Background Variability Modeling for Statistical Layout Analysis

Faisal Shafait[1], Joost van Beusekom[2], Daniel Keysers[1], Thomas M. Breuel[1,2]
Image Understanding and Pattern Recognition (IUPR) Research Group
[1]German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany
[2]Technical University of Kaiserslautern, Germany
{faisal.shafait, daniel.keysers}@dfki.de, {joost, tmb}@iupr.net

Abstract

Geometric layout analysis plays an important role in document image understanding. Many algorithms known in literature work well on standard document images, achieving high text line segmentation accuracy on the UW-III dataset. These algorithms rely on certain assumptions about document layouts, and fail when their underlying assumptions are not met. Also, they do not provide confidence scores for their output. These two problems limit the usefulness of general purpose layout analysis methods in large scale applications. In this contribution, we propose a statistically motivated model-based trainable layout analysis system that allows assumption-free adaptation to different layout types and produces likelihood estimates of the correctness of the computed page segmentation. The performance of our approach is tested on a subset of the Google 1000 *books dataset where it achieved a text line segmentation accuracy of* 98.4% *on layouts where other general-purpose algorithms failed to do a correct segmentation.*

1 Introduction

Many methods have been proposed for geometric layout analysis of scanned documents. A recent evaluation [8] of some widely used layout analysis algorithms has shown that these methods work quite well on standard datasets.

However, these methods have a major drawback: they use implicit or explicit assumptions. One common assumption e.g. is that the inter word spacing is smaller than the inter column spacing. If one of the assumptions made by an algorithm is not fulfilled on a particular page, the method is not able to segment that page correctly. Moreover, the methods do not return any confidence of the obtained segmentation, which limits their use in practical high volume applications, as human operators have to check the results.

In this work[1] we present a model-based trainable approach for high volume page segmentation applications. Our method is able to train given models on a small training set without the need for manually labeled page segmentation. The probabilistic framework allows automatic detection of likely wrongly segmented pages.

Many researchers have focused on the use of probabilistic grammars to develop a trainable layout analysis system [4, 9, 11]. A common limitation of stochastic grammars is that optimal geometric parsing is exponential in the number of terminal symbols. Furthermore, parsing a page with stochastic grammars might result in page layouts that do not appear in practice. Other attempts for statistical modeling of page layout include the Markov Random Field (MRF) approach by Liang et al. [6], and the generative zone model approach of Gao et al. [3]. Both these approaches are capable of learning layout information from training data. However, they require large amounts of labeled training data just to capture coarse document layout structure.

Instead of trying to model generic page layouts, as done in [3,6], we take the approach of style-directed layout analysis [4, 10]. An advantage of style-directed layout analysis is that it closely resembles the document generation process, hence it can obtain better performance on a specific class of documents. We model page layout as a mixture of layout structures (Section 2.1). Then, we use a probabilistic matching algorithm to find the most likely layout of a page, given its layout model (Section 2.2). Finally, we present an EM-like training algorithm (Section 2.3) that learns geometric variability of model components from training data without the need for page segmentation ground-truth. Experimental results demonstrate the effectiveness of our method (Section 3). Section 4 concludes the paper.

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

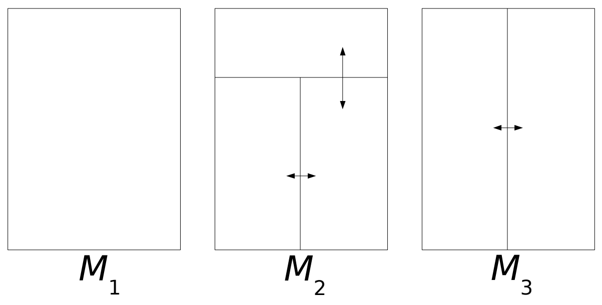

Figure 1. Representation of page layouts modeled as a mixture of layout components. The geometric variability of the components is visualized by the arrows.

2 Statistical Layout Analysis

2.1 Statistical Layout Model

In contrast to previous rule-based systems used to model document style, this work represents page layout as a statistical mixture model of layouts. Each layout is represented as a hierarchical X-Y tree of whitespace rectangles. A visualization of the model is shown in Figure 1. The focus of this work are document images with a Manhattan layout that can be represented as an X-Y tree. However, the algorithms presented here can be readily applied to non-Manhattan layouts if a suitable representation is available for them.

Background rectangles in one document type may vary from page to page. Column separators e.g. may have different height according to the content of the page, although their position and width is quite fixed. To model these variances, we represent each background rectangle by 4 parameters that are assumed to have independent Gaussian distributions. We model a layout component as a sequence of rectangles $M = \{m_1, \ldots, m_N\}$. For each rectangle 8 values are needed to define the parameters: four for the mean center position (x, y), width Δ_x, and height Δ_y of the corresponding rectangle and four for the deviations of position and size. Using an automatic training of the parameter values allows robust adaption of the model to the layout type.

2.2 Statistical Model Matching

The goal of statistical model matching is to find a set of whitespace rectangles in a target document that correspond to the layout model with the highest probability. First a whitespace cover of the page background is extracted [2]. Then, each layout component in the structural mixture model is considered as a candidate that explains the layout of the target document. We are interested in finding the layout model that explains best the target document. Last, the whitespaces corresponding to the best matching model are extracted.

Given a layout model $M = \{m_1, \ldots, m_N\}$ consisting of N model rectangles, and a set S of K whitespace rectangles $\{w_1, \ldots, w_K\}$, where $N < K$, that constitute a whitespace cover of page background. We are interested in $p(W|M)$, i.e. the likelihood of observing W given M where

- $W = (w_1, \ldots, w_N)$ is an n-tuple with $w_i \in S$ and $w_i \neq w_j \; \forall i, j : \; i \neq j$
- each element of W corresponds to an element of M

Overall, we want to find the most likely subset of whitespaces:

$$\hat{W} = \arg\max_{W} p(W|M) \tag{1}$$

The likelihood of observing whitespace rectangles $W = (w_1, \ldots, w_N)$ given a layout model $M = \{m_1, \ldots, m_N\}$ can be written as

$$\begin{aligned} p(W|M) &= p(w_1|m_1^N)p(w_2|w_1, m_1^N)\cdots \\ &\qquad p(w_N|w_1, \cdots, w_{N-1}, m_1^N) \end{aligned} \tag{2}$$

where w_i is the whitespace cover rectangle that m_i has been matched on. Due to the hierarchical structure of models, the likelihood of observing whitespace w_i does not depend on model cuts that are lower in the hierarchy, i.e. model cuts with indices $i+1$ to N. Hence, the first term on the right hand side of Equation 2 - $p(w_1|m_1^N)$ - is computed as

$$\begin{aligned} p(w_1|m_1^N) &= p(w_1|m_1) \\ &= \mathcal{N}(x_1; \mu_{x_1}, \sigma_{x_1})\mathcal{N}(y_1; \mu_{y_1}, \sigma_{y_1}) \\ &\quad \mathcal{N}(\Delta_{x1}; \mu_{\Delta_{x1}}, \sigma_{\Delta_{x1}}) \\ &\quad \mathcal{N}(\Delta_{y1}; \mu_{\Delta_{h1}}, \sigma_{\Delta_{h1}}) \end{aligned} \tag{3}$$

where $\mathcal{N}(a; \mu_a, \sigma_a)$ stands for the value of the normal distribution at point a with mean μ_a and variance σ_a. Similarly, other terms in Equation 2 can be written as:

$$p(w_j|w_1^{j-1}, m_1^N) = p(w_j|w_1^{j-1}, m_1^j) \tag{4}$$

We model the dependency between whitespace cut w_j and its ancestors by the hierarchy of the tree. The ancestors of w_j define the page segment to which the cut w_j is to be applied. The coordinates of whitespace cuts are computed relative to the page segment to which they are applied. These need to be recomputed based on the current page segment. This is done by first intersecting the whitespace w_j with the current page segment to trim its part extending beyond that segment, and then normalizing its coordinates with the page segment's width or height (x-center and width are divided by the page segment width, whereas y-center and height are divided by page segment height). The likelihood of the updated whitespace can then be computed using Equation 3.

Using Equations 3 and 4 in Equation 2 gives the likelihood of matching a particular combination of whitespaces to the layout model. The main challenge then is to find the global maximum in Equation 1. This is a combinatorial optimization problem and brute-force search to find the globally optimal solution is not practically possible. A* search is employed to find the globally optimal combination of whitespaces that best matches the layout model. Using A* search, mean running time of matching one layout model to an image is less than one second on a 2GHz PC.

Note that the likelihood of match defined by: $q = \log p(\hat{W}|M)$ will usually have lower values for complex models due to additional Gaussians involved for each model cut (see Equation 2). To avoid this problem, first the quality per cut is computed by simply dividing the log-likelihood of match by the number of model cuts. Then, the per-cut quality is normalized by the complexity of the model to give complex models a better score as compared to their submodels when both have a good matching score.

The model matching so far contains only background information. This may lead to matching, where e.g. only whitespace is separated. In order to incorporate foreground information and to avoid the segmentation of white background, the following foreground constraint has been added: a solution is regarded as valid, if at least two connected components are on each side of a vertical separator. If this is not the case, the solution is discarded.

2.3 Learning Model Parameters

Learning layout models from training images is done in two steps. In the first step, the goal is to find the structure of layout model components. This step is done by grouping documents of the same layout together and defining a structural layout model for each layout. In this preliminary work, this task is done manually. First, the user selects documents with the same layout. Then, a layout model is built for one document of that layout with the help of an interactive GUI.

In the second step, the goal is to learn geometric variability of the structural layout models built in the first step. An EM-like training algorithm is used. Consider training images $\{1, 2, \cdots, T\}$. The total quality of matching a layout model on the training set is computed as:

$$q = -\sum_{i=1}^{T} \log p_i(\hat{W}|M) \qquad (5)$$

The training algorithm tries to minimize this quantity iteratively. Starting with initial parameters, the model matching and parameter updating is done iteratively, until no improvement of quality can be made, i.e. the training algorithm terminates when $q^{(t)} \geq q^{(t-1)}$.

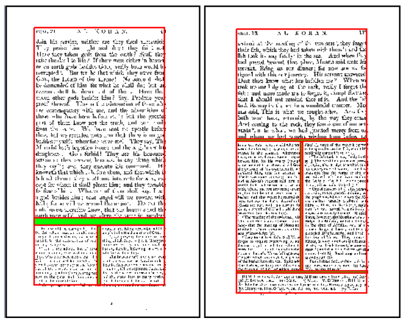

Figure 2. Structural layout components of the mixture model in our experiment.

3 Experiments and Results

To demonstrate the effectiveness of our approach, we chose a subset of the Google 1000 Books dataset [12] for our experiments. The dataset was released by Google Inc. in Sep. 2007 and contains scans of old books for which copyrights have expired. We chose volume 328 for our tests since the gap between column separators was very low for pages of this book, making it a good candidate for a non-stereotypical layout. We chose the Voronoi [5] and the Docstrum [7] algorithms as representative state-of-the-art generic layout analysis algorithms.

We used text line segmentation accuracy [8] as the error measure. Since the Google 1000 books dataset does not provide page segmentation ground-truth, we manually created zone-level ground-truth for 200 images (image index 100 to 299) from volume 328 using techniques described in [8]. Individual text lines were then automatically extracted from the zone-level ground-truth using the unconstrained text line extraction algorithm [1]. Unconstrained text line finding finds lines that stretch across the zone, avoiding oversegmentation of lines. The results of text line finding for generating the ground truth text lines were inspected manually and showed to be accurate enough for a meaningful evaluation. The same text line finding method was then used to obtain text lines from the blocks return by the Docstrum algorithm, the Voronoi algorithm, and our statistical layout matching algorithm. Differences in the obtained set of lines thus result from differing segmentations.

For our method, we picked two layout structures as components of our mixture model. Examples of these two model components are shown in Figure 2. Each of these components was trained on 10 images of the same layout structure. The results are shown in Table 1. The measures in the table are: *GT* gives the number of lines found in the ground truth; *FN* gives the number of lines found by the page segmentation in combination with the text line extractor (this is the total number of found text lines and not the

Table 1. Results for Voronoi, Docstrum and our approach on text line accuracy. The left columns give the total absolute number, the right ones the mean per page.

	Docstrum		Voronoi		Stat. Layout	
GT	12200	61.1	12200	61.1	12200	61.1
FN	10622	53.1	10033	50.2	12146	60.7
T_o	788	3.9	660	3.3	11	0.1
T_u	2370	11.8	2840	14.2	78	0.4
C_m	16	0.1	2	0.0	7	0.0
C_f	5	0.0	10	0.1	2	0.0
T_c	59.0%		51.0%		98.4%	

number of correctly found text lines); T_o gives the number of resulting line segments obtained by over-segmenting ground truth lines, e.g. a ground truth line being split into three parts will count for three oversegmentations; T_u is the number of ground truth lines being undersegmented to one line, e.g. three ground truth lines being merged into one line will be counted as three undersegmentations; C_m is the number of missed lines; C_f gives the number of falsely detected lines that are not represented in the ground truth (false alarms); T_c gives the percentage of lines that were found correctly (without over- or undersegmentation).

The results show that the segmentation results achieve a text line accuracy of 98.4%. The few errors that occur are mainly due to outliers of the layout model which our method was not trained on. The few oversegmentations result from word gaps being at exactly the same position and of the same width as the column separator tends to be. This may happen when two column footnotes are followed by the title of a new chapter. In some rare cases the wrong model was matched, or, the model was matched wrongly, resulting mainly in too long column separators. In one case, noise prevented from finding the whitespace needed to correctly match the model. This error lead to half of the undersegmentation errors. Example images can be found in Figure 3

Interestingly, Voronoi and Docstrum algorithms fail to segment the two column illustrations using their default parameters. Automatic paramter tuning of these algorithms for this layout is part of on going work and will be reported in a future publication.

4 Conclusion

In this work we presented a novel approach to document image segmentation. Our method uses trainable models and a probabilistic matching, allowing on the one hand adaptability to variations in the layout model and plausibility check of the obtained segmentation on the other hand. We demonstrated its usability on a book of the Google 1000 books dataset, with a line segmentation accuracy of 98.4%.

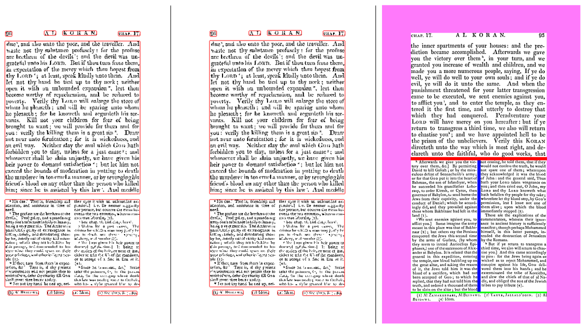

Figure 3. Examples of segmentation results obtained by Voronoi (left), Docstrum (middle), and the presented approach (right).

References

[1] T. M. Breuel. Robust least square baseline finding using a branch and bound algorithm. In *Proc. SPIE DRR IX*, pages 20–27, San Jose, CA, Jan. 2002.

[2] T. M. Breuel. Two geometric algorithms for layout analysis. In *Proc. DAS*, pages 188–199, Princeton, NY, Aug. 2002.

[3] D. Gao, Y. Wang, H. Hindi, and M. Do. Decompose document image using integer linear programming. In *Proc. ICDAR*, pages 397–401, Curitiba, Brazil, Sep. 2007.

[4] T. Kanungo and S. Mao. Stochastic language models for style-directed layout analysis of document images. *IEEE Trans. on Image Processing*, 12(5):583–596, 2003.

[5] K. Kise, A. Sato, and M. Iwata. Segmentation of page images using the area Voronoi diagram. *Computer Vision and Image Understanding*, 70(3):370–382, 1998.

[6] J. Liang, R. M. Haralick, and I. T. Phillips. A statistically based, highly accurate text-line segmentation method. In *Proc. ICDAR*, pages 551–555, Bangelore, India, Sep. 1999.

[7] L. O'Gorman. The document spectrum for page layout analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(11):1162–1173, 1993.

[8] F. Shafait, D. Keysers, and T. M. Breuel. Performance evaluation and benchmarking of six page segmentation algorithms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30(6), 2008. Accepted for publication.

[9] M. Shilman, P. Liang, and P. Viola. Learning non-generative grammatical models for document analysis. In *Proc. ICCV*, pages 962–969, Beijing, China, Oct. 2005.

[10] A. L. Spitz. Style-directed document segmentation. In *Proc. Symp. Document Image Understanding Technology*, pages 195–199, Baltimore, MD, Apr. 2001.

[11] T. Tokuyasu and P. A. Chou. Turbo recognition: a statistical approach to layout analysis. In *Proc. SPIE DRR VIII*, pages 123–129, San Jose, CA, Jan. 2001.

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