Performance Evaluation and Benchmarking of Six Page Segmentation Algorithms

Faisal Shafait, Daniel Keysers, and Thomas M. Breuel

Abstract— **Informative benchmarks are crucial for optimizing the page segmentation step of an OCR system, frequently the performance limiting step for overall OCR system performance. We show that current evaluation scores are insufficient for diagnosing specific errors in page segmentation and fail to identify some classes of serious segmentation errors altogether. This paper introduces a vectorial score that is sensitive to, and identifies, the most important classes of segmentation errors (over-, under-, and miss-segmentation) and what page components (lines, blocks, etc.) are affected. Unlike previous schemes, our evaluation method has a canonical representation of ground truth data and guarantees pixel-accurate evaluation results for arbitrary region shapes. We present the results of evaluating widely used segmentation algorithms (x-y cut, smearing, whitespace analysis, constrained text-line finding, docstrum, and Voronoi) on the UW-III database and demonstrate that the new evaluation scheme permits the identification of several specific flaws in individual segmentation methods.**

Index Terms— **Document page segmentation, OCR, performance evaluation, performance metric**

I. Introduction

The task of page segmentation is to divide the document image into homogeneous zones, each consisting of only one physical layout structure (text, graphics, pictures, . . .). Therefore, the performance of optical character recognition (OCR) systems depends heavily on the page segmentation algorithm used. Over the last three decades, several page segmentation algorithms have been proposed in the literature (for a literature survey, please refer to [1]–[3]). In this paper, we present three main contributions to the state-of-the-art in page segmentation:

1) Performance evaluation and comparison of six well-known algorithms for page segmentation using a state-of-the-art evaluation methodology [4]. We identify a severe flaw in the evaluation scheme when used for single column documents.
2) A novel, portable, and pixel-accurate representation for arbitrarily shaped page segments.
3) Several performance measures to identify and analyze different classes of segmentation errors made by a page segmentation algorithm.

The rest of the introduction section presents an overview of the state-of-the-art in the above-mentioned areas, and describes how our work augments the state-of-the-art.

The problem of automatic evaluation of page segmentation algorithms is increasingly becoming an important issue [5], [6]. 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 the benchmarking of different page segmentation algorithms a difficult task. Recent page segmentation competitions [7], [8] 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.

The quantitative evaluation of page segmentation algorithms has received some attention in the past. An approach for measuring the quality of page segmentation algorithms by analyzing the errors in the text recognized by OCR was first proposed in [9]. 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. Yanikoglu et al. [10] presented a region-based page segmentation benchmarking environment, named Pink Panther. Their approach is based on representing regions as arbitrary polygons, and hence becomes quite complex and cumbersome to use. Liang et al. [11] proposed a performance metric for document structure extraction algorithms by finding the correspondences between detected entities and ground-truth. Das et al. [12] suggested an empirical measure of performance of a segmentation algorithm based on a graph-like model of the document. However, their performance measure does not support evaluation of non-Manhattan page layouts. Similar approaches have been presented for range image segmentation in [13], and for image segmentation in general [14]. Mao et al. [4] presented an empirical benchmarking methodology based on text-line measure of page segmentation accuracy. This measure is particularly useful because it does not make assumptions about the layout of the document. Besides, it requires only text-line level ground-truth. They have compared three research algorithms and two commercial products using this method. We extend the work by Mao et al. [4], and add three more algorithms to the comparison (Section II). The algorithms compared in [4] are x-y cut [15], docstrum [16], and the Voronoi-diagram based approach [17]. The algorithms added to the comparison in this work are the smearing algorithm [18], whitespace analysis [19], and the constrained text-line finding algorithm [20]. In addition, we also identify a limitation of the evaluation scheme when it is used for single-column documents and use a dummy segmentation algorithm to highlight the problem in such cases. An analysis of the errors coincides with our finding. We overcome this limitation by presenting a vectorial score that helps in judging the kind and extent of segmentation errors made by the analyzed algorithm (Section II-C). This score is particularly useful in analyzing the behavior of a page segmentation algorithm for a given set of parameters. Our performance measures are based on a new way of representing layout information by embedding it in the color channels of a document image (Section II-A). Such a

F. Shafait and D. Keysers are with the Image Understanding and Pattern Recognition (IUPR) research group at the German Research Center for Artificial Intelligence (DFKI GmbH), Kaiserslautern, Germany. E-mail: {faisal.shafait, daniel.keysers}@dfki.de

T.M. Breuel is with the Computer Science Department at the Technical University of Kaiserslautern, Germany. E-mail: tmb@informatik.uni-kl.de

representation allows convenient interchange of ground truth and segmentation results in terms of standard image formats.

We use the University of Washington III (UW-III) dataset [21] for performance evaluation and benchmarking of the analyzed algorithms. The UW-III dataset has a large number of documents with different degradation types and is one of the standard dataset for evaluating different document analysis tasks. The main strength of the dataset is that for each document, along with the ground-truth information for words, text-lines, zones, and ASCII text, a number of document and zone attributes are available. This makes the database suitable for quantitatively evaluating a wide variety of tasks related to document image analysis. Researchers have used the UW-III dataset for evaluating their approaches related to different document analysis tasks like page segmentation [4], [11], block classification [22], layout analysis [23], table zone extraction [24], and document image classification [25]. We discuss the experiments performed and results obtained in Section IV, followed by the conclusion in Section V. We have reported parts of the work presented in this paper in [26] and [27]. In this paper we extend the performance measures presented in [27] by introducing the notion of correct segmentation. Additionally, we explain our methods in more detail than in [26] and [27], illustrating with visual examples where necessary. We have also included a correlation analysis of the errors made by each algorithm, which gives us interesting insights into the similarities of the behavior of different algorithms.

II. Performance evaluation of page segmentation algorithms

The performance evaluation measure proposed in [4] is based on set theory. This measure is based on the assumption that a text block can be easily segmented into text lines using horizontal projection. Let G be the set of all the ground-truth text-line in a document image, and $|G|$ denote the cardinality of the set G. Then, three subsets of text-lines are defined as follows:

1) The set of ground-truth text-lines that are missed (C), i.e. they are not part of any detected text region.
2) The set of 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) The set of ground-truth text-lines that are horizontally merged (M), i.e. two horizontally overlapping ground-truth lines are part of one detected segment.

The overall error rate is measured as the percentage of ground-truth text-lines that are not identified correctly:

$$\rho = \frac{|C \cup S \cup M|}{|G|} \tag{1}$$

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 two length thresholds in number of pixels. The thresholds control the tolerance level along the horizontal and vertical directions such that differences in overlap less than the threshold in that particular direction are ignored.

Despite the many useful features, there is also a limitation 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 [4]. To check the severity of the problem, we have added a dummy segmentation algorithm into the comparison that returns the whole page as one segment, as discussed in Section III-A.

We overcome this limitation by defining a vectorial score that clearly identifies the common classes of segmentation errors including the under-segmentation problem identified above. This score is based on a new representation scheme for page segmentation described in Section II-A. The vectorial score is described in Section II-C.

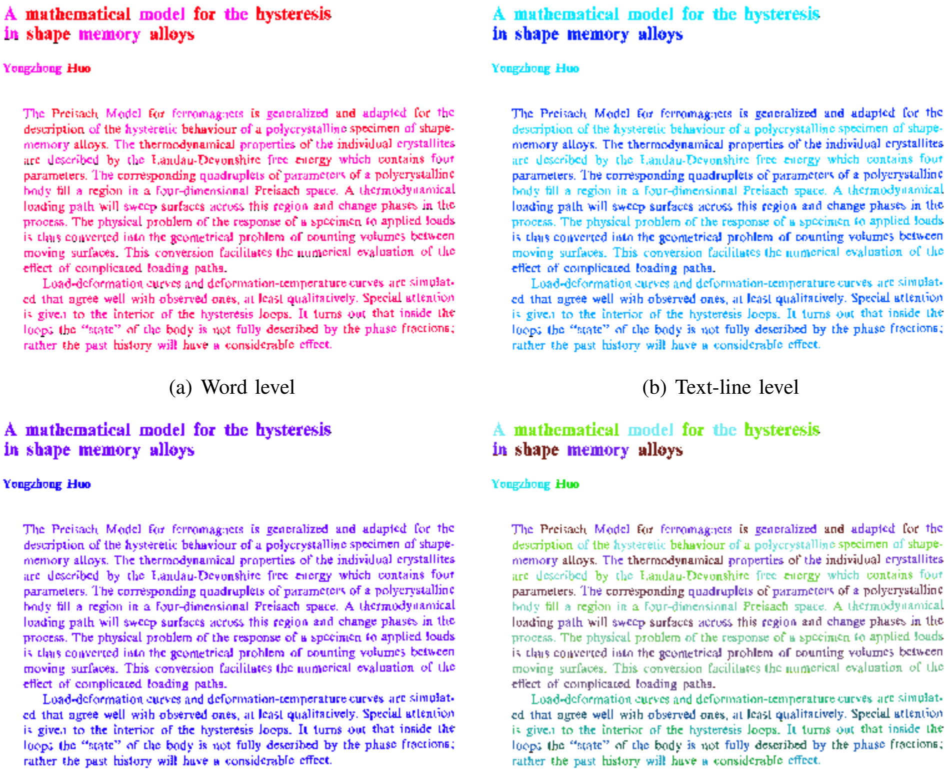
A. *Representation of page segments*

Layouts of a document image are generally categorized into two main classes: *Manhattan* layouts, and *non-Manhattan* layouts [1]. Manhattan layouts are defined as layouts that can be decomposed into individual segments by vertical and horizontal cuts. For Manhattan layouts, the individual zones can be represented by non-overlapping rectangles. This representation is particularly useful due to its simplicity and segments of most of the structured documents like technical journals or business letters can be represented by their bounding rectangles. Therefore, this representation was adapted in the Document Attribute Format Specification (DAFS) format [28] used for representing the ground-truth zones for the UW-III dataset. The DAFS format was developed with the intention to be used as a standard for the representation of document images. However, it did not come to widespread use and other representations based on XML have emerged [29] for Manhattan layouts. For non-Manhattan layouts, the zones cannot be represented accurately by non-overlapping rectangles. Instead, a XML based representation of document zones by their bounding isothetic polygons was used in [7], [8]. A common problem with these approaches is that they need specialized software to view the files representing the page segmentation, thereby limiting their portability and ease of use.

To overcome these problems, we propose a new way of representing the page segments in color image format. Consider a document image decomposed into N homogeneous zones $Z_i, i = 1, \dots, N$. The document segmentation can be represented as an image in which each foreground pixel is assigned as its value the index of the segment Z_i to which it belongs. In practice, the pixel-based representation of page segmentation can be implemented as 24-bit RGB color images. This enables the use of up to $N = 2^{24}$ labels, which will be sufficient for virtually all images that are of interest. A particular color can be assigned to the page background (e.g. 0xffffff) and to the noise pixels (e.g. 0x000000). This representation of page segmentation is particularly convenient because it can be used to accurately represent different levels of layout in the same image as shown in Figure 1. Secondly, it is independent of the zone shape and it can be saved and exchanged using any lossless color image format.

B. *Preparation of pixel-level ground-truth*

An image of a 300-dpi scanned A4 document usually contains over one million foreground pixels. The cost of coloring all foreground pixels using their respective segment label can be too high if all pixels are labelled individually. To overcome this problem, we consider two alternatives for preparing pixel-level ground-truth.



A mathematical model for the hysteresis in shape memory alloys

Yongzhong Huo

The Preisach Model for ferromagnets is generalized and adapted for the description of the hysteretic behaviour of a polycrystalline specimen of shape-memory alloys. The thermodynamical properties of the individual crystallites are described by the Landau-Devonshire free energy which contains four parameters. The corresponding quadruplets of parameters of a polycrystalline body fill a region in a four-dimensional Preisach space. A thermodynamical loading path will sweep surfaces across this region and change phases in the process. The physical problem of the response of a specimen to applied loads is thus converted into the geometrical problem of counting volumes between moving surfaces. This conversion facilitates the numerical evaluation of the effect of complicated loading paths.

Load-deformation curves and deformation-temperature curves are simulated that agree well with observed ones, at least qualitatively. Special attention is given to the interior of the hysteresis loops. It turns out that inside the loops the "state" of the body is not fully described by the phase fractions; rather the past history will have a considerable effect.

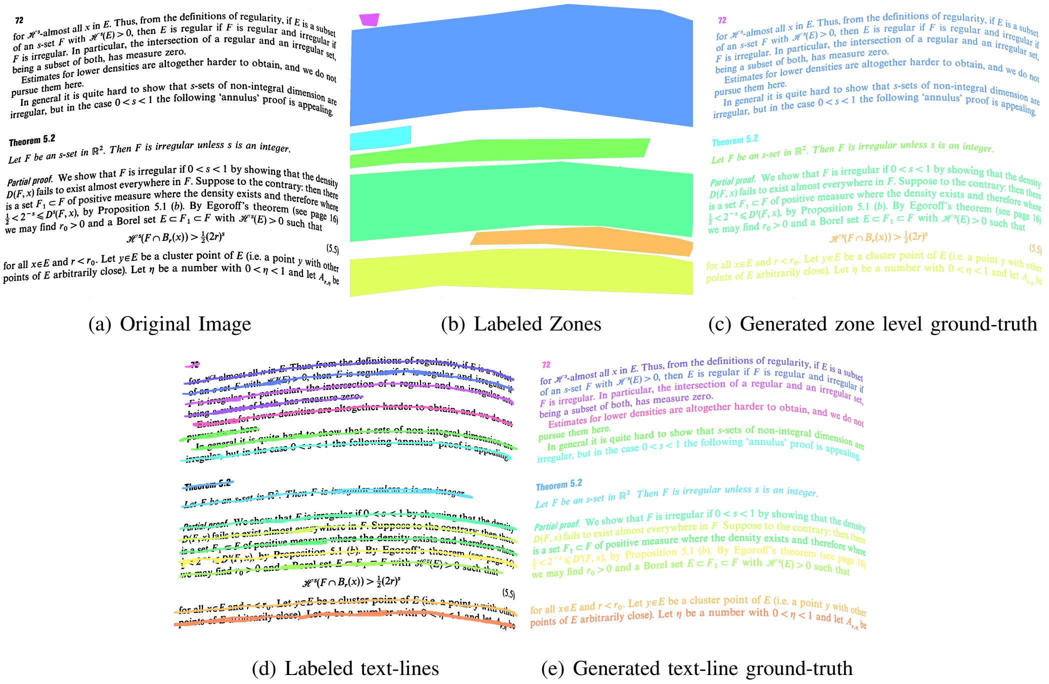
(a) Word level

(b) Text-line level

(c) Zone level

(d) Multiple layout levels

Fig. 1. An example image to demonstrate color encoding of multiple layout levels. The top images show word level and text-line level segmentation representation, whereas the bottom images show zone level and multiple layout levels information encoded in different color channels of the same image.



72

for $\mathcal{H}^s$-almost all x in E. Thus, from the definitions of regularity, if E is a subset of an s-set F with $\mathcal{H}^s(E) > 0$, then E is regular if F is regular and irregular if F is irregular. In particular, the intersection of a regular and an irregular set, being a subset of both, has measure zero.

Estimates for lower densities are altogether harder to obtain, and we do not pursue them here.

In general it is quite hard to show that s-sets of non-integral dimension are irregular, but in the case $0 < s < 1$ the following 'annulus' proof is appealing.

Theorem 5.2

Let F be an s-set in $\mathbb{R}^2$. Then F is irregular unless s is an integer.

Partial proof. We show that F is irregular if $0 < s < 1$ by showing that the density $D(F, x)$ fails to exist almost everywhere in F. Suppose to the contrary: then there is a set $F_1 \subset F$ of positive measure where the density exists and therefore where $\frac{1}{2} < 2^{-s} \leqslant D^s(F, x)$, by Proposition 5.1 (*b*). By Egoroff's theorem (see page 16) we may find $r_0 > 0$ and a Borel set $E \subset F_1 \subset F$ with $\mathcal{H}^s(E) > 0$ such that

$$\mathcal{H}^s(F \cap B_r(x)) > \tfrac{1}{2}(2r)^s \qquad (5.5)$$

for all $x \in E$ and $r < r_0$. Let $y \in E$ be a cluster point of E (i.e. a point y with other points of E arbitrarily close). Let η be a number with $0 < \eta < 1$ and let $A_{r,\eta}$ be

(a) Original Image

(b) Labeled Zones

(c) Generated zone level ground-truth

(d) Labeled text-lines

(e) Generated text-line ground-truth

Fig. 2. An example image to demonstrate the process of generating pixel-level ground-truth. The zone-level ground-truth is prepared by first drawing a polygon around each zone (Fig. 2(b)) and then transferring the colors to foreground pixels in the zone (Fig. 2(c)). The text-line level ground-truth is created by drawing lines (Fig. 2(d)) and then labeling connected components touching these lines with the line color (Fig. 2(e)).

1) A bounding polygon is drawn for each zone in the page image. The polygon is filled with a color representing the index of the zone contained inside the polygon (Figure 2(b)). Then each foreground pixel is assigned the color of the polygon that contains it (Figure 2(c)). This approach is suitable when separation between zones in a page is

significant. A benefit of preparing pixel-level ground-truth with this approach is that a polygon of any shape can be drawn. For Manhattan layouts a simple rectangle can do the task. For non-Manhattan layouts, a polygon can be drawn quickly around each zone. Hence the cost of producing ground-truth in this way is equal to the cost of producing any other bounding-box based ground-truth in the case of Manhattan layouts. For non-Manhattan layouts, the cost for producing pixel-level ground-truth can be much lower than other approaches because the polygons can be arbitrarily shaped and need not tightly enclose the containing zones.

2) If separation between page zones is not large, for instance in the case of text-lines, the approach of creating ground-truth with bounding polygons can become cumbersome. In such a situation, another approach can be taken. First, a line is drawn on a zone such that it touches or passes through all the connected components of that zone (Figure 2(d)). The color of the line is chosen to be the index of that zone. Then, connected components are extracted from the page and all the foreground pixels in a connected component are assigned the color of the line that touches or passes through that component (Figure 2(e)). In the final step, all small-sized components like i-dots, punctuation marks etc. are assigned the color of their closest neighbor if their distance to the closest neighbor is less than a threshold, chosen equal to x-height in our case. This step makes sure that any components that might not have been intersected in the first step get labeled as well.

Both the above methods for creating pixel-level ground-truth can be applied using any off-the-shelf image manipulation program like Gimp, MS-Paint, etc. These methods were applied in creating ground-truth for the DFKI-1 warped documents dataset used in the document image dewarping contest [30] held with CBDAR 2007.

C. Performance evaluation

Based on the pixel-accurate representation of page segmentation, we define several performance measures to evaluate different aspects of the behavior of a page segmentation algorithm. Consider that we are given two segmentations in image form, the hypothesized segmentation H, and the ground truth G. The images representing these segmentations should have the same dimensions, and for each corresponding pair of pixels in the two images, either both pixels should belong to the background or to the foreground. To compare the quality of a hypothesized segmentation against a ground truth segmentation, we can construct a weighted bipartite graph called *pixel-correspondence graph* [31] as follows. We associate with each color value in H or in G one node of the components in the graph, where the two components correspond to pixels of H and G respectively. Since each segment has a unique color, each node represents a unique segment (either in H or in G). A segment that is labeled with a special color like noise (see Section II-A) can be removed at this stage. Then, an edge is constructed between two nodes such that the weight of the edge equals the number of foreground pixels in the intersection of the regions covered by the two segments represented by these nodes. If their corresponding segments do not overlap in H and G, no edge is needed.

If the hypothesized segmentation H agrees perfectly with the ground truth segmentation G, then the pixel-correspondence graph will be a perfect matching. That is, each node in the two component of the graph has exactly one edge incident to it. If there are differences between the two segmentations, then the graph will not be a perfect matching. Instead, a node representing a segmentation in H or G may have multiple edges.

If P be the total number of pixels corresponding to one node (segment), M be the number of edges incident to that node, and $w_i, i = 1, 2, \ldots, M$ be the weight associated with each edge, then $P = \sum_{i=1}^{M} w_i$. For each node on either component of the graph, w_i/P gives the fraction of pixels overlapping with each of its corresponding nodes.

An edge between two nodes is considered *significant* if $w_i/P \geq t_r$ or $w_i \geq t_a$, where t_r is a relative threshold and t_a is an absolute threshold. The use of t_r allows a tolerance in the evaluation by ignoring fractional overlaps less than t_r. In practice, we have found $t_r = 0.1$ to be a good choice. However, if a segmentation algorithm completely fails and gives the whole page as one segment, regions containing less than 10% of the foreground pixels may get ignored. Therefore, an absolute threshold t_a is used to ensure that overlaps of more than t_a pixels are not ignored. The exact value of t_a can be chosen based on the properties of the document images under consideration (minimum font size, resolution, $\cdots$) and the desired geometric accuracy of the evaluation results. For the UW-III document images, we used $t_a = 500$ pixels for zone-level evaluation and $t_a = 100$ pixels for textline-level evaluation.

If there is more than one significant edge incident to a node in G or in H, the node is considered *oversegmented* or *undersegmented*, respectively. Using these definitions, we can introduce several measures for evaluating a page segmentation algorithm. An illustration of these measures is given in Figure 3. These measures are defined as follows:

Total correct segmentations (T_c): the total number of one-to-one matches between the ground truth components and the segmentation components

Total oversegmentations (T_o): the total number of significant edges that ground truth components have, minus the number of ground truth components to which at least one significant edge is incident

Total undersegmentations (T_u): the total number of significant edges that segmentation components have, minus the number of segmentation components to which at least one significant edge is incident

Oversegmented components (C_o): the number of ground truth components having more than one significant edge.

Undersegmented components (C_u): the number of segmentation components having more than one significant edge.

Missed components (C_m): the number of ground truth components that did not match any foreground component in the hypothesized segmentation.

False alarms (C_f): Number of components in the hypothesized segmentation that did not match any foreground component in the ground truth segmentation.

III. Algorithms for Page Segmentation

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

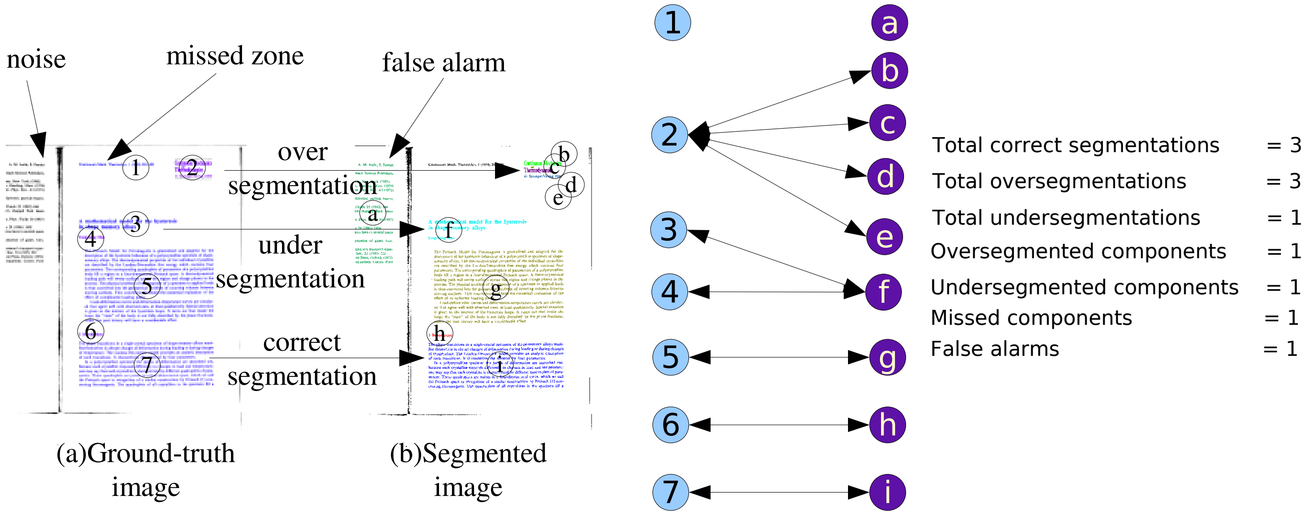


Fig. 3. Example image to illustrate different performance measures. The left image shows two color coded document images. A pixel correspondence graph obtained from these images is shown on the right side. The nodes corresponding to the ground-truth segments are labeled 1-7, whereas the nodes in the segmented image are labeled a-i. Only significant edges are shown in the pixel correspondence graph. Based on the definitions given in Section II-C, the values of each performance measure for this example are given on the right side of the graph.

A. *Dummy algorithm*

The dummy segmentation algorithm always outputs 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 IV-A.

B. *X-Y Cut*

The x-y cut segmentation algorithm [15], 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 zones which represent the nodes of the tree. At each step of the recursion, the horizontal and vertical projection profiles of each node are computed. To compute the valleys in the projection profile histograms, noise removal thresholds t_x^n and t_y^n are used. First the thresholds t_x^n and t_y^n are scaled linearly based on the current zone's width and height. Then, all bins of the histograms that contain values less than the scaled thresholds are set to zero. The valleys along the horizontal and vertical directions, v_x and v_y, are then compared to the 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.

C. *Smearing*

The run-length smearing algorithm (RLSA) [18] 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 t_{sh}, and column-wise using threshold t_{sv}, yielding two distinct bitmaps. These two bitmaps are combined in a logical AND operation. Additional horizontal smearing is done to obtain a smoothed final bitmap using a smaller threshold, t_{sm}. Then, connected component analysis is performed on this bitmap to obtain document zones. The mean horizontal run-length R_m of the black pixels in the original image, and the mean block height H_m are calculated. Then, a block is classified into a text block if

$$R < f_{tr} R_m \qquad \text{and} \qquad H < f_{th} H_m \tag{2}$$

where f_{tr} and f_{th} are two thresholds, R is the horizontal run-length of the black pixels in the current block, and H is the block height.

D. *Whitespace analysis*

The whitespace analysis algorithm described by Baird [19] 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 [20] is used in our implementation for this step. These covers are then sorted with respect to the sort key, $K(c)$:

$$K(c) = \sqrt{\text{area}(c) * W(|\log_2(\text{height}(c)/\text{width}(c))|)} \tag{3}$$

where c is the cover and $W(.)$ is a dimensionless weighting function. Baird [19] chose a special weighting function using experiments on a particular dataset. We used an approximation of the original weighting function as

$$W(x) = \begin{cases} 0.5 & \text{if } x < 3 \\ 1.5 & \text{if } 3 \leq x < 5 \\ 1 & \text{if } x \geq 5 \end{cases} \tag{4}$$

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 $c_i, i = 1, \dots, m$, where m is the total number of whitespace covers, are combined one by one to generate a corresponding sequence $s_j, j = 1, \dots, m$ 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 (5) is satisfied:

$$K(s_j) - f_w * j/m \leq t_s \tag{5}$$

where $K(s_j)$ is the sort key $K(c_j)$ of the last cover unified in making segmentation s_j, f_w is a weighting factor, and t_s is stopping threshold. At the final segmentation, connected components within the remaining uncovered parts are candidate text regions. Since the uncovered regions thus obtained are not necessarily rectangular in shape, we take bounding boxes of these uncovered regions as representative of the text segments.

E. Constrained text-line detection

The layout analysis approach by Breuel [20] finds text-lines as a three step process:

1) Find empty whitespace rectangles that completely cover the page background. The algorithm for finding maximal empty rectangles is described in [20]. The algorithm returns whitespace rectangles in order of decreasing area. The rectangles are allowed a maximum overlap of t_o. Usually 300 rectangles are sufficient to completely cover the page background.
2) The whitespace rectangles are evaluated as candidates for column separators or gutters based on their aspect ratio, width, and proximity to text-sized connected components.
3) The whitespace rectangles representing the gutters are used as obstacles in a robust least square text-line detection algorithm [32]. 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 many applications. It is also nearly parameter free and resolution independent.

F. Docstrum

The docstrum algorithm proposed by O'Gorman [16] 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 characters of the dominant font size and another one with characters in titles and section headings, using a character size ratio factor f_d. Then, K nearest neighbors are found for each connected component. A histogram of the distance and angle of each connected component from its K nearest neighbors is computed. The peak of the angle histogram gives the dominant skew in the document image. This skew estimate is used to compute within-line nearest neighbor pairs. Then, text-lines are found by computing the transitive closure on within-line nearest neighbor pairings using a threshold t_{tc}. Finally, text-lines are merged to form text blocks using a parallel distance threshold t_{pa} and a perpendicular distance threshold t_{pe}.

G. Voronoi-diagram based algorithm

The Voronoi-diagram based segmentation algorithm by Kise et al. [17] 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 r_s. Then, noise removal is done using a maximum noise zone size threshold t_n, in addition to width, height, and aspect ratio thresholds. After that a Voronoi diagram is generated using sample points obtained from the borders of the connected components. The Voronoi edges that pass through a connected component are deleted to obtain an area Voronoi diagram. Finally, superfluous Voronoi edges are deleted to obtain boundaries of document components. An edge is declared superfluous if it satisfies any of the following criterion:

1) The minimum distance d between its associated connected components is less than the inter-character gap in body text regions.
2) The minimum distance d between its associated connected components is less than the inter-line spacing times a margin control factor f_m, or the area ratio of the two connected components is above an area ratio threshold t_a.
3) At least one of its terminals is neither shared by another Voronoi edge nor lies on the edge of the document image.

The output of the algorithm consists of arbitrarily shaped regions bounded by Voronoi edges. Since we evaluate all algorithms on document pages with Manhattan layouts, we represent each Voronoi region by its bounding box.

IV. Experiments and Results

Based on the performance measures defined in Section II, we evaluated the performance of six algorithms for page segmentations, namely, x-y cut [15], the smearing algorithm [18], whitespace analysis [19], docstrum [16], the Voronoi-diagram based approach [17], and the constrained text-line finding algorithm [20]. The evaluation of the algorithms was done on the University of Washington III (UW-III) database [21].

The UW-III database consists of 1600 English document images with Manhattan layouts scanned from different archival journals with manually edited ground-truth of entity bounding boxes. These bounding boxes enclose text and non-text zones, text-lines, and words. For each document, a number of page and zone attributes are available as well. The UW-III dataset provides a good basis for comparative evaluation of page segmentation algorithms since the majority of documents available today like books, journals, magazines, letters etc. have Manhattan layouts. Researchers have used the University of Washington dataset for quantitatively evaluating different document analysis tasks like noise removal [33], [34], skew estimation [35], table recognition [24], document zone classification [22], [36], and layout-based document image retrieval [37]. Manhattan layouts pose specific constraints on page layout that can be used by page segmentation algorithms to achieve lower error rates. To evaluate the performance of page segmentation algorithms on non-Manhattan layouts, it would be useful to experiment on a dataset containing non-Manhattan layouts only. This way, one can find which algorithms are suitable for Manhattan layouts and which algorithms are suitable for non-Manhattan layouts.

We have divided the experiments into two parts:

A. *Benchmarking* of the algorithms based on text-line based measure of block segmentation accuracy given by Equation 1.
B. *Performance evaluation* of the algorithms based on the vectorial score defined in Section II-C.

The first experiment augments the work of Mao et al. [4] and adds three more algorithms to the comparison. We also do a detailed analysis of the errors to show that the limitation of the algorithm as pointed out in Section II is reflected in the results. We also show that this gives completely mis-leading results in certain cases. The second experiments demonstrates the benefits of our vectorial score based evaluation method as compared to the single-score based measure.

A. Benchmarking

The benchmarking of the page segmentation algorithms was done on a subset of the UW-III database. We chose the 978 images that correspond to the UW-I dataset pages as was done in [4]. 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 I). 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 [15]. The optimized parameter values used for x-y cut, docstrum, and Voronoi-diagram based algorithms were the same as in [4]. 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.

We have used the page segmentation evaluation toolkit (PSET) [38] that implements the training and evaluation scheme by [4]. The average text-line detection error rate for each algorithm is given in Table II. 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.

Table III 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, respectively. We can observe that the smearing, whitespace, and text-line algorithms perform 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. The respective results are shown in Table III. 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 affected by the photocopy effects. This suggests that they would perform better for current layout analysis tasks in which most documents are directly scanned.

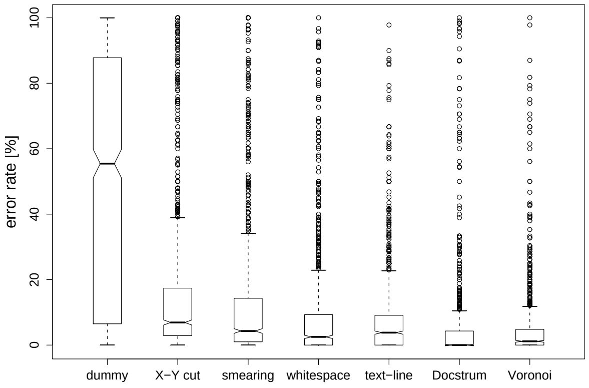


Fig. 4. Box plot for the results obtained with optimized parameters on the test data.

Figure 4 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 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. This performance 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.

To study the similarities in the behavior of different algorithms, we plot the correlation of the errors made by each algorithm in Figure 5. Each dot in the correlation plot represents one document image. The horizontal and vertical axis represent the error made by the corresponding algorithms. It can be seen from the correlation plot that docstrum and the Voronoi algorithms show strong correlation because they both are bottom-up approaches. Also, the x-y cut and the dummy algorithm are highly correlated. This is due to the fact that the x-y cut algorithm fails on documents with a large amount of noise and reports the whole page as one segment, which is the same output as generated by the dummy algorithm. When this happens for a single column document, the error rate computed by Equation 1 is zero. However, in the case of single-column documents with a large amount of noise, it is not possible to segment them into text-lines merely by horizontal projection. Hence the error rates reported in these cases give mis-leading results. An example of such a document from the test set is shown in Figure 6. Since there are only a few images in the test set that fall into this category, the experimental results are still valid. An

TABLE I

PARAMETER VALUES USED FOR EACH ALGORITHM IN THE EVALUATION GIVEN IN TABLE II. 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	$t_{sh} = 300, t_{sv} = 500, t_{sm} = 30, f_{tr} = 3, f_{th} = 3$	
Text-line	$t_o = 0.8$	
Whitespace	$f_w = 42.43, t_s = 34.29$	$f_w = 42.43, t_s = 65$
Docstrum	$K = 5, t_{tc} = 2.578, f_d = 9,$ $t_{pe} = 1.3, t_{pa} = 1.5$	$K = 8, t_{tc} = 2.578, f_d = 9,$ $t_{pe} = 0.6, t_{pa} = 2.345$
Voronoi	$s_r = 6, t_n = 11,$ $f_m = 0.34, t_a = 40$	$s_r = 6, t_n = 11,$ $f_m = 0.083, t_a = 200$

TABLE II

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. 1).

	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

TABLE III

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

interesting observation that can be made from the correlation plot is that for each algorithm, there are some documents on which it performs better than all the other algorithms. This indicates that combining the output of more than one algorithm might yield better results.

B. Performance evaluation

The performance of the six page segmentation algorithms was evaluated on the complete UW-III dataset based on the measures defined in Section II-C. These measures evaluate different aspects of a page segmentation algorithm for a given parameter setting. The goal of these performance measures is not to optimize the parameters of an algorithm on this basis because the importance of different measures is entirely application-dependent. If an OCR system expects single text-line images as input, under-segmentation (e.g. putting two consecutive lines together) poses a much more serious problem than over-segmentation (like segmenting a text-line into words). If the OCR system accepts both text-lines and text-blocks as input, the only major problem is under-segmentation (e.g. merging two text-columns). In any case, the ground-truth should also fulfil the demands of the target application. For instance, for single-line OCR text-line level ground-truth should be used. Whereas for block-level OCR either text-column or text-zone level ground-truth should be used. Since the parameters of the page segmentation algorithms given in Table I were optimized with respect to block-level OCR application, these parameters can be used in these evaluations as well. The parameters for x-y cut, whitespace analysis, docstrum, and Voronoi-diagram-based algorithms were tuned to segment text-zones. Hence, they were evaluated on zone-level ground truth with the results given in Table IV. The smearing, and the constrained textline finding algorithms locate text-lines in the given image. So they are evaluated on textline-level ground truth with the results given in Table V.

A problem with the text-zone level ground truth, in the UW-III dataset, is that a single paragraph is considered one text zone. Hence, two consecutive paragraphs on the same page make two different zones. In many documents, the segmentation of text columns into paragraphs is indicated by indentation rather than spacing. Determining paragraphs from indentations is usu-

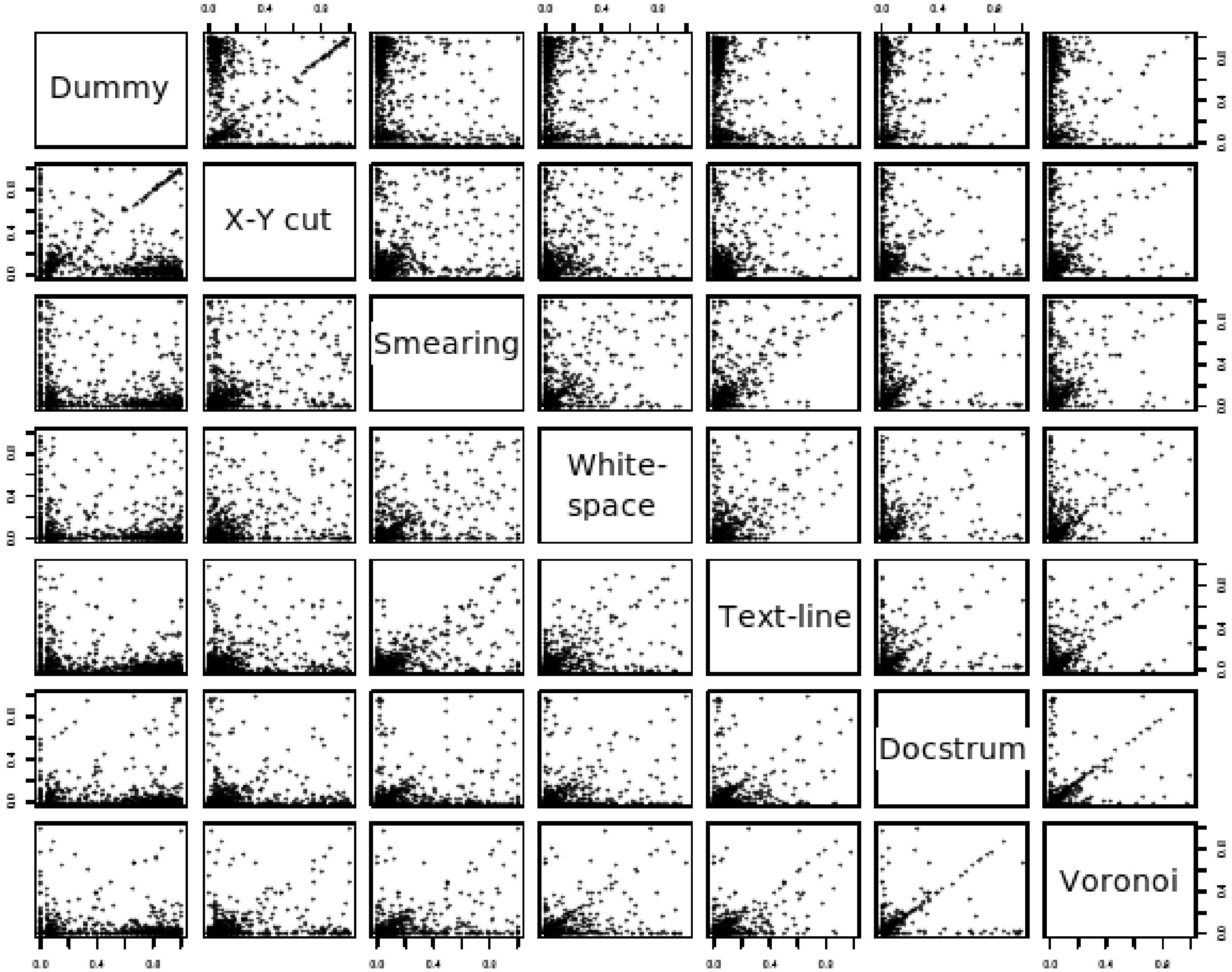


Fig. 5. Correlation plot of the errors made by each algorithm. Each dot in the correlation plot represents one document image. The horizontal and vertical axis represent the error made by the corresponding algorithms.

TABLE IV

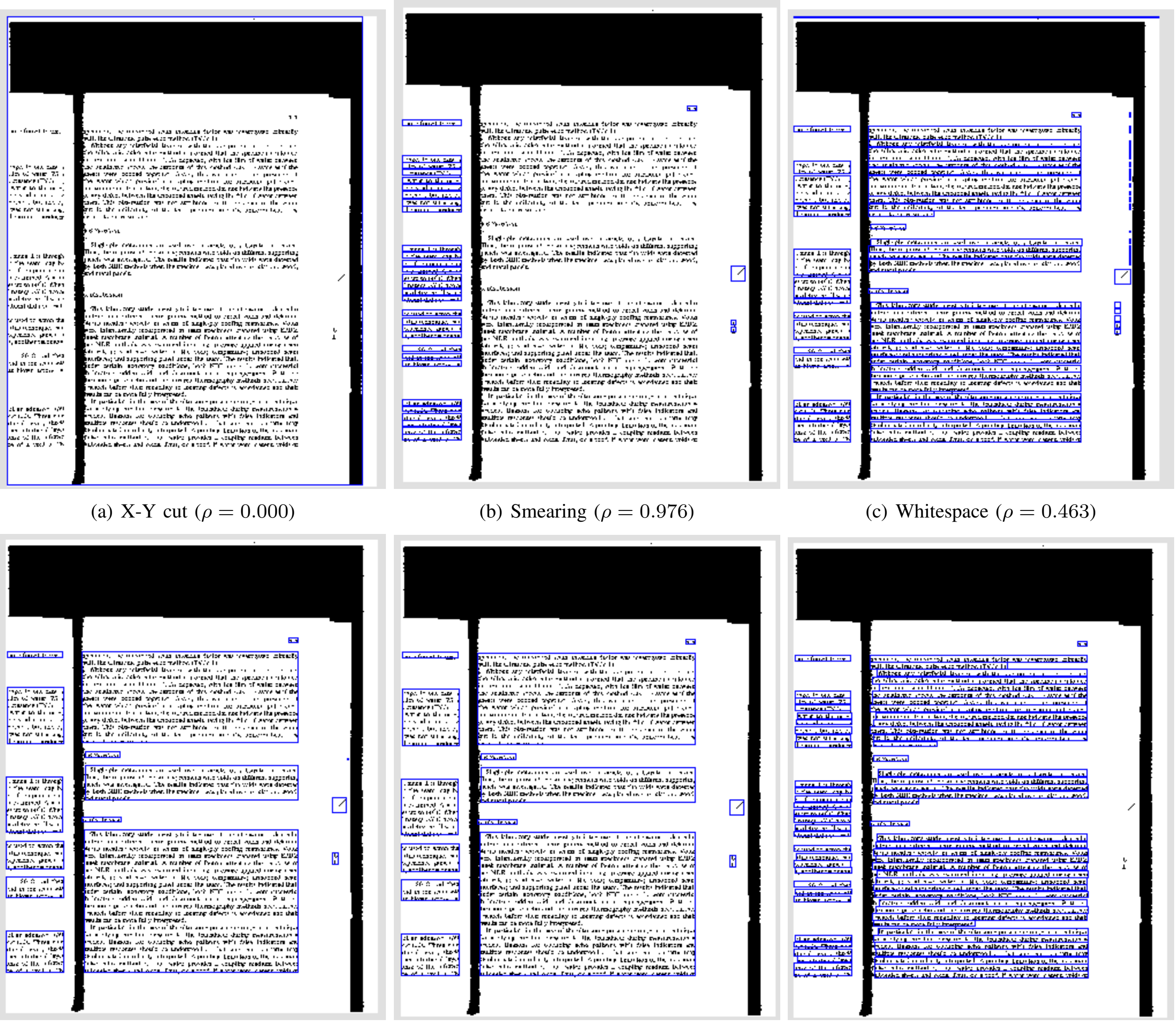
DIFFERENT TYPES OF ERRORS MADE BY EACH ALGORITHM ON ORIGINAL ZONE-LEVEL GROUNDTRUTH. EACH TEXT PARAGRAPH IS CONSIDERED A SEPARATE TEXT ZONE. ALL ENTRIES ARE NORMALIZED BY THE TOTAL NUMBER OF ZONES - 24247, AND ARE EXPRESSED IN PERCENTAGE. THE COLUMN LABELS ARE: TOTAL CORRECT SEGMENTATIONS (T_c), TOTAL OVERSEGMENTATIONS (T_o), TOTAL UNDERSEGMENTATIONS (T_u), OVERSEGMENTED COMPONENTS (C_o), UNDERSEGMENTED COMPONENTS (C_u), MISSED COMPONENTS (C_m), FALSE ALARMS (C_f)

Algorithm	Segmented zones	T_c	T_o	T_u	C_o	C_u	C_m	C_f
Dummy	6.60	0.00	0.00	93.34	0.00	6.60	0.00	0.00
X-Y cut	61.74	19.66	28.29	57.23	11.94	17.70	1.73	24.90
Whitespace	84.27	35.22	28.78	43.29	11.98	18.27	0.47	31.10
Docstrum	155.88	44.13	81.23	30.38	13.26	13.94	1.34	67.89
Voronoi	165.22	38.34	92.32	34.59	14.57	15.25	1.72	42.21

TABLE V

DIFFERENT TYPES OF ERRORS MADE BY EACH ALGORITHM ON TEXTLINE-LEVEL GROUND TRUTH. FOR A KEY TO COLUMN LABELS PLEASE REFER TO TABLE IV.. ALL ENTRIES ARE NORMALIZED BY THE TOTAL NUMBER OF TEXT-LINES - 105443, AND ARE EXPRESSED IN PERCENTAGE.

Algorithm	Segmented lines	T_c	T_o	T_u	C_o	C_u	C_m	C_f
Textline	100.13	97.17	3.77	1.64	2.82	1.23	0.26	36.05
Smearing	98.55	92.82	3.30	1.25	1.64	0.86	3.92	38.24



(a) X-Y cut ($\rho = 0.000$)

(b) Smearing ($\rho = 0.976$)

(c) Whitespace ($\rho = 0.463$)

(d) Docstrum ($\rho = 0.561$)

(e) Voronoi ($\rho = 0.561$)

(f) Text-line ($\rho = 0.756$)

Fig. 6. Segmentation results from applying each algorithm to one page image (D047) in the test set. The error rates calculated according to [4] (Equation 1) show that x-y cut algorithm performs the best in this case, which is clearly mis-leading.

TABLE VI

DIFFERENT TYPES OF ERRORS MADE BY EACH ALGORITHM ON MODIFIED ZONE-LEVEL GROUND TRUTH. FOR A KEY TO COLUMN LABELS PLEASE REFER TO TABLE IV.

Algorithm	Segmented zones	T_c	T_o	T_u	C_o	C_u	C_m	C_f
Dummy	8.05	0.00	0.00	91.88	0.00	8.04	0.00	0.00
X-Y cut	74.55	22.07	37.18	51.83	16.29	19.32	1.95	31.09
Whitespace	101.61	37.41	40.18	37.15	16.49	18.45	0.52	39.05
Docstrum	188.68	45.73	105.99	23.54	21.63	13.43	1.23	84.16
Voronoi	199.89	40.68	118.18	27.43	22.02	14.96	1.66	53.03

ally a separate processing step. Therefore, an evaluation based on paragraph-level ground truth may not correctly reflect the performance of a page segmentation algorithm by giving more undersegmentation errors than the algorithm actually made.

We modified the ground truth for UW-III to get text-zones instead of paragraphs. For this purpose, we first specified a partial order of the text paragraphs based on their spatial relationships, and then used a topological sorting algorithm to find the reading order as in [23]. Then the bounding boxes of two consecutive paragraphs in the reading order were merged if their start and end positions along the horizontal direction are within 5 pixels of each other. These modified text-zones were used to evaluate the page segmentation algorithms, with the results as shown in Table VI.

The result of applying each algorithm to an example image are shown in Figure 7. Based on the results in Tables V and VI, we can make the following observations about each algorithm.

- The dummy algorithm has no correct segmentations and all the components are under-segmented.
- The x-y cut algorithm fails in the presence of noise and tends

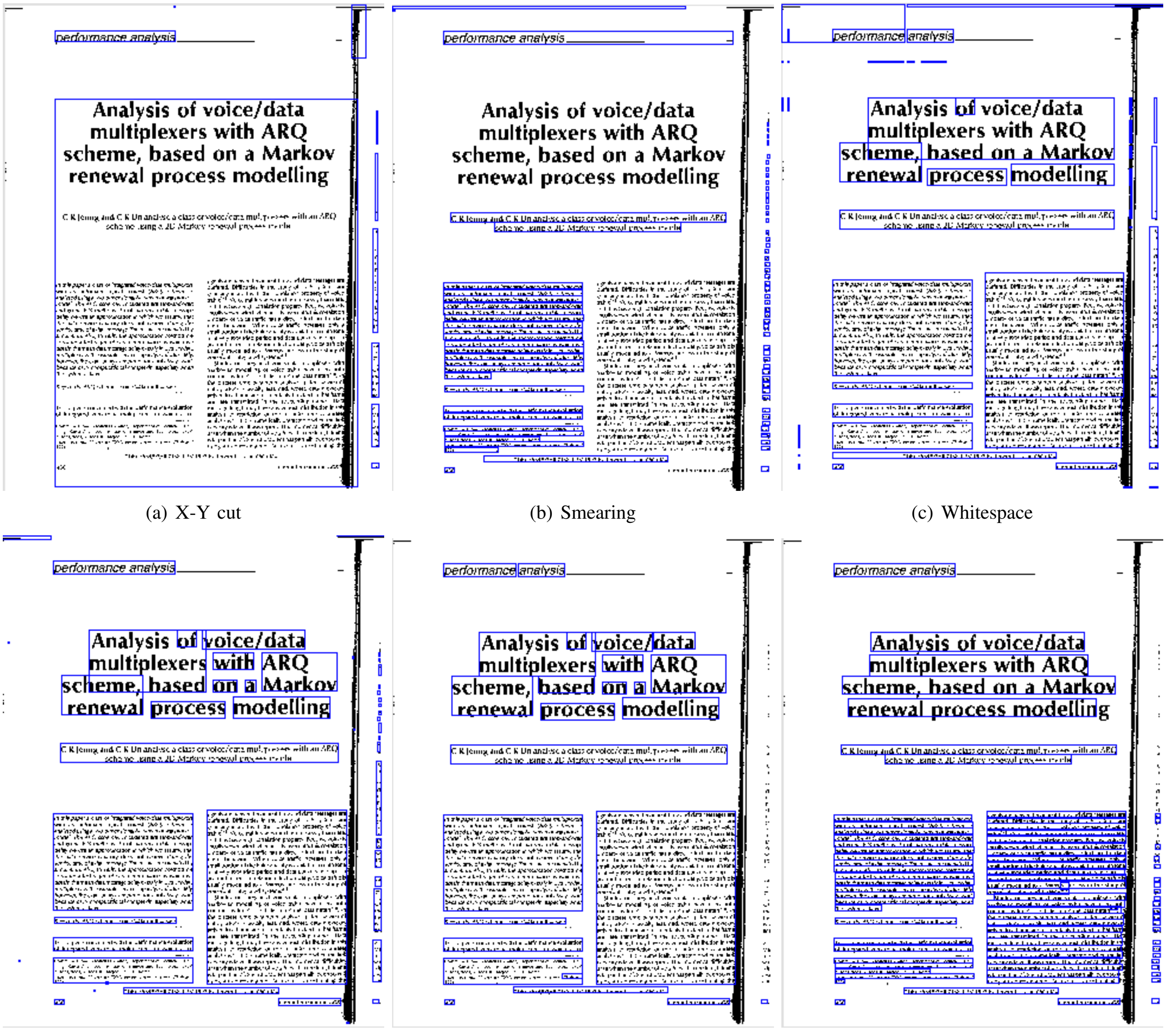

(a) X-Y cut (b) Smearing (c) Whitespace (d) Docstrum (e) Voronoi (f) Text-line

Fig. 7. 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.

to take the whole page as one segment. This results in many undersegmentation errors.

- The whitespace algorithm is sensitive to the stopping rule. Early stopping results in a higher number of undersegmentation errors, late stopping results in more over-segmentation errors. The whitespace algorithm also made few missed errors because all connected components with width larger than half the page width or height greater than half the page height were removed prior to the computation of whitespaces. Hence separator lines in header or footer, which are considered as zones in UW-III ground truth, were missed by the algorithm.
- 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 over-segmentation errors in both algorithms. For instance, in many cases, they fail to estimate the inter-line distance correctly, and hence split the zones into individual text-lines, resulting in a large number of over-segmentation errors. The number of segmented zones for these two algorithms is much higher than the number of zones in the ground truth. In some cases, text-lines in page title are incorrectly segmented (see Figure 7) due to large variation in font size.
- The smearing algorithm classifies text-lines merged with noise blocks as non-text, resulting in a large number of missed errors.
- The major part of the errors made by the constrained text-line finding algorithm are missed errors. 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 some cases, the characters from two consecutive lines are

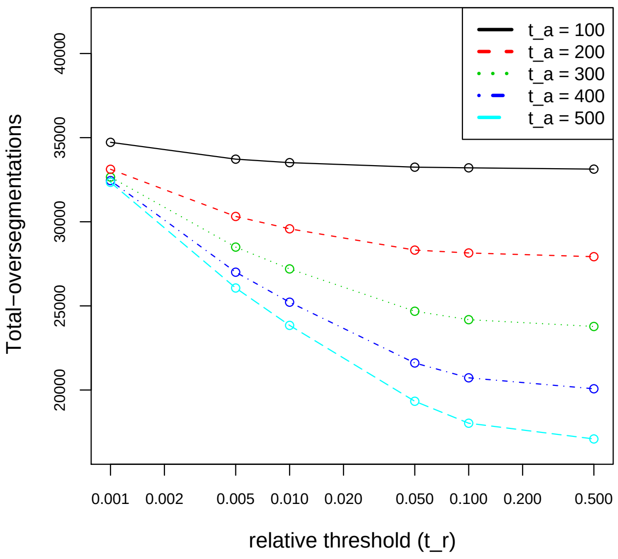


Fig. 8. A plot of the values of total over-segmentations made by the Voronoi algorithm as the values of thresholds t_r and t_a defining significant edges are changed.

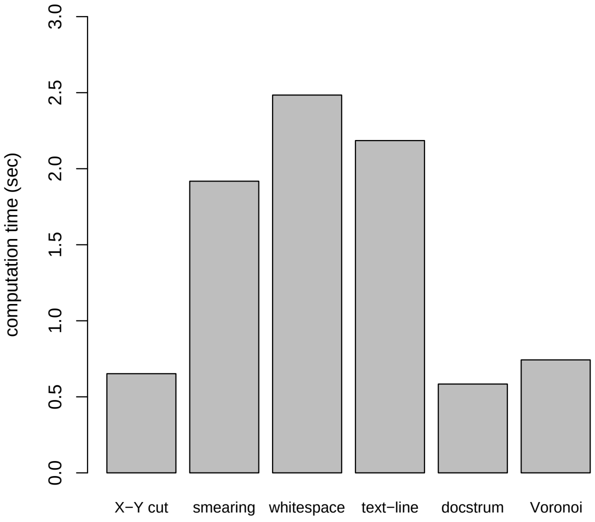


Fig. 9. Average running time for each algorithm on the UW-III dataset. The experiment was run on an AMD Opteron 2.4 GHz machine running Linux.

merged. Hence, the bounding box of the lower textline spans across both text-lines, resulting in both over-segmentation and under-segmentation errors.

The choice of the values of thresholds t_r and t_a defining significant edges is application-dependent. In the case of OCR, it might be important to keep the thresholds low so that even a missed dot is reported as an error. However, other applications like layout-based document image retrieval have less strict demands on the geometric accuracy of page segmentation. To evaluate the sensitivity of the performance measures with respect to the thresholds t_r and t_a, we have conducted an experiment. We have chosen the Voronoi algorithm as a sample page segmentation algorithm and have observed the changes in the number of reported total over-segmentation errors as the values of the thresholds t_r and t_a are varied over a broad range. The algorithm was run over the complete UW-III dataset. Then the output was compared to the zone-level ground-truth using different combinations of t_r and t_a. The resulting plot is shown in Figure 8. From the plot it can be noticed that setting either t_r or t_a to a very low value makes the performance measure independent of the other threshold. As expected the number of detected total over-segmentations decreases when the values of both thresholds are increased simultaneously. For OCR applications, just setting t_a to a very small value, for instance equal to the size of a dot, and ignoring t_r altogether might be a good choice. In the case of layout-based retrieval, we have to consider both thresholds because the size of small zones like page numbers might be smaller than a moderately chosen value of t_a. In such case t_r helps by keeping the threshold low for small zones.

The average running time of the evaluated page segmentation algorithms is shown in Figure 9. The timing of the algorithms cannot be directly compared because of the differences in their input and output. The 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 off-line and stored in the database. In general, x-y cut, docstrum, and Voronoi algorithms took less than half the time as compared to smearing, whitespace analysis, and constrained text-line finding algorithms.

C. Recommendations

Based on the experimental results and observations, the following recommendations can be made about the choice of page segmentation algorithm for different applications and document types.

- For clean documents with little or no skew, x-y cut algorithm might be a good choice as it is fast and easy to implement.
- For a homogeneous collection of documents (same resolution, similar layouts, similar font sizes and styles) with variable amount of noise, docstrum and Voronoi algorithms can be used. However, the parameters of these algorithms should be tuned to segment the given document collection to obtain good results.
- For documents containing many font sizes and styles, constrained text-line finding algorithm works best because it is based on geometric models that are invariant to font size, font style, and scan resolution.
- For a diverse document collection with documents having different font sizes and layouts or documents scanned at different resolutions, the constrained text-line finding algorithm is a good choice because it is nearly parameter-free.

- For non-Manhattan layouts, or layouts having text in different orientations, the Voronoi algorithm can be a good choice.

One problem with the evaluated page segmentation algorithms is that they give a single segmentation of a page without any confidence value. Therefore, if the output has to be verified manually, one has to look at the segmentation done for each page individually. This can be very cumbersome, even prohibitive, for large scale applications like Google book search [39]. One solution to this problem is to do page segmentation in a probabilistic framework, allowing the operator to look at only those pages for which the confidence value returned by the algorithm is low. Hence, in our opinion, an important direction of future research in page segmentation will be to develop probabilistic algorithms that can handle real-world documents.

V. Conclusion

We presented an approach for evaluating page-segmentation algorithms using color-based representation. The color-based representation of segmentation is independent of zone shape, and it can be saved and exchanged using any lossless color image format. Instead of using a single score for the performance of each algorithm, different aspects of the algorithms are evaluated separately. Depending on the target application, different error measures may be weighted according to their significance in that application. Using these performance measures, we have analyzed the strengths and weaknesses of six popular algorithms for page segmentation.

Our experiments showed that the x-y cut and the smearing algorithms fail to segment a page in the presence of noise. The whitespace analysis algorithm is sensitive to the stopping rule and results in either over-segmentations of under-segmentations. The docstrum and the Voronoi algorithms tend to over-segment title and section headings if the font size is much different from body text in that page. The constrained text-line finding algorithm misses single-digit page numbers as it requires at least two components to make a line.

Based on our experiments, we can conclude that for a homogeneous document collection with a large proportion of documents with Manhattan layouts, docstrum and Voronoi algorithms are the best choice. In the case of a heterogeneous document collection with different font sizes, styles, and scan resolutions; the constrained text-line finding algorithm appears to be the best choice.

Acknowledgments

This work was partially funded by the BMBF (German Federal Ministry of Education and Research), project IPeT (01 IW D03).

References

[1] R. Cattoni, T. Coianiz, S. Messelodi, and C. M. Modena, "Geometric layout analysis techniques for document image understanding: a review," available from http://citeseer.nj.nec.com/, IRST, Trento, Italy, Tech. Rep. 9703-09, 1998.

[2] G. Nagy, "Twenty years of document image analysis in PAMI," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 38–62, 2000.

[3] S. Mao, A. Rosenfeld, and T. Kanungo, "Document structure analysis algorithms: a literature survey," *Proc. SPIE Electronic Imaging*, vol. 5010, pp. 197–207, Jan. 2003.

[4] S. Mao and T. Kanungo, "Empirical performance evaluation methodology and its application to page segmentation algorithms," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 242–256, March 2001.

[5] F. Lotti, P. Heroux, S. Adam, G. Sanchez, E. Valveny, P. Dosch, and J. Llados, "Performance analysis and evaluation working group report," in *Document Analysis Systems*, Florence, Italy, Sep. 2004, http://www.dsi.unifi.it/DAS04/DASPerfEv.pdf.

[6] A. Antonacopoulos, D. Karatzas, and D. Bridson, "Ground truth for layout analysis performance evaluation," in *Document Analysis Systems*, Nelson, New Zealand, Feb. 2006, pp. 302–311.

[7] A. Antonacopoulos, B. Gatos, and D. Karatzas, "ICDAR 2003 page segmentation competition," in *Proc. 7th Intl. Conf. on Document Analysis and Recognition*, Edinburgh, UK, 2003, pp. 688–692.

[8] A. Antonacopoulos, B. Gatos, and D. Bridson, "ICDAR 2005 page segmentation competition," in *Proc. 8th Intl. Conf. on Document Analysis and Recognition*, Seoul, Korea, Aug. 2005, pp. 75–80.

[9] J. Kanai, T. A. Nartker, S. V. Rice, and G. Nagy, "Performance metrics for document understanding systems," in *Proc. 2nd Intl. Conf. on Document Analysis and Recognition*, Tsukuba, Japan, Oct. 1993, pp. 424–427.

[10] B. A. Yanikoglu and L. Vincent, "Ground-truthing and benchmarking document page segmentation," in *Proc. 3rd Intl. Conf. on Document Analysis and Recognition*, Montreal, Canada, Aug. 1995, pp. 601–604.

[11] J. Liang, I. T. Phillips, and R. M. Haralick, "Performance evaluation of document structure extraction algorithms," *Computer Vision and Image Understanding*, vol. 84, pp. 144–159, 2001.

[12] A. K. Das, S. K. Saha, and B. Chanda, "An empirical measure of the performance of a document image segmentation algorithm," *International Journal on Document Analysis and Recognition*, vol. 4, no. 3, pp. 183–190, 2002.

[13] A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. B. Goldgof, K. Bowyer, D. W. Eggert, A. Fitzgibbon, and R. B. Fisher, "An experimental comparison of range image segmentation algorithms," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, no. 7, pp. 673–689, 1996.

[14] X. Jiang, C. Marti, C. Irniger, and H. Bunke, "Distance measures for image segmentation evaluation," *EURASIP Journal on Applied Signal Processing*, vol. 2006, pp. Article ID 35 909, 10 pages, 2006.

[15] G. Nagy, S. Seth, and M. Viswanathan, "A prototype document image analysis system for technical journals," *Computer*, vol. 7, no. 25, pp. 10–22, 1992.

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

[17] K. Kise, A. Sato, and 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.

[18] K. Y. Wong, R. G. Casey, and F. M. Wahl, "Document analysis system," *IBM Journal of Research and Development*, vol. 26, no. 6, pp. 647–656, 1982.

[19] H. S. Baird, "Background structure in document images," in *Document Image Analysis*, H. Bunke, P. Wang, and H. S. Baird, Eds. World Scientific, Singapore, 1994, pp. 17–34.

[20] T. M. Breuel, "Two geometric algorithms for layout analysis," in *Document Analysis Systems*, Princeton, NY, Aug. 2002, pp. 188–199.

[21] I. Guyon, R. M. Haralick, J. J. Hull, and I. T. Phillips, "Data sets for OCR and document image understanding research," in *Handbook of character recognition and document image analysis*, H. Bunke and P. Wang, Eds. World Scientific, Singapore, 1997, pp. 779–799.

[22] Y. Wang, R. Haralick, and I. Phillips, "Document zone content classification and its performance evaluation," *Pattern Recognition*, vol. 39, no. 1, pp. 57–73, Jan. 2006.

[23] T. M. Breuel, "High performance document layout analysis," in *Symposium on Document Image Understanding Technology*, Greenbelt, MD, April 2003.

[24] S. Mandal, S. Chowdhury, A. Das, and B. Chanda, "A simple and effective table detection system from document images," *International Journal on Document Analysis and Recognition*, vol. 8, no. 2-3, pp. 172–182, June 2006.

[25] C. Shin and D. Doermann, "Classification of document page images," in *Symposium on Document Image Understanding Technology*, Annapolis, MD, April 1999, pp. 166–175.

[26] F. Shafait, D. Keysers, and T. M. Breuel, "Performance comparison of six algorithms for page segmentation," in *7th IAPR Workshop on Document Analysis Systems*, Nelson, New Zealand, Feb. 2006, pp. 368–379.

[27] ——, "Pixel-accurate representation and evaluation of page segmentation in document images," in *18th Int. Conf. on Pattern Recognition*, Hong Kong, China, Aug. 2006, pp. 872–875.

[28] D. Dori, D. Doermann, C. Shin, R. Haralick, I. Phillips, M. Buchman, and D. Ross, “The representation of document structure: A generic object-process analysis,” in *Handbook of character recognition and document image analysis*, H. Bunke and P. Wang, Eds. World Scientific, Singapore, 1997, pp. 421–456.

[29] G. Ford and D. Thoma, “Ground truth data for document image analysis,” in *Proceedings of 2003 Symposium on Document Image Understanding and Technology*, Greenbelt, MD, April 2003, pp. 199–205.

[30] F. Shafait and T. M. Breuel, “Document image dewarping contest,” in *2nd Int. Workshop on Camera-Based Document Analysis and Recognition*, Curitiba, Brazil, Sep. 2007, pp. 181–188.

[31] T. M. Breuel, “Representations and metrics for off-line handwriting segmentation,” in *8th International Workshop on Frontiers in Handwriting Recognition*, Ontario, Canada, Aug. 2002, pp. 428–433.

[32] ——, “Robust least square baseline finding using a branch and bound algorithm,” in *Document Recognition and Retrieval VIII, SPIE, San Jose*, 2002.

[33] L. Cinque, S. Levialdi, L. Lombardi, and S. Tanimoto, “Segmentation of page images having artifacts of photocopying and scanning,” *Patt. Recog.*, vol. 35, pp. 1167–1177, 2002.

[34] F. Shafait, J. van Beusekom, D. Keysers, and T. M. Breuel, “Page frame detection for marginal noise removal from scanned documents,” in *15th Scandinavian Conference on Image Analysis*, Aalborg, Denmark, June 2007, pp. 651–660.

[35] O. Okun, M. Pietikainen, and J. Sauvola, “Robust skew estimation on low-resolution document images,” in *5th Int. Conf. on Document Analysis and Recognition*, Bangalore, India, Sep. 1999, pp. 621–624.

[36] D. Keysers, F. Shafait, and T. M. Breuel, “Document image zone classification - a simple high-performance approach,” in *2nd Int. Conf. on Computer Vision Theory and Applications*, Barcelona, Spain, Mar. 2007, pp. 44–51.

[37] S. Marinai, E. Marino, and G. Soda, “Layout based document image retrieval by means of XY tree reduction,” in *Proc. 8th Intl. Conf. on Document Analysis and Recognition*, Seoul, Korea, Aug. 2005, pp. 432–436.

[38] S. Mao and T. Kanungo, “Software architecture of PSET: a page segmentation evaluation toolkit,” *International Journal on Document Analysis and Recognition*, vol. 4, no. 3, pp. 205–217, 2002.

[39] L. Vincent, “Google book search: Document understanding on a massive scale,” in *9th Int. Conf. on Document Analysis and Recognition*, Curitiba, Brazil, Sep. 2007, pp. 819–823.

Faisal Shafait is a PhD student in the Computer Science Department at the Technical University of Kaiserslautern and a researcher in the Image Understanding and Pattern Recognition research group at the German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany. He did his masters in Information and Communication Systems from Hamburg University of Technology (TUHH), Germany and Bachelors in Electrical Engineering from University of Engineering and Technology, Taxila, Pakistan. His research interests include image processing and pattern recognition with a focus on document image analysis.

Daniel Keysers received the Dipl. degree (with honors) in 2000 and the Dr. degree (summa cum laude) in 2006, both in computer science, from the RWTH Aachen University, Germany. During his Ph.D. studies he was with the Department of Computer Science of the RWTH and headed the image processing and understanding group at the Human Language Technology and Pattern Recognition Chair. He visited the Instituto Tecnolgico de Informtica at the Unversidad Politcnica de Valencia, Spain, in 2002 and Microsoft Live Labs in 2006. From 2005 to 2007 he was a senior researcher at the German Research Center for Artificial Intelligence (DFKI), Image Understanding and Pattern Recognition Group (IUPR), in Kaiserslautern, Germany. Currently, he is working at Google Switzerland. His research interests include pattern recognition and statistical modeling, especially for computer vision, image object recognition, image retrieval, and document processing.

Thomas M. Breuel is professor of computer science at the Technical University of Kaiserslautern Computer Science Department, head of the Image Understanding and Pattern Recognition (IUPR) research group at the DFKI, and a consultant in Palo Alto, CA, USA. His research group works in the areas of image understanding, document imaging, computer vision, and pattern recognition. Previously, he was a researcher at Xerox PARC, the IBM Almaden Research Center, IDIAP, Switzerland, as well as a consultant to the US Bureau of the Census. He is an alumnus of the Massachusetts Institute of Technology and Harvard University.