been proposed in the literature (for a literature survey, please refer to [1, 2]).

The problem of automatic evaluation of page segmentation algorithms is increasingly becoming an important issue. Major problems arise due to the lack of a common dataset, a wide diversity of objectives, a lack of meaningful quantitative evaluation, and inconsistencies in the use of document models. This makes

H. Bunke and A.L. Spitz (Eds.): DAS 2006, LNCS 3872, pp. 368–379, 2006.

the comparison of different page segmentation algorithms a difficult task. Meaningful and quantitative evaluation of page segmentation algorithms has received attention in the past. Yanikoglu et al. [3] presented a region-based page segmentation benchmarking environment, named Pink Panther. Liang et al. [4] proposed a performance metric for document structure extraction algorithms by finding the correspondences between detected entities and ground-truth. The quality of page segmentation algorithms was also evaluated by analyzing the errors in the recognized text [5]. However, text-based approaches have found little use since they measure the output of multiple steps and cannot be used to evaluate page segmentation alone. Das et al. [6] suggested an empirical measure of performance of a segmentation algorithm based on a graph-like model of the document.

There has been little effort in the past to compare different algorithms on a quantitative basis. Mao et al. [7] presented an empirical performance evaluation methodology and compared three research algorithms and two commercial products. Their evaluation methodology is based on text-line detection accuracy, so it is particularly useful for evaluating text segmentation approaches. Recent page segmentation competitions [8, 9] address the need of comparative performance evaluation under realistic circumstances. However, a limitation of the competition-based approach is that competing methods only participate if they are implemented and used by a participant. It means several well-known algorithms might not be a part of the comparison at all.

This paper focuses on comparative performance evaluation of six representative algorithms for page segmentation. It is an extension of the work by Mao et al. [7], and adds three more algorithms to the comparison. The algorithms compared in [7] are X-Y cut [10], Docstrum [11], and the Voronoi-diagram based approach [12]. The algorithms added to the comparison in this work are the smearing algorithm [13], whitespace analysis [14], and the constrained text-line finding algorithm [15]. A brief description of the six algorithms used in the comparison will be given in Section 2, followed by the error metric definition in Section 3. Section 4 describes the experiments performed and results obtained, followed by discussion of the results in Section 5 and conclusion in Section 6.

2 Algorithms for Page Segmentation

We selected six representative algorithms for page segmentation. Furthermore, we have introduced a dummy algorithm to determine a bottom line of the possible performance. A brief description of each algorithm and its parameters are described in turn in the following.

2.1 Dummy Algorithm

The dummy segmentation algorithm takes the whole page as one segment. The purpose of this algorithm is to see how well we can perform without doing anything. Then the performance of other algorithms can be seen as gains over that achieved by the dummy algorithm. Using the dummy algorithm also highlights limitations of the evaluation scheme as detailed in Section 3.

2.2 X-Y Cut

The X-Y cut segmentation algorithm [10], also referred to as recursive X-Y cuts (RXYC) algorithm, is a tree-based top-down algorithm.

The root of the tree represents the entire document page. All the leaf nodes together represent the final segmentation. The RXYC algorithm recursively splits the document into two or more smaller rectangular blocks which represent the nodes of the tree. At each step of the recursion, the horizontal and vertical projection profiles of each node are computed. Then, the valleys along the horizontal and vertical directions, V_X and V_Y, are compared to corresponding predefined thresholds T_X and T_Y. If the valley is larger than the threshold, the node is split at the mid-point of the wider of V_X and V_Y into two children nodes. The process continues until no leaf node can be split further. Then, noise regions are removed using noise removal thresholds T_X^n and T_Y^n.

2.3 Smearing

The run-length smearing algorithm (RLSA) [13] works on binary images where white pixels are represented by 0's and black pixels by 1's. The algorithm transforms a binary sequence x into y according to the following rules:

1. 0's in x are changed to 1's in y if the number of adjacent 0's is less than or equal to a predefined threshold C.
2. 1's in x are unchanged in y.

These steps have the effect of linking together neighboring black areas that are separated by less than C pixels. The RLSA is applied row-wise to the document using a threshold C_h, and column-wise using threshold C_v, yielding two distinct bitmaps. These two bitmaps are combined in a logical AND operation. Additional horizontal smearing is done using a smaller threshold, C_s, to obtain the final bitmap. Then, connected component analysis is performed on this bitmap, and using threshold C_{21} and C_{22} on the mean run of black pixel in a connected component and block height, connected components are classified into text and non-text zones.

2.4 Whitespace Analysis

The whitespace analysis algorithm described by Baird [14] analyzes the structure of the white background in document images. The first step is to find a set of maximal white rectangles (called *covers*) whose union completely covers the background. Breuel's algorithm for finding the maximal empty whitespace [15] is used in our implementation for this step. These covers are then sorted with respect to the sort key, $K(c)$:

$$K(c) = \sqrt{\text{area}(c) * W(|log_2\,(\text{height}(c)/\text{width}(c))|)} \tag{1}$$

where c is the cover and $W(.)$ is a dimensionless weighting function. Baird [14] chose a special weighting function using experiments on a particular dataset. We used an approximation of the original weighting function as

$$W(x) = \begin{cases} 0.5 & \text{if } x < 3 \\ 1.5 & \text{if } 3 \leq x < 5 \\ 1 & \text{otherwise} \end{cases} \quad (2)$$

The purpose of the weighting function is to assign higher weight to tall and long rectangles because they are supposed to be meaningful separators of text blocks.

In the second step, the rectangular covers $c_i, i = 1, \ldots, m$, where m is the total number of whitespace covers, are combined one by one to generate a corresponding sequence $s_j, j = 1, \ldots, m$ of segmentations. A segmentation is the uncovered area left by the union of the covers combined so far. Before a cover c_i is unified to the segmentation s_j, a trimming rule is applied to avoid early segmentation of narrow blocks. The unification of covers continues until the stopping rule (3) is satisfied:

$$K(s_j) - W_s * F(s_j) \leq T_s \quad (3)$$

where $K(s_j)$ is the sort key $K(c_j)$ of the last cover unified in making segmentation s_j, $F(s_j) = j/m$, W_s is a weight, and T_s is stopping threshold. At the final segmentation, the uncovered regions represent the union of interiors of all black input rectangles. We take bounding boxes of the uncovered regions as representative of the text segments.

2.5 Constrained Text-Line Detection

The layout analysis approach by Breuel [15] finds text-lines as a two step process:

1. Find tall whitespace rectangles and evaluate them as candidates for gutters, column separators, etc. The algorithm for finding maximal empty whitespace is described in [15]. The whitespace rectangles are returned in order of decreasing quality and are allowed a maximum overlap of O_m.
2. The whitespace rectangles representing the columns are used as obstacles in a robust least square, globally optimal text-line detection algorithm [16]. Then, the bounding box of all the characters making the text-line is computed.

The method was merely intended by its author as a demonstration of the application of two geometric algorithms, and not as a complete layout analysis system; nevertheless, we included it in the comparison because it has already proven useful in some applications. It is also nearly parameter free and resolution independent.

2.6 Docstrum

The Docstrum algorithm by Gorman [11] is a bottom-up approach based on nearest-neighborhood clustering of connected components extracted from the document image. After noise removal, the connected components are separated into two groups, one with dominant characters and another one with characters in titles and section heading, using a character size ratio factor f_d. Then, K nearest neighbors are found for each connected component. Then, text-lines are found by computing the transitive closure on within-line nearest neighbor pairings using a threshold f_t. Finally, text-lines are merged to form text blocks using a parallel distance threshold f_{pa} and a perpendicular distance threshold f_{pe}.

2.7 Voronoi-Diagram Based Algorithm

The Voronoi-diagram based segmentation algorithm by Kise et al. [12] 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 sr. Then, noise removal is done using a maximum noise zone size threshold nm, 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. Superfluous Voronoi edges are deleted using a criterion involving the area ratio threshold ta, and the inter-line spacing margin control factor fr. Since we evaluate all algorithms on document pages with Manhattan layouts, a modified version of the algorithm [7] is used to generate rectangular zones.

3 Error Metrics

We use text-line detection accuracy [7] as the error metric in our evaluation. The text-lines are represented as bounding boxes enclosing all the characters constituting the text-line. Three types of errors are defined.

1. Ground-truth text-lines that are missed (C), i.e. they are not part of any detected text region.
2. Ground-truth text-lines whose bounding boxes are split (S), i.e. the bounding box of a text-line does not lie completely within one detected segment.
3. Ground-truth text-lines that are horizontally merged (M), i.e. two horizontally overlapping ground-truth lines are part of one detected segment.

These error measures are defined based on set theory and mathematical morphology [7]. Let G be the set of all the ground-truth text-line, and $|G|$ denote the cardinality of the set G, then the overall performance is measured as the percentage of ground-truth text-lines that are not found correctly:

$$\rho = \frac{|C \cup S \cup M|}{|G|} \tag{4}$$

A ground-truth text-line is said to lie completely within one detected text segment if the area overlap between the two is significant. Significance is determined using four tolerance parameters which are identical here to those used in [7].

The main advantages of this approach are that it is independent of OCR recognition error, is independent of zone shape, and requires only text-line level ground-truth. Since a text block can be easily decomposed into text-lines by projecting each block parallel to its baseline and analyzing the resulting one-dimensional profile, the assignment of text-lines to text blocks is not critical. Therefore, the evaluation scheme does not take into account vertical merge errors. However, there is a drawback of this approach. If a segmentation algorithm just takes the whole page as one segment, the split and missed errors vanish

($C = \emptyset, S = \emptyset$). Typically for single-column documents, $M = \emptyset$. Hence, without doing anything, the segmentation accuracy can be high if there is a large proportion of single-column document images in the test dataset. This effect was not considered in the original evaluation [7]. In order to check the severity of the problem, we have introduced a dummy segmentation algorithm into the comparison, as discussed in Section 2.1. Furthermore, we report the performance of each algorithm separately for single-column, two-column, and three-column document images. This allows us to assess the strengths and weaknesses of different algorithms.

4 Experiments and Results

The evaluation of the page segmentation algorithms was done on the University of Washington III (UW-III) database [17]. The database consists of 1600 English document images with manually edited ground-truth of entity bounding boxes. These bounding boxes enclose text and non-text zones, text-lines and words. We used the 978 images that correspond to the UW-I dataset pages. Only the text regions are evaluated, and non-text regions are ignored. The dataset is divided into 100 training images and 878 test images. The purpose of the training images is to find suitable parameter values for the segmentation algorithms. The experiments are done using both default parameters as mentioned in the respective papers and tuned/optimized parameters (Table 1). This allows us to assess how much the performance of each algorithm depends on the choice of good parameters for the task. The parameters for the X-Y cut algorithm are highly application dependent, so no default parameters are specified in [10]. The optimized parameter values used for X-Y cut, Docstrum, and Voronoi-diagram based algorithms were the same as in [7]. For the smearing, whitespace, and constrained text-line finding algorithms, we experimented with different parameter values and selected those which gave lowest error rates on the training set.

Table 1. Parameter values used for each algorithm in the evaluation given in Table 2. For dummy, X-Y cut, smearing, and text-line finding algorithms, default and optimized parameters are the same.

Algorithm	Default values	Optimal values
Dummy	None	
X-Y cut	$T_X = 35, T_Y = 54, T_X^n = 78, T_Y^n = 32$	
Smearing	$C_h = 300, C_v = 500, C_s = 30, C_{21} = 3, C_{22} = 3$	
Text-line	$O_m = 0.8$	
Whitespace	$W_s = 42.43, T_s = 34.29$	$W_s = 42.43, T_s = 65$
Docstrum	$K = 5, f_t = 2.578, f_d = 9,$ $f_{pe} = 1.3, f_{pa} = 1.5$	$K = 8, f_t = 2.578, f_d = 9,$ $f_{pe} = 0.6, f_{pa} = 2.345$
Voronoi	$s_r = 6, nm = 11,$ $fr = 0.34, ta = 40$	$s_r = 6, nm = 11,$ $fr = 0.083, ta = 200$

Table 2. The evaluation results for different page segmentation algorithms on 100 train images and 878 test images. The results are reported in terms of percentage of text-lines detection errors (Eq. 4).

	Default parameters			Optimized parameters		
	Train	Test		Train	Test	
Algorithm	Mean	Mean	Stdev	Mean	Mean	Stdev
Dummy	52.2	48.8	39.0	52.2	48.8	39.0
X-Y cut	14.7	17.1	24.4	14.7	17.1	24.4
Smearing	13.4	14.2	23.0	13.4	14.2	23.0
Whitespace	12.7	12.2	20.0	9.1	9.8	18.3
Text-line	8.9	8.5	14.4	8.9	8.5	14.4
Docstrum	8.7	11.2	22.6	4.3	6.0	15.2
Voronoi	6.8	7.5	12.9	4.7	5.5	12.3

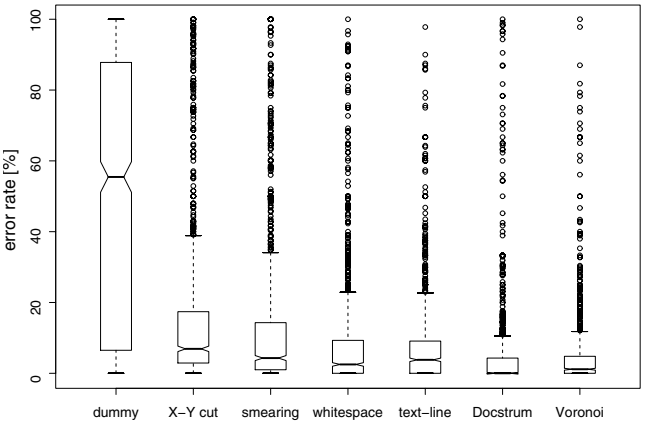
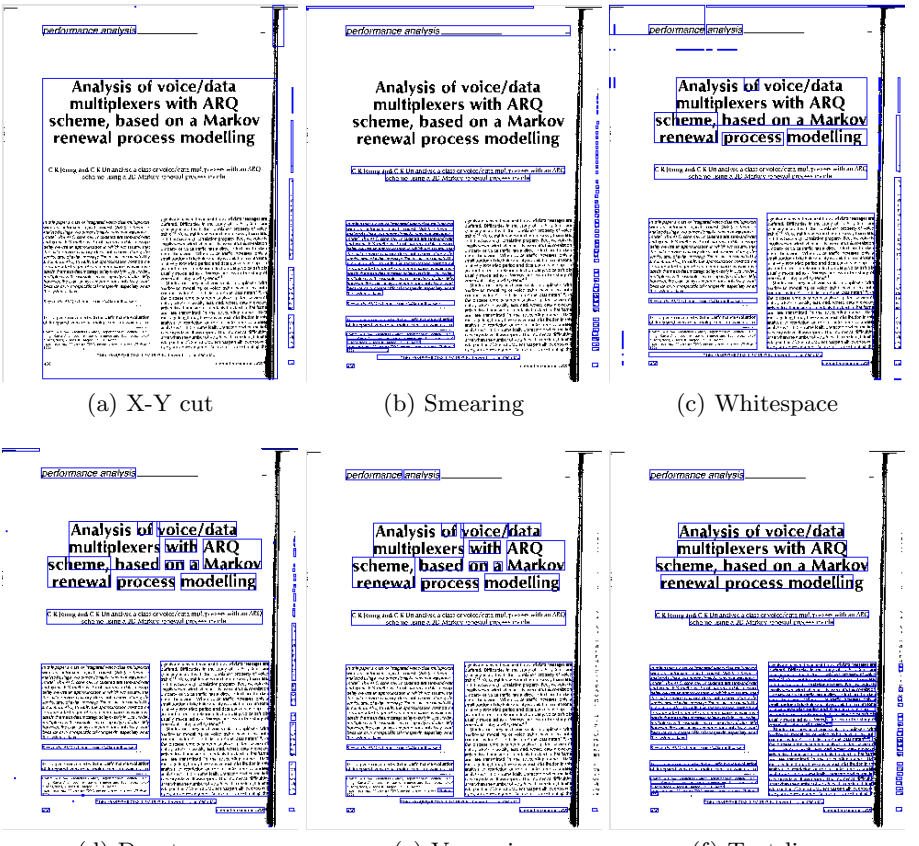


Fig. 1. Box plot for the results obtained with optimized parameters on the test data

We have used the page segmentation evaluation toolkit (PSET) [18] to accelerate the evaluation procedure. The PSET evaluation package implements the training and evaluation scheme by [7], and can be easily extended to evaluate new algorithms and experiment with new metrics and datasets. The average text-line detection error rate for each algorithm is given in Table 2. The high standard deviation in the error rate of each algorithm shows that the algorithms work very well on some images, while failing badly on some other images.

Fig. 1 shows a box plot of the error rates observed for each algorithm. The boxes in the box plot represent the interquartile range, i.e. they contain the middle 50% of the data. The lower and upper edges represent the first and third quartiles, whereas the middle line represents the median of the data. The notches represent the expected range of the median. The 'whiskers' on the two sides show



(a) X-Y cut (b) Smearing (c) Whitespace

(d) Docstrum (e) Voronoi (f) Text-line

Fig. 2. Segmentation results from applying each algorithm to one page image. The page contains a title in large font and a big noise strip along the right border. (a) The X-Y cut algorithm fails in the presence of noise and tends to take the whole page as one segment. (b) The smearing algorithm also classifies the detected regions as text/non-text, and thus misses the lines joined by the noise bar. (c),(d),(e) Due to the large font size and big inter-word spacing, the Voronoi, Docstrum, and whitespace algorithms split the title lines. (f) Due to the noise bar, several characters on the right side of each line in the second column were merged with the noise bar and the text-line finding algorithm did not include these characters.

inliers, i.e. points within 1.5 times the interquartile range. The outliers are represented by small circles outside the whiskers. We can observe the following details: A ranking of the algorithms based on their median error would deviate from the ranking based on the average error. Remarkably, the Docstrum algorithm does not make any errors for more than 50% of the pages in the test set, which is not achieved by any other algorithm. This might be a property that would be preferable in certain applications, while for other applications the average error rate may be more important.

5 Error Analysis and Discussion

The results of applying each algorithm to one test image (A005BIN.TIF) are shown in Fig. 2. The different types of errors made by each algorithm are shown in Table 3 and are discussed in the following. Many of these results are based on a visual inspection of the obtained results.

- The dummy algorithm results in a large number of merge errors as expected.
- The X-Y cut algorithm fails in the presence of noise and tends to take the whole page as one segment. This results in several merge errors for two column pages. For clean documents, the algorithm tends to split text-lines at the borders, resulting in some split errors on each page.
- The smearing algorithm classifies text-lines merged with noise blocks as non-text, resulting in a large number of missed errors. This classification step is necessary because otherwise for L-shaped noise blocks appearing due to photocopy effect several merge errors occur.
- The whitespace algorithm is sensitive to the stopping rule. Early stopping results in a higher number of merge errors, late stopping results in more split errors. This effect can be observed in Table 3. This may present problems for more diverse collections than UW-III.
- The major part of the errors made by the constrained text-line finding algorithm are split errors. The main source of these errors are characters merged with noise blocks, because they are rejected beforehand as noise by the connected component analysis algorithm. Note that in contrast to the prior evaluation of the algorithm, these errors occur here because the evaluation is based on bounding boxes; if the text-lines are continued to the left and right side, these errors would vanish. Furthermore, these errors are not relevant to better scans. Some split errors also occur due to page curl at the border, which results in two separate baselines fit to a single text-line. Merge errors appear when page numbers are in close proximity to journal or article names in footers or headers. Single digit page numbers are missed by the text-line finding algorithm, because it requires at least two connected components to form a line.
- In the Voronoi and Docstrum algorithms, the inter-character and inter-line spacings are estimated from the document image. Hence spacing variations due to different font sizes and styles within one page result in split errors in both algorithms. However, since both algorithms find text regions, and not individual text-lines, huge noise bars along the border do not result in split errors, as the bounding box of the text region includes the characters merged in the noise bar if at least one text line has no merged characters.

Table 4 shows the error rates of the algorithms separated for different document characteristics. First, the documents were separated according to the 'maximum columns number' attribute recorded for each page. There are 362, 449, and 67 one-, two-, and three-column documents in the test set of 878 pages. We can observe that the smearing, whitespace, and text-line algorithms perform

Table 3. Percentage of different types of errors made by each algorithm

	Default parameters			Optimized parameters		
Algorithm	Split	Merge	Missed	Split	Merge	Missed
Dummy	0.0	65.5	0.0	0.0	65.5	0.0
X-Y cut	5.6	7.8	0.4	5.6	7.8	0.4
Smearing	3.8	1.0	5.7	3.8	1.0	5.7
Whitespace	6.6	1.3	0.0	5.0	2.6	0.0
Text-line	5.1	1.3	0.2	5.1	1.3	0.2
Docstrum	4.5	9.0	0.0	2.5	3.6	0.01
Voronoi	4.9	0.8	0.02	2.9	1.3	0.02

Table 4. Text-line detection errors [%] for each of the algorithms separated for one-, two-, and three-column documents, and separated for photocopies or direct scans

	No. of columns			Photocopy	
Algorithm	1	2	3	No	Yes
Dummy	8.3	75.6	88.5	68.7	46.2
X-Y cut	19.9	15.6	11.7	14.7	17.4
Smearing	23.5	7.9	5.8	6.6	15.1
Whitespace	14.5	6.7	5.6	2.9	10.8
Text-line	13.3	5.3	4.4	3.6	9.2
Docstrum	5.8	6.2	5.2	6.2	5.9
Voronoi	6.9	4.6	3.4	2.8	5.8

much worse on one-column documents than on the average. This behavior can be explained by the stronger effect of the noise blocks occurring in photocopied images for these one-column documents, because each line is affected. We further investigated this hypothesis by separating the documents according to their 'degradation type' attribute. There are 776 photocopied and 102 directly scanned documents in the test set and the respective results are shown in Table 4. We can observe that the algorithms performing worse on one-column documents in fact also perform worse on the photocopied images due to the noise blocks. Interestingly, especially the Docstrum algorithm does not gain accuracy for clean documents, while the Voronoi-based algorithm still performs best. The smearing, whitespace and text-line algorithms are most effected by the photocopy effects. This suggests that they would perform better for current layout analysis tasks in which most documents are directly scanned.

The timing of the algorithms cannot be directly compared because of the difference in their output level. Whitespace, Docstrum, Voronoi, and X-Y cut algorithms give text blocks which have still to be separated into text-lines. Whereas the constrained text-line finding algorithm directly gives the text-lines as output. Secondly, the smearing algorithm also includes a block-classification step,

which is missing in other algorithms. Furthermore, the Docstrum, whitespace, and constrained text-line finding algorithms depend on the computation of connected components in the image, which were calculated offline and stored in the database. However, an informal ranking of the running times from fastest to slowest is: X-Y cut, Voronoi, Docstrum, whitespace, smearing, and constrained text-line finding. All algorithms completed page processing in an average of less than 7 seconds on a 2GHz AMD PC running Linux.

6 Conclusion

In this paper, we compared the performance of six different page segmentation algorithms on the UW-III dataset using a text-line detection accuracy based performance metric. The evaluation results showed that generally the X-Y cut, smearing, and whitespace algorithms perform poorer than the Docstrum, Voronoi, and constrained text-line finding algorithms. The high variance in the detection error rate prohibits the marking of any algorithm as a clear winner.

We also inspected the results visually and generally observed the following. The constrained text-line finding algorithm is more robust to within page variations in font size, style, inter-character and inter-line spacing than any other algorithm. However, due to page curl and photocopy effect, the characters merged with a noise block are ignored by the text-line finding algorithm resulting in more split errors. Note, though, that the text-line method is not a full-fledged layout analysis method yet. On the other hand, page numbers are more reliably found by Voronoi and Docstrum algorithms. Since each algorithm has different strengths and weaknesses, combining the results of more than one algorithm may yield promising results.

The results also reveal several limitation of the UW-III database. First, all images have the same resolution. This fact favors algorithms with many parameters that can be tuned to the task and contributes significantly to the good results of algorithms based on thresholds on layout distances. Furthermore, the presence of many photocopied pages that are not well binarized is not representative of the majority of scanned documents that are captured today. In summary, although the diversity of layouts found in the UW-III database is good, we consider the variability of physical properties like resolution and image degradation to be too limited for a good evaluation; and we therefore need to test the robustness of algorithms using more heterogeneous document collections. It would be an important step to make available a larger, ground-truthed database with more variability for future research.

Acknowledgments

This work was partially funded by the BMBF (German Federal Ministry of Education and Research), project IPeT (01 IW D03). The authors would also like to acknowledge Joost van Beusekom's work in implementing the run-length smearing algorithm.

References

1. Cattoni, R., Coianiz, T., Messelodi, S., Modena, C.M.: Geometric layout analysis techniques for document image understanding: a review. Technical report, IRST, Trento, Italy (1998)
2. Mao, S., Rosenfeld, A., Kanungo, T.: Document structure analysis algorithms: a literature survey. Proc. SPIE Electronic Imaging **5010** (2003) 197–207
3. Yanikoglu, B.A., Vincent, L.: Ground-truthing and benchmarking document page segmentation. In: Proc. ICDAR, Montreal, Canada (1995) 601–604
4. Liang, J., Phillips, I.T., Haralick, R.M.: Performance evaluation of document structure extraction algorithms. CVIU **84** (2001) 144–159
5. Kanai, J., Nartker, T.A., Rice, S.V., Nagy, G.: Performance metrics for document understanding systems. In: Proc. ICDAR, Tsukuba, Japan (1993) 424–427
6. Das, A.K., Saha, S.K., Chanda, B.: An empirical measure of the performance of a document image segmentation algorithm. IJDAR **4** (2002) 183–190
7. Mao, S., Kanungo, T.: Empirical performance evaluation methodology and its application to page segmentation algorithms. IEEE TPAMI**23** (2001) 242–256
8. Antonacopoulos, A., Gatos, B., Bridson, D.: ICDAR 2005 page segmentation competition. In: Proc. ICDAR, Seoul, Korea (2005) 75–80
9. Antonacopoulos, A., Gatos, B., Karatzas, D.: ICDAR 2003 page segmentation competition. In: Proc. ICDAR, Edinburgh, UK (2003) 688–692
10. Nagy, G., Seth, S., Viswanathan, M.: A prototype document image analysis system for technical journals. Computer **7** (1992) 10–22
11. O Gorman, L.: The document spectrum for page layout analysis. IEEE TPAMI **15** (1993) 1162–1173
12. Kise, K., Sato, A., Iwata, M.: Segmentation of page images using the area Voronoi diagram. CVIU **70** (1998) 370–382
13. Wong, K.Y., Casey, R.G., Wahl, F.M.: Document analysis system. IBM Journal of Research and Development **26** (1982) 647–656
14. Baird, H.S.: Background structure in document images. In: Document Image Analysis, World Scientific, (1994) 17–34
15. Breuel, T.M.: Two geometric algorithms for layout analysis. In: Document Analysis Systems, Princeton, NJ. (2002)
16. Breuel, T.M.: Robust least square baseline finding using a branch and bound algorithm. In: Doc. Recognition & Retrieval, SPIE, San Jose, CA. (2002) 20–27
17. Guyon, I., Haralick, R.M., Hull, J.J., Phillips, I.T.: Data sets for OCR and document image understanding research. In: Handbook of character recognition and document image analysis, World Scientific, (1997) 779–799
18. Mao, S., Kanungo, T.: Software architecture of PSET: a page segmentation evaluation toolkit. IJDAR **4** (2002) 205–217