Geometric Layout Analysis of Scanned Documents

Dissertation

submitted to the
Department of Computer Science
Technical University of Kaiserslautern
for the fulfillment of the requirements for the doctoral degree
Doctor of Engineering
(Dr.-Ing.)

by

Faisal Shafait

Thesis supervisors:
Prof. Dr. Thomas Breuel, TU Kaiserslautern
Prof. Dr. Horst Bunke, University of Bern

Chair of supervisory committee:
Prof. Dr. Karsten Berns, TU Kaiserslautern

Kaiserslautern, April 22, 2008

D 386

Abstract

Layout analysis–the division of page images into text blocks, lines, and determination of their reading order–is a major performance limiting step in large scale document digitization projects. This thesis addresses this problem in several ways: it presents new performance measures to identify important classes of layout errors, evaluates the performance of state-of-the-art layout analysis algorithms, presents a number of methods to reduce the error rate and catastrophic failures occurring during layout analysis, and develops a statistically motivated, trainable layout analysis system that addresses the needs of large-scale document analysis applications. An overview of the key contributions of this thesis is as follows.

First, this thesis presents an efficient local adaptive thresholding algorithm that yields the same quality of binarization as that of state-of-the-art local binarization methods, but runs in time close to that of global thresholding methods, independent of the local window size. Tests on the UW-1 dataset demonstrate a 20-fold speedup compared to traditional local thresholding techniques.

Then, this thesis presents a new perspective for document image cleanup. Instead of trying to explicitly detect and remove marginal noise, the approach focuses on locating the *page frame*, i.e. the actual page contents area. A geometric matching algorithm is presented to extract the page frame of a structured document. It is demonstrated that incorporating page frame detection step into document processing chain results in a reduction in OCR error rates from 4.3% to 1.7% ($n = 4,831,618$ characters) on the UW-III dataset and layout-based retrieval error rates from 7.5% to 5.3% ($n = 815$ documents) on the MARG dataset.

The performance of six widely used page segmentation algorithms (x-y cut, smearing, whitespace analysis, constrained text-line finding, docstrum, and Voronoi) on the UW-III database is evaluated in this work using a state-of-the-art evaluation methodology. It is shown that current evaluation scores are insufficient for diagnosing specific errors in page segmentation and fail to identify some classes of serious segmentation errors altogether. Thus, a vectorial score is introduced that is sensitive to, and identifies, the most important classes of segmentation errors (over-, under-, and mis-segmentation) and what page components (lines, blocks, etc.) are affected. Unlike previous schemes, this evaluation method has a canonical representation of ground truth data and guarantees pixel-accurate evaluation results for arbitrary region shapes. Based on a detailed analysis of the errors made by different page segmentation algorithms, this thesis presents a novel combination of the line-based approach by Breuel [Bre02c] with the area-based approach of Baird [Bai94] which solves the over-segmentation problem in area-based approaches. This new approach achieves a mean text-line extraction error rate of 4.4% ($n = 878$ documents) on the UW-III dataset, which is the lowest among the analyzed algorithms.

This thesis also describes a simple, fast, and accurate system for document image zone classification that results from a detailed comparative analysis of performance of widely used features in document analysis and content-based image retrieval. Using a novel combination of known algorithms, an error rate of 1.46% ($n = 13,811$ zones) is achieved on the UW-III dataset in comparison to a state-of-the-art system that reports an error rate of 1.55% ($n = 24,177$ zones) using more complicated techniques.

In addition to layout analysis of Roman script documents, this work also presents the first high-performance layout analysis method for Urdu script. For that purpose a geometric text-line model for Urdu script is presented. It is shown that the method can accurately extract Urdu text-lines from documents of different layouts like prose books, poetry books, magazines, and newspapers.

Finally, this thesis presents a novel algorithm for probabilistic layout analysis that specifically addresses the needs of large-scale digitization projects. The presented approach models known page layouts as a structural mixture model. A probabilistic matching algorithm is presented that gives multiple interpretations of input layout with associated probabilities. An algorithm based on A* search is presented for finding the most likely layout of a page, given its structural layout model. For training layout models, an EM-like algorithm is presented that is capable of learning the geometric variability of layout structures from data, without the need for a page segmentation ground-truth. Evaluation of the algorithm on documents from the MARG dataset shows an accuracy of above 95% for geometric layout analysis.

To my mother,

You were my strength when I was weak
You were my voice when I couldn’t speak
You were my eyes when I couldn’t see
You saw the best there was in me
Lifted me up when I couldn’t reach
You gave me faith because you believed
I’m everything I am
Because you loved me

You gave me wings and made me fly
You touched my hand I could touch the sky
I lost my faith, you gave it back to me
You said no star was out of reach
You stood by me and I stood tall
I had your love I had it all
I’m grateful for each day you gave me
Maybe I don’t know that much
But I know this much is true
I was blessed because I was loved by you

(Diane Warren)

Acknowledgements

It is a great pleasure for me to thank Prof. Thomas Breuel, my doctoral advisor, for guiding me throughout my Ph.D. work. Very few researchers in the document analysis community put so much emphasis on principled approaches with strong statistical foundations that he has been pushing forth and it has been a great privilege working with him. His strong emphasis on publishing regularly helped me a lot in keeping my focus on scientific issues, thereby completing this work in three years. I would also like to thank Prof. Horst Bunke for agreeing to review my thesis as a referee. His comments were always very useful in improving the quality of this work.

I would like to thank Dr. Daniel Keysers for supervising my work. He remained a source of constant inspiration throughout his stay at DFKI. Despite his busy schedule he always took out time for discussions. He took keen interest in this work, and even after leaving DFKI he always gave prompt replies to my long emails. Thanks also go to Prof. Habibullah Jamal and Prof. Rolf-Rainer Grigat for giving me a good start in the field of scientific research. Their appreciation and encouragement of my ideas built the motivation in me to pursue a career in research.

I would also like to thank all the colleagues in the IUPR lab for stimulating discussions, and specially to Joost van Beusekom for sharing a lot of ideas, and for proof-reading and providing me feedback on my dissertation. Thanks also go to Jane Bensch and Ingrid Romani for their always-ready-to-help attitude and their efforts in trying to help me with both office-related and personal problems; Dr. Armin Stahl for his guidance and support especially during writing this thesis; Christian Kofler for helping me with implementations and building demonstrators for this work; Hagen Kaprykowsky for his appreciation and encouragement of my ideas; Markus Goldstein for helping me out of my Linux problems; Ilya Mezhirov and Oliver Wirjadi for discussions on statistical machine learning concepts; Aleksander Lodwich and Adrian Ulges for their critical remarks on several parts of this thesis; and Syed Saqib Bukhari for organizing the parties after my thesis defense.

Last but definitively not least I have to thank my wife Sabahat for her love, encouragement, and patience throughout my Ph.D. work; her understanding for my being absent even when I was at home; and for getting up at night and letting me sleep when Talha needed something. I also want to thank Talha for the very refreshing smiles on his face during the time I wrote this thesis. My final thanks goes to my parents for helping me start off with a good education that paved the way that led to this dissertation.

Contents

1 Introduction **1**
- 1.1 Overview of Geometric Layout Analysis 2
- 1.2 Contributions of this Dissertation 3
- 1.3 Dissertation Overview 5

2 Binarization **7**
- 2.1 Introduction and related work 7
- 2.2 Otsu's Global Thresholding Algorithm 10
- 2.3 Sauvola's Local Thresholding Technique 12
- 2.4 Binarization Using Integral Images 14
- 2.5 Experiments and Results 15
- 2.6 Summary 17

3 Page Frame Detection **19**
- 3.1 Introduction 19
- 3.2 Geometric Matching for Page Frame Detection 23
  - 3.2.1 Document Model 23
  - 3.2.2 Page Frame Model 24
  - 3.2.3 Design of Quality Function 24
  - 3.2.4 Branch-and-Bound Optimization 26
  - 3.2.5 Parameter Refinement 27
- 3.3 Performance Measures 29
  - 3.3.1 Page Frame Detection Accuracy 29
  - 3.3.2 Performance Gain in Practical Applications 31
- 3.4 Experiments and Results 32
  - 3.4.1 Page Frame Detection Accuracy 33
  - 3.4.2 Performance Gain in Practical Applications 36

3.5 Summary 39

4 Page Segmentation 41

4.1 Introduction 42

4.2 Page Segmentation Algorithms 45

4.2.1 X-Y Cut 45

4.2.2 Smearing 46

4.2.3 Whitespace Analysis 46

4.2.4 Constrained Text-Line Detection 47

4.2.5 Docstrum 48

4.2.6 Voronoi-Diagram Based Algorithm 48

4.3 Performance Evaluation 49

4.3.1 Representation of Page Segments 49

4.3.2 Preparation of Pixel-Level Ground-Truth 50

4.3.3 Evaluation Methods for Page Segmentation 53

4.3.4 Vectorial Score for Performance Evaluation 55

4.3.5 Experiments and Results 57

4.4 Page Segmentation Using Whitespace Cuts 68

4.4.1 Whitespace Cover Computation 69

4.4.2 Extraction of Vertical Separators 69

4.4.3 Extraction of Horizontal Separators 70

4.4.4 Extraction of Page Segments 71

4.4.5 Experiments and Results 71

4.5 Summary 72

5 Zone Classification 75

5.1 Introduction 76

5.2 Feature Extraction 79

5.3 Classification 80

5.4 Experiments and Results 80

5.5 Summary 88

6 Layout Analysis of Urdu Documents 89

6.1 Introduction 90

6.2 Urdu Document Layout Analysis 92

6.2.1 Geometric Model for an Urdu Text-Line 93

    - 6.2.2 Extraction of Urdu Text-Lines 94
    - 6.2.3 Reading Order Determination 96
  - 6.3 Experiments, Results, and Discussion 96
    - 6.3.1 Text-Line Detection Accuracy 97
    - 6.3.2 Reading Order Determination Accuracy 103
    - 6.3.3 Problems and Future Directions 104
  - 6.4 Summary 105

7 Statistical Layout Analysis 107

  - 7.1 Introduction and Related Work 108
  - 7.2 Statistical Layout Analysis 113
    - 7.2.1 Statistical Layout Model 113
    - 7.2.2 Statistical Model Matching 116
    - 7.2.3 Learning Model Parameters 120
  - 7.3 Experiments and Results 123
    - 7.3.1 Experiment 1 126
    - 7.3.2 Experiment 2 126
    - 7.3.3 Experiment 3 127
    - 7.3.4 Experiment 4 129
  - 7.4 Summary 129

8 Conclusions and Outlook 131

List of Figures

1.1 A visualization of the structure of this thesis illustrating the relationship between different chapters and their contribution to different areas of layout analysis. 6

2.1 A typical scanned book page exhibiting print-through from the back side of the page and non-uniform illumination near page border. 8
2.2 The effect of choosing different binarization thresholds on the image and the resulting OCR output. 9
2.3 Results of applying Otsu thresholding on some example images. 11
2.4 Results of applying Sauvola's binarization algorithm on some example images. 13
2.5 The result of applying Otsu thresholding, Sauvola binarization, and the proposed binarization algorithm on a scanned book page. 16
2.6 A comparison of mean running times of the original Sauvola binarization method, the proposed algorithm, and the Otsu binarization method on the UW-1 dataset. 17

3.1 The results of a page segmentation algorithm on pages with different amounts of global translation 20
3.2 Example image showing marginal noise along the page border 21
3.3 Example image demonstrating parameter refinement in order to adapt the parameters to text alignment 28
3.4 Example image demonstrating inclusion of non-text zones into the detected page frame 29
3.5 Some example images showing the detected page frame in yellow color. . 32
3.6 An example image showing the result of page frame detection in case of border noise overlapping the page content area. 32

3.7 Example of the original and corrected ground-truth page frames 33
3.8 Histogram of the noise level of the documents in the test set. 35
3.9 Performance of the page frame detection on different documents categorized by their noise level. 36
3.10 Screenshot of Omnipage 14 showing the recognized text of the original document (left) and the document cleaned using page frame detection (right). 37

4.1 An example image showing different segmentations of the same document 42
4.2 The OCR result of an in-correctly segmented zone containing both images and text 43
4.3 The OCR result of an in-correctly segmented zone containing multiple text-columns 44
4.4 An example image demonstrating color encoding of multiple layout levels. 51
4.5 An example image demonstrating the process of generating pixel-level ground-truth. 52
4.6 Example image illustrating different performance measures. 56
4.7 A box plot of the error rates of different page segmentation algorithms . 60
4.8 Correlation plot of the errors made by each page segmentation algorithm. 61
4.9 Segmentation results from applying each page segmentation algorithm to one page image (D047) in the test set and the corresponding error rates . 62
4.10 Segmentation results from applying each page segmentation algorithm to one page image (A005) in the test set 65
4.11 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. The plot shows that setting either t_r or t_a to a very low value makes the performance measure independent of the other threshold.(cp. [SKB08b]) 67
4.12 Average running time for each page segmentation algorithm on the UW-III dataset. 68
4.13 An example image illustrating different steps of the whitespace-cuts algorithm. 69
4.14 A bar plot showing the number of documents for each page segmentation algorithm on which it had the lowest error rate among all the compared algorithms. 73

5.1 An example page of an article containing different types of zones. 76
5.2 Examples of document image block types distinguished in this work. . . . 78
5.3 Example document pages from the UW-III database showing the presence of duplicates. 81
5.4 Examples of misclassification showing the misclassified image and its nearest neighbor from a different class. 86
5.5 Some more examples of misclassification showing the misclassified image and its nearest neighbor from a different class. 87

6.1 An example of printed Arabic and Urdu text 90
6.2 An illustration of Roman script text-line model proposed by Breuel [Bre02b] 93
6.3 An example image showing the descender distance of different characters in Nastaliq script 93
6.4 Urdu text-line modeling using two descender lines 94
6.5 An example of typeset and handwritten Urdu text 97
6.6 A comparison of text-line segmentation accuracies of the proposed technique and the x-y cut algorithm. 98
6.7 Horizontal projection of Urdu text samples 99
6.8 Results of segmenting an Urdu piece of text using the x-y cut algorithm. 99
6.9 Example images illustrating results of the proposed layout analysis system on a book and a poetry page. 101
6.10 Example image illustrating results of the proposed layout analysis system on a magazine page. 102
6.11 An example image cropped from the front page of a newspaper containing both normal and inverted text in the same image. 103

7.1 An example image from the Google 1000 books dataset with its ASCII text ground-truth. 109
7.2 Segmentation results of applying state-of-the-art page segmentation algorithms on the example image in Figure 7.1. 110
7.3 A screen-shot of processing the example document image with ABBYY FineReader 7.0 Professional Edition. 111
7.4 An illustration of modeling page layout as a structural mixture model . . 114
7.5 An example image illustrating the hierarchical tree model of page layout as used in this work. 115

7.6 Example image illustrating the extraction of page foreground segments from the layout tree using depth-first-search traversal of the tree. 117
7.7 An overview of the different steps of the presented statistical layout matching algorithm. 118
7.8 Figure illustrating the first step of learning layout models. 122
7.9 A plot showing the convergence of training algorithm. 123
7.10 An illustration of the result of learning geometric variability of structural layout models. 124
7.11 Example images from each of the journals used in the experiments. . . . 125
7.12 Plots showing the distribution of the y-center and the height parameters for horizontal whitespace cuts. 127

List of Tables

3.1 Results for the connected component based evaluation of page frame detection . . . 34
3.2 Results for the zone based evaluation of page frame detection . . . 34
3.3 Results showing the effect of using page frame detection on OCR accuracy 37
3.4 Comparison of the error rates [%] for layout-based document image retrieval with and without using page frame detection. . . . 38

4.1 Parameter values used for each page segmentation algorithm in the evaluation given in Table 4.2. . . . 58
4.2 Mean text-line detection error rates for different page segmentation algorithms on 100 train images and 878 test images. . . . 58
4.3 Text-line detection errors [%] for each page segmentation algorithm separated for one-, two-, and three-column documents, and separated for photocopies or direct scans. . . . 59
4.4 Different types of errors made by each page segmentation algorithm on original zone-level groundtruth. . . . 63
4.5 Different types of errors made by each page segmentation algorithm on textline-level ground truth. . . . 63
4.6 Different types of errors made by each page segmentation algorithm on modified zone-level ground truth. . . . 64
4.7 Evaluation results for different page segmentation algorithms on the 878 test images in terms of percentage of text-lines detection errors (Eq. 4.5). 72

5.1 Summary of UW zone classification error rates from the literature along with the number of pages, zones and block types used. . . . 81
5.2 Leave-one-out nearest neighbor error rates and extraction run-times for each feature and for combinations. . . . 83

5.3 Contingency table showing the distribution of the classification of zones of a particular type in percent. 85

6.1 The performance of the x-y cut algorithm on the Urdu documents dataset. 100
6.2 The performance of the presented text-line extraction algorithm on each category of the dataset. 100

7.1 The number of samples of each journal used in the experiments. 126
7.2 Segmentation accuracy of pages of a given journal. 127
7.3 Confusion matrix for matching multiple models using log-likelihood as the match quality. 128
7.4 Confusion matrix for normalized matching 128
7.5 Confusion matrix for documents from the MARG dataset 129

Chapter 1

Introduction

Layout analysis is often a performance-limiting step of optical character recognition (OCR) systems since the errors made at this stage propagate to all further stages of the system. Typical use cases of OCR are desktop scanning and large volume document conversion. Although, the layout analysis components of existing commercial and research OCR systems are powerful enough to meet the expectations of many users, they do not fulfill the requirements of large scale digitization tasks [Bai04, Vin07]. This dissertation presents a high performance layout analysis system that satisfies the demands of both application scenarios. The layout analysis system has a robust and highly accurate generic layout analysis component that addresses the needs of desktop scanning applications, and a statistically motivated, style-directed, and trainable layout analysis component that is specifically designed to fulfill the requirements of large-scale document analysis applications.

Paper documents like books, handwritten manuscripts, magazines, newspapers, etc. have traditionally been used as the main source for acquiring, disseminating, and preserving knowledge. The advent of personal computers has given birth to another class of documents called electronic documents. An electronic document is a representation of a document using data structures that can be understood by computers. Typical examples of electronic documents are PDF, Word, XML, E-mails, Web pages, etc.

Electronic documents offer several advantages over traditional paper documents like easier editing, retrieval, indexing, and sharing since document contents can be accessed electronically. Therefore, most documents are created today by electronic means [Hol04]. An electronic document can be converted into a paper document by means of a printing device. Converting a paper document into electronic form, on the other hand, needs a way to transform the document into data structures that can be understood by computers.

A scanning device can be used to obtain a digital image of a paper document. The transformation of a scanned document image into a structured electronic representation is a complex artificial intelligence task and is the focus of research in the field of document analysis and recognition.

A document image may contain different types of contents like text, graphics, halftones, etc. The goal of optical character recognition (OCR) is to extract text from a document image. This is achieved in two steps. The first step locates text-lines in the image and identifies their reading order. This step is called *geometric layout analysis*. In the second step, text-lines identified by the layout analysis step are fed to a character recognition engine which converts them into an appropriate format (ASCII, UTF-8, . . .). Different processes involved in layout analysis are outlined in Section 1.1. The key contributions of this thesis are presented in Section 1.2. Finally, an overview of the structure of this thesis is given in Section 1.3.

1.1 Overview of Geometric Layout Analysis

Geometric layout analysis of a document image typically involves different processes. The exact order in which these processes are applied varies from one algorithm to another. Also, some algorithms might skip one or more of these processes or apply them in a hybrid way. However, most of the layout analysis systems use these processes in some form. Therefore, a brief outline of these processes is given here.

- **Binarization** is the process that converts a given input greyscale or color document image into a bi-level representation.
- **Noise removal** is a process that tries to detect and remove noise pixels in a document that are introduced by scanning or binarization process.
- **Skew correction** is a process that detects and corrects the deviation of a document's orientation angle from the horizontal direction.
- **Page segmentation** is a process that divides a document image into homogeneous zones, each consisting of only one physical layout structure (text, graphics, pictures, . . .) while respecting the columnar structure of the document.
- **Zone classification** is a process that classifies page regions into one of a set of predefined classes (e.g. text, image, graphics, . . .).

- **Reading order determination** tries to recover the order in which a human will go through different parts (segments) of the document.

Binarization, noise removal, and skew correction are typically considered as preprocessing steps in layout analysis. The core part of geometric layout analysis consists of page segmentation and zone classification modules. Reading order determination is generally considered as a post-processing step in which simple ordering criterion can be used to identify the reading order of the detected page segments.

1.2 Contributions of this Dissertation

The main contributions that are presented in this dissertation are:

1. An efficient algorithm for local adaptive binarization (Chapter 2)
    (a) A new way of computing local thresholds is presented that makes the computation time of local thresholds independent of local window size.
    (b) The same quality of binarization result as that of state-of-the-art local binarization techniques is achieved in a time close to that of global binarization techniques.
    (c) A 20 times gain in speed as compared to a state-of-the-art local thresholding scheme is demonstrated on the UW-I database.
2. Identification of page frame detection (PFD) as a new preprocessing step in document image analysis (Chapter 3)
    (a) Use of a geometric matching algorithm for page frame detection
    (b) Demonstration of a significant reduction in OCR error rate from 4.3% to 1.7% for a commercial OCR system by using PFD
    (c) Demonstration of up to 30% lower error rates for layout-based document image retrieval by using PFD
3. A comprehensive performance evaluation of well-known page segmentation algorithms (Chapter 4)
    (a) Performance evaluation and comparison of six well-known algorithms for page segmentation using a state-of-the-art evaluation methodology [MK01].

    (b) Identification of a severe flaw in the state-of-the-art evaluation scheme when used for single column documents.
    (c) A novel, portable, and pixel-accurate representation for arbitrarily shaped page segments.
    (d) A vectorial score to identify and analyze different common classes of segmentation errors made by a page segmentation algorithm.
    (e) Identification of several specific flaws in widely used page segmentation algorithms.
    (f) A novel combination of different approaches for page segmentation that results in the lowest mean error rate on the UW-III dataset.

4. A novel combination of features for document zone classification (Chapter 5)
    (a) A detailed performance comparison of widely used features in document analysis and content-based image retrieval (CBIR) communities.
    (b) Introduction of the use of histograms for the measurements of connected components and run lengths and show that this leads to a performance increase.
    (c) Introduction of a new class of blocks containing speckles that has not been considered by other researchers. This typical class of noise is important to detect during the layout analysis especially for images of bound book pages.
    (d) Achieving an error rate of 1.5% with simple and computationally efficient features on the UW-III dataset without using duplicates, which equals the lowest error rate reported in literature [WPH06] using more complex features.

5. Development of the first high-performance layout analysis system for Urdu script (Chapter 6)
    (a) A geometric model is developed for representing an Urdu text-line.
    (b) A geometric matching method is used to extract text-lines from printed Urdu documents based on the geometric model of an Urdu text-line.
    (c) A reading-order determination algorithm for Roman script documents is modified to adapt to Urdu script.

6. A novel statistical approach for trainable layout analysis (Chapter 7)

(a) A new approach to model known page layouts as a structural mixture model is presented.

(b) A probabilistic matching algorithm is presented that gives multiple interpretations of input layout with associated probabilities.

(c) An algorithm based on A* search is presented for finding the most likely layout of a page, given its structural layout model.

(d) An interactive graphical user interface (GUI) is developed that enables the user to quickly build structural layout models.

(e) An EM-like training algorithm is presented that is capable of learning the geometric variability of structural layout models from training data *without* the need for page segmentation ground-truth.

1.3 Dissertation Overview

Figure 1.1 gives an overview of the organization of this thesis highlighting the relationship between different chapters of the thesis.

Chapters 2 and 3 describe novel methods developed in this work for preprocessing document images. Then, a high performance layout analysis system is explained in Chapters 4 and 5 that can be used to accurately segment documents with stereotypical layouts.

Using techniques well-established in segmenting Roman script documents, the first high accuracy layout analysis system for Urdu script is presented in Chapter 6.

Finally, a statistical approach to layout analysis is presented in Chapter 7 that can be trained on specific layouts and gives a probabilistic output. This method is particularly useful in large scale digitization works where the system can be trained to accurately segment thousands of pages having the same layout structure.

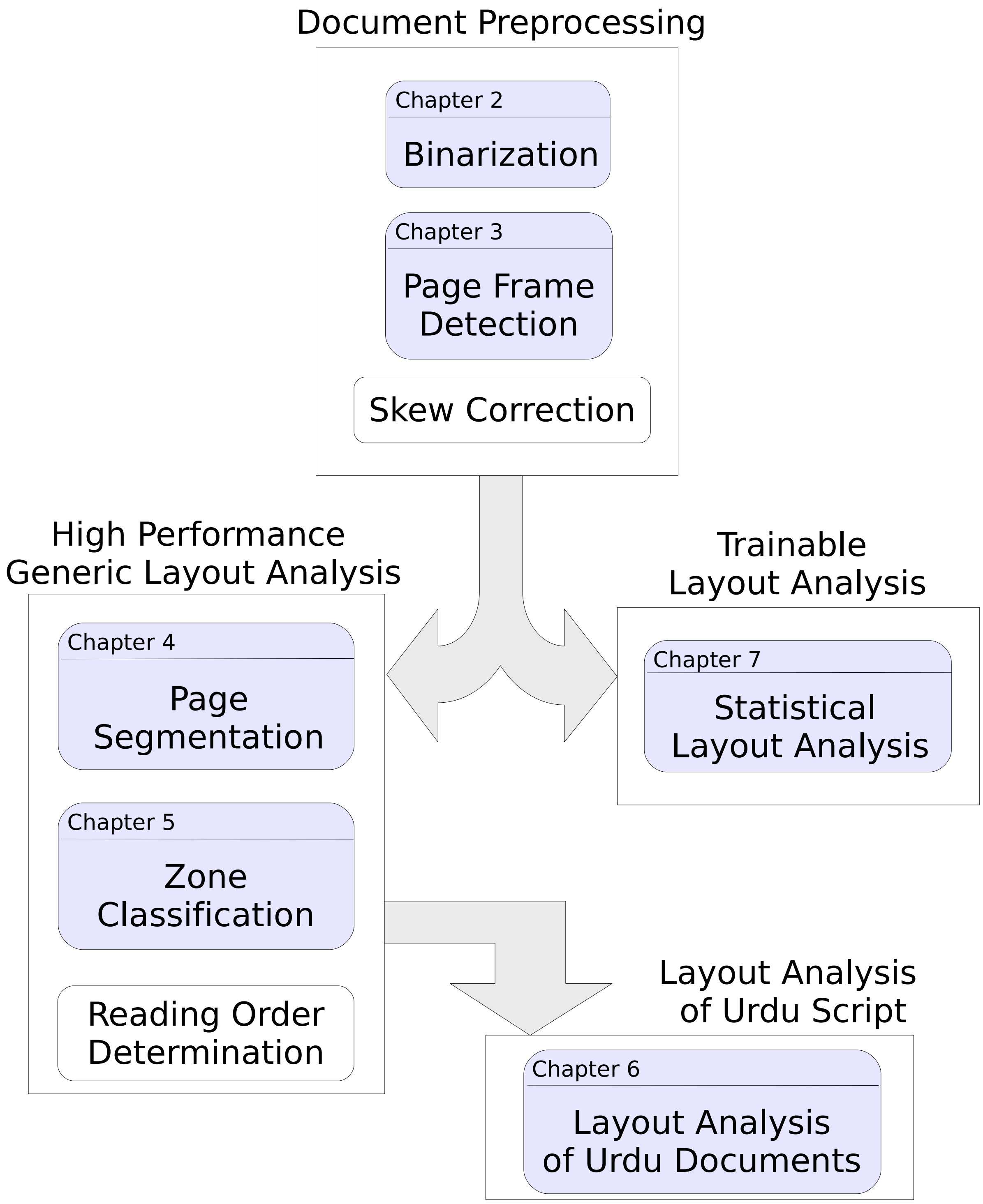

Document Preprocessing
Chapter 2
Binarization
Chapter 3
Page Frame Detection
Skew Correction
High Performance Generic Layout Analysis
Chapter 4
Page Segmentation
Chapter 5
Zone Classification
Reading Order Determination
Trainable Layout Analysis
Chapter 7
Statistical Layout Analysis
Layout Analysis of Urdu Script
Chapter 6
Layout Analysis of Urdu Documents

Figure 1.1: A visualization of the structure of this thesis illustrating the relationship between different chapters and their contribution to different areas of layout analysis. Filled blocks show the areas to which this thesis contributes.

Chapter 2

Binarization

Document binarization is an important first step in most document analysis systems. The goal of document binarization is to convert a given greyscale or color document image into a bi-level representation. This chapter presents a fast approach to binarize a document image using local adaptive thresholding. The presented approach uses the concept of sum-tables [Cro84] that were made popular in the computer vision community by Viola and Jones [VJ04]. The following contributions to the state-of-the-art in document binarization are presented in this chapter[1] :

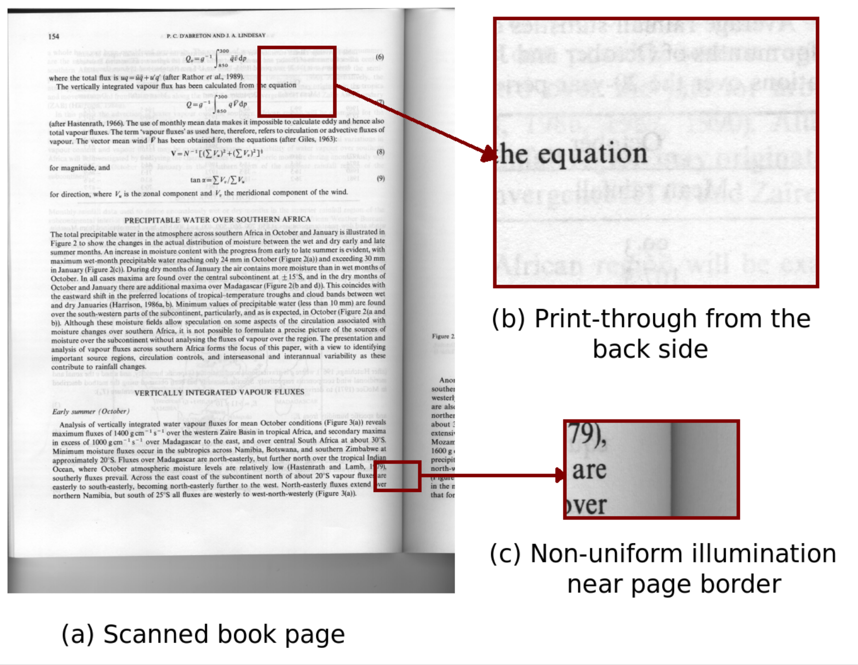
1. A new way of computing local thresholds using the concept of integral images is presented.
2. The computation time of local thresholds is made independent of local window size.
3. The same quality of binarization result as that of the state-of-the-art local binarization techniques is achieved in a time close to that of global binarization techniques.
4. A 20 times gain in speed as compared to a state-of-the-art local thresholding scheme is demonstrated on the UW-I database.

2.1 Introduction and related work

Consider a document with black text on white background. When such a paper is scanned with a flatbed scanner to convert it to digital form, noise from several sources is added to its digital counterpart. This noise comes both from the imaging mechanisms like finite

---

[1] This chapter is based on the author's work in [SKB08a].



154 P. C. D'ABRETON AND J. A. LINDESAY

$$Q_v = g^{-1}\int_{850}^{300} \bar{q}\bar{v}\,\mathrm{d}p \qquad (6)$$

where the total flux is $uq = \bar{u}\bar{q} + u'q'$ (after Rathor *et al.*, 1989).

The vertically integrated vapour flux has been calculated from the equation

$$Q = g^{-1}\int_{850}^{300} q\bar{V}\,\mathrm{d}p \qquad (7)$$

(after Hastenrath, 1966). The use of monthly mean data makes it impossible to calculate eddy and hence also total vapour fluxes. The term 'vapour fluxes' as used here, therefore, refers to circulation or advective fluxes of vapour. The vector mean wind $\bar{V}$ has been obtained from the equations (after Giles, 1963):

$$V = N^{-1}[(\sum V_u)^2 + (\sum V_v)^2]^{\frac{1}{2}} \qquad (8)$$

for magnitude, and

$$\tan\alpha = \sum V_v / \sum V_u \qquad (9)$$

for direction, where V_u is the zonal component and V_v the meridional component of the wind.

PRECIPITABLE WATER OVER SOUTHERN AFRICA

The total precipitable water in the atmosphere across southern Africa in October and January is illustrated in Figure 2 to show the changes in the actual distribution of moisture between the wet and dry early and late summer months. An increase in moisture content with the progress from early to late summer is evident, with maximum wet-month precipitable water reaching only 24 mm in October (Figure 2(a)) and exceeding 30 mm in January (Figure 2(c)). During dry months of January the air contains more moisture than in wet months of October. In all cases maxima are found over the central subcontinent at ±15°S, and in the dry months of October and January there are additional maxima over Madagascar (Figure 2(b and d)). This coincides with the eastward shift in the preferred locations of tropical–temperature troughs and cloud bands between wet and dry Januaries (Harrison, 1986a, b). Minimum values of precipitable water (less than 10 mm) are found over the south-western parts of the subcontinent, particularly, and as is expected, in October (Figure 2(a and b)). Although these moisture fields allow speculation on some aspects of the circulation associated with moisture changes over southern Africa, it is not possible to formulate a precise picture of the sources of moisture over the subcontinent without analysing the fluxes of vapour over the region. The presentation and analysis of vapour fluxes across southern Africa forms the focus of this paper, with a view to identifying important source regions, circulation controls, and interseasonal and interannual variability as these contribute to rainfall changes.

VERTICALLY INTEGRATED VAPOUR FLUXES

Early summer (October)

Analysis of vertically integrated water vapour fluxes for mean October conditions (Figure 3(a)) reveals maximum fluxes of 1400 g cm^{-1} s^{-1} over the western Zaïre Basin in tropical Africa, and secondary maxima in excess of 1000 g cm^{-1} s^{-1} over Madagascar to the east, and over central South Africa at about 30°S. Minimum moisture fluxes occur in the subtropics across Namibia, Botswana, and southern Zimbabwe at approximately 20°S. Fluxes over Madagascar are north-easterly, but further north over the tropical Indian Ocean, where October atmospheric moisture levels are relatively low (Hastenrath and Lamb, 1979), southerly fluxes prevail. Across the east coast of the subcontinent north of about 20°S vapour fluxes are easterly to south-easterly, becoming north-easterly further to the west. North-easterly fluxes extend over northern Namibia, but south of 25°S all fluxes are westerly to west-north-westerly (Figure 3(a)).

(a) Scanned book page

(b) Print-through from the back side

(c) Non-uniform illumination near page border

Figure 2.1: A typical scanned book page exhibiting print-through from the back side of the page and non-uniform illumination near page border.

spatial sampling rate, noise in electronic components, pixel sensor sensitivity variations, and from the scanning process like de-focusing, non-uniform or poor illumination, and print-through from the other side of the page [Bai00]. The effect of some of these degradations on a scanned image are shown in Figure 2.1. Although the original paper document was bi-level, the image obtained after scanning is greyscale.

A majority of document analysis systems have been developed to work on binary images [CCMM98]. The performance of subsequent steps in document analysis like page segmentation or optical character recognition (OCR) heavily depend on the result of binarization algorithm. Binarization with a high threshold results in merged components which are difficult to recognize with an OCR system. On the other hand, binarization with a low threshold results in broken characters that is again a problem for OCR. An example of OCR results on differently binarized images of the same greyscale image are shown in Figure 2.2.

Several approaches for binarizing a greyscale document [Ots79, WR83, Ber86, Nib86, O'G94, SP00] or color document [SKPB00, TL02, BNP06, KKR07] have been proposed in literature. This chapter focuses on the binarization of greyscale documents because in most cases color documents can be converted to greyscale without losing much information as far as distinction between page foreground (text) and background is concerned.

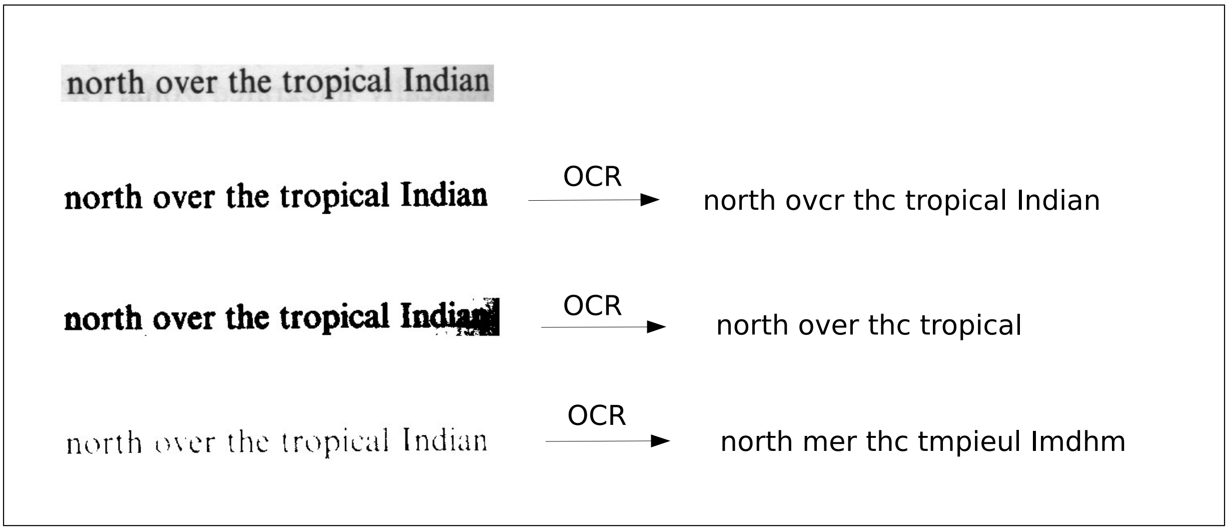

north over the tropical Indian
north over the tropical Indian
OCR
north ovcr thc tropical Indian
north over the tropical Indian
OCR
north over thc tropical
north over the tropical Indian
OCR
north mer thc tmpieul Imdhm

Figure 2.2: The effect of choosing different binarization thresholds on the image and the resulting OCR output.

The binarization techniques for greyscale documents can be grouped into two broad categories:

1. Global binarization methods
2. Local binarization methods

Global binarization methods, like that of Otsu [Ots79], try to find a single threshold value for binarizing the whole document. Each pixel in the document image is assigned to page foreground or background based on its grey value. Global binarization methods are computationally inexpensive and they give good results for typical scanned documents. However, if the illumination over the document is not uniform, for instance in the case of scanned book pages, global binarization methods tend to produce noise along the page borders [SvBKB07]. Another class of documents are historical documents in which image intensities can change significantly within a document due to poor paper quality. Local binarization methods [SP00, Ber86, Nib86] try to overcome these problems by computing thresholds individually for each pixel using information from the local neighborhood of the pixel. These methods are able to achieve good results even on severely degraded documents, but they are often slow since the computation of image features from the local neighborhood is to be done for each image pixel.

Several authors have compared different techniques for both local and global thresholding. Trier et al. [TT95] evaluated eleven different local adaptive binarization methods for greyscale images with low contrast, variable background intensity and noise.

Their evaluation showed that Niblack's method [Nib86] performed better than other local thresholding techniques. Badekas et al. [BP05] evaluated seven different algorithms for binarizing historical Greek documents. They found that Sauvola's binarization method [SP00] - which is an improvement over Niblack's method - outperforms the other local thresholding techniques; whereas Otsu's binarization method [Ots79] works best among the global binarization techniques. Another comprehensive evaluation of thresholding techniques was done by Sezgin et al. [SS04]. They have evaluated 40 different thresholding methods in the applications of non-destructive testing and document image analysis. In the document analysis application, they synthetically degraded document images using Baird's degradation model [Bai00]. They also concluded that on document images Sauvola's binarization method works better than other local binarization techniques. Based on these findings, Sauvola's binarization technique was chosen as a representative state-of-the-art technique for local thresholding, and Otsu's binarization method was selected as a representative algorithm for global thresholding.

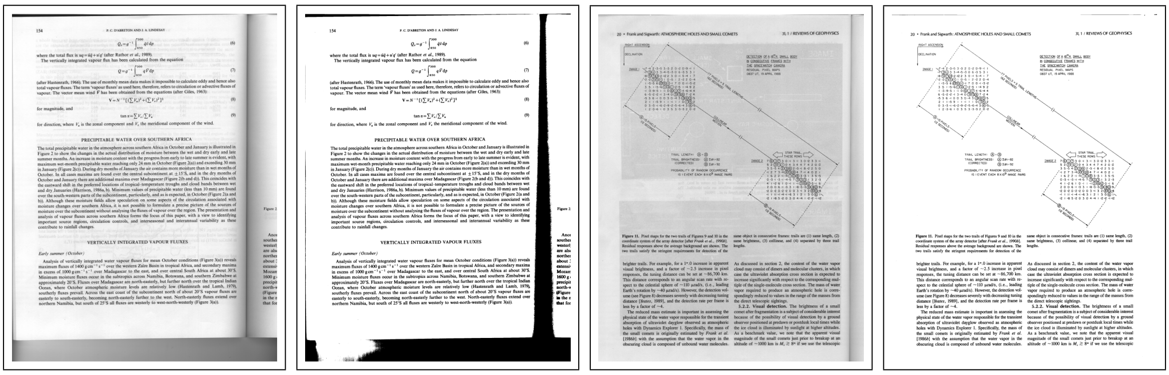
The rest of the chapter is organized as follows. Otsu's global thresholding technique is outlined in Section 2.2, followed by a description of Sauvola's local adaptive thresholding technique in Section 2.3. The main contribution of this chapter - using integral images for local thresholding - is explained in Section 2.4. The experimental setup and results are discussed in Section 2.5 followed by a conclusion in Section 2.6.

2.2 Otsu's Global Thresholding Algorithm

Consider a greyscale document image in which $g(x, y) \in [0, 255]$ be the intensity of a pixel at the location (x, y). The aim of global thresholding is to compute a threshold T such that all the pixels that have grey value lower than the threshold T are assigned to page foreground, and all pixels with grey value larger than the T are assigned to page background. Hence the pixel intensities $o(x, y)$ in the output binary image are computed as follows:

$$o(x, y) = \begin{cases} 0 & \text{if } g(x, y) \leq T \\ 255 & \text{otherwise} \end{cases} \tag{2.1}$$

Otsu's thresholding algorithm views document thresholding as a two class classification problem. All the pixels that have grey value below the threshold T belong to one class (i.e. page foreground), while all the pixels that have grey values above the threshold T belong to the other class (i.e. page background). The global threshold T is then chosen such that it maximizes the inter-class variance and minimizes the intra-class



(a) Input image (b) Otsu's result (c) Input image (d) Otsu's result

Figure 2.3: Results of applying Otsu thresholding on some example images.

variance of grey values. Let h be the normalized histogram of the given image. Then, the probabilities of the two classes defined by a threshold T are given as

$$p_1 = \sum_{g=0}^{T} h_g \tag{2.2}$$

$$p_2 = \sum_{g=T+1}^{L-1} h_g = 1 - p_1 \tag{2.3}$$

where L is the number of grey levels in the image. The mean of the two classes is given by

$$\mu_1 = \frac{1}{p_1} \sum_{g=0}^{T} g h_g \tag{2.4}$$

$$\mu_2 = \frac{1}{p_2} \sum_{g=T+1}^{L-1} g h_g \tag{2.5}$$

Then, the global threshold $\hat{T}$ chosen by the Otsu's method is given by Equation 2.6:

$$\hat{T} = \arg\max_{T} p_1 p_2 (\mu_1 - \mu_2)^2 \tag{2.6}$$

Otsu's method works well for images having a uniform illumination over the page since it adapts to the global changes in illumination in the given image. Some example images and their binarization results obtained with Otsu's technique are shown in Figure 2.3.

2.3 Sauvola's Local Thresholding Technique

The aim of local adaptive thresholding techniques is to compute a threshold $t(x, y)$ for each pixel such that

$$o(x, y) = \begin{cases} 0 & \text{if } g(x, y) \leq t(x, y) \\ 255 & \text{otherwise} \end{cases} \tag{2.7}$$

In contrast to the global thresholding methods, where a single threshold T is used to binarize the whole page (Equation 2.1), local thresholding methods allow the use of a different threshold for each pixel (Equation 2.7) so that the threshold can be adapted to local changes in illumination in different parts of the document. There are several ways in which the intensities of pixels in the local neighborhood of pixel (x, y) can be incorporated in the computation of the threshold $t(x, y)$.

White et al. [WR83] proposed to use the mean $m(x, y)$ of grey values in a $w \times w$ window centered around the pixel position (x, y):

$$t(x, y) = km(x, y) \tag{2.8}$$

where the parameter k controls the amount by which the threshold is lowered below the mean value. Usually, the value of k is set to 0.2 [SS04].

Niblack [Nib86] introduced the concept of using both local mean and standard deviation to compute local threshold. In Niblack's method, local threshold in a $w \times w$ window is computed using

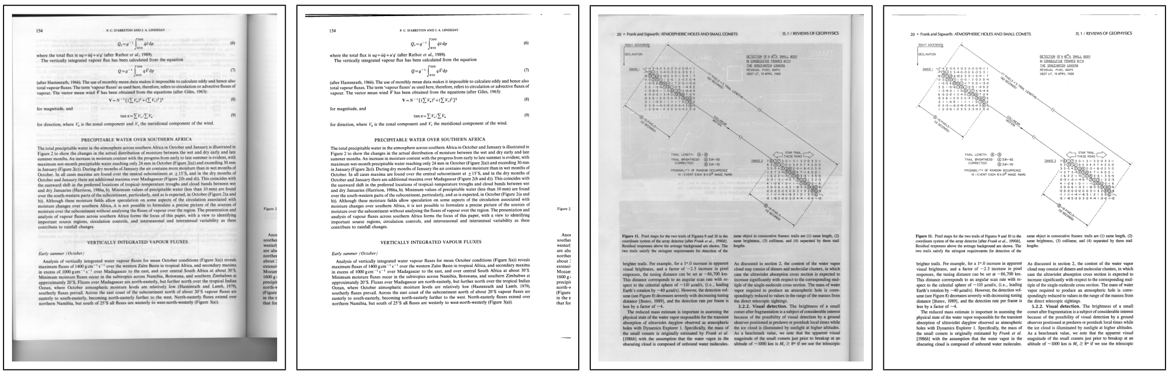
$$t(x, y) = m(x, y) + ks(x, y) \tag{2.9}$$

where $s(x, y)$ is the standard deviation of the grey values in the local window. Trier et al. [TT95] found $k = -0.2$ to be a good choice. Niblack's method works better than that of White et al. specifically in the case of poor illumination because the threshold is adapted to the local variance inside the $w \times w$ window.

Sauvola's binarization method [SP00] is an improvement of the Niblack's method. In Sauvola's method, the threshold $t(x, y)$ is computed as

$$t(x, y) = m(x, y) \left[1 + k \left(\frac{s(x, y)}{R} - 1 \right) \right] \tag{2.10}$$

where R is the maximum value of the standard deviation ($R = 128$ for a greyscale document). The parameter k takes values in the range $[0.2, 0.5]$ and controls the value



(a) Input image (b) Sauvola's result (c) Input image (d) Sauvola's result

Figure 2.4: Results of applying Sauvola's binarization algorithm on some example images.

of the threshold in the local window such that the higher the value of k, the lower the threshold from the local mean $m(x, y)$. A value of $k = 0.5$ was used by Sauvola [SP00] and Sezgin [SS04]. Badekas et al. [BP05] experimented with different values and found that $k = 0.34$ gives the best results. In general, the algorithm is not very sensitive to the value of k used.

The local mean $m(x, y)$ and standard deviation $s(x, y)$ adapt the value of the threshold according to the contrast in the local neighborhood of the pixel (x, y). For high contrast regions in the image, $s(x, y) \approx R$ which results in $t(x, y) \approx m(x, y)$. However, when the contrast in the local neighborhood is very low, the threshold $t(x, y)$ goes below the mean value thereby successfully removing the relatively dark regions of the background. Some example images binarized with Sauvola's technique are shown in Figure 2.4.

The statistical constraint in Equation 2.10 gives good results even for severely degraded documents. However in order to compute the threshold $t(x, y)$, local mean and standard deviation have to be computed for each pixel in the image. Computing $m(x, y)$ and $s(x, y)$ in a naive way results in a computational complexity of $O(w^2N^2)$ for an image of size $N \times N$ using $w \times w$ local windows. To speed up the computation, Sauvola et al. propose computing a threshold for every nth pixel and then using interpolation for the rest of the pixels. This approach speeds up the computation by some factors at the cost of reduced accuracy of determining the threshold. In addition, the computational complexity is still a quadratic function of w - the local window dimension. In the following, an efficient way of computing local means and variances using sum tables (integral images) is presented.

2.4 Binarization Using Integral Images

An integral image I of an input greyscale image G is defined as the image in which the intensity at a pixel position is equal to the sum of the intensities of all the pixels above and to the left of that position in the original image. The intensity at position (x, y) can be written as

$$g_i(x,y) = \sum_{u=0}^{x} \sum_{v=0}^{y} g(u,v) \tag{2.11}$$

The concept of integral images was popularized in the computer vision community by Viola and Jones [VJ04] based on prior work in computer graphics [Cro84]. The integral image of a greyscale image can be efficiently computed in a single pass. Using integral image, the local mean $m(x, y)$ of pixel intensities in the greyscale image can be computed simply by using two addition and one subtraction operations [PT06]:

$$\begin{aligned} m(x,y) \quad &= \quad (g_i(x+w/2, y+w/2) + g_i(x-w/2, y-w/2) - \\ &\qquad g_i(x+w/2, y-w/2) - g_i(x-w/2, y+w/2))\,/w^2 \end{aligned} \tag{2.12}$$

where the local window has dimensions $w \times w$. Similarly, the computation of the local variance can be done as:

$$s^2(x,y) = \frac{1}{w^2} \sum_{u=x-w/2}^{x+w/2} \sum_{v=y-w/2}^{y+w/2} g^2(u,v) - m^2(x,y) \tag{2.13}$$

the first term in Equation 2.13 can be computed in a similar way as Equation 2.12 by using an integral image of the squared pixel intensities.

Once the integral images of the pixel intensities and the square of the pixel intensities are obtained, local means and variances in Equation 2.10 can be computed very efficiently using Equations 2.12 and 2.13. It is important to note here that the computation of local mean and variance does not depend on the local window size anymore. Hence the same statistical constraint as that of Sauvola can be implemented in $O(N^2)$ instead of $O(w^2N^2)$. An outline of performing local adaptive thresholding using integral images is as follows:

1. Compute the integral images of the grey values and the square of the grey values in a single pass through the given greyscale image.

2. Set the local window size larger than the size of a typical character in the image.

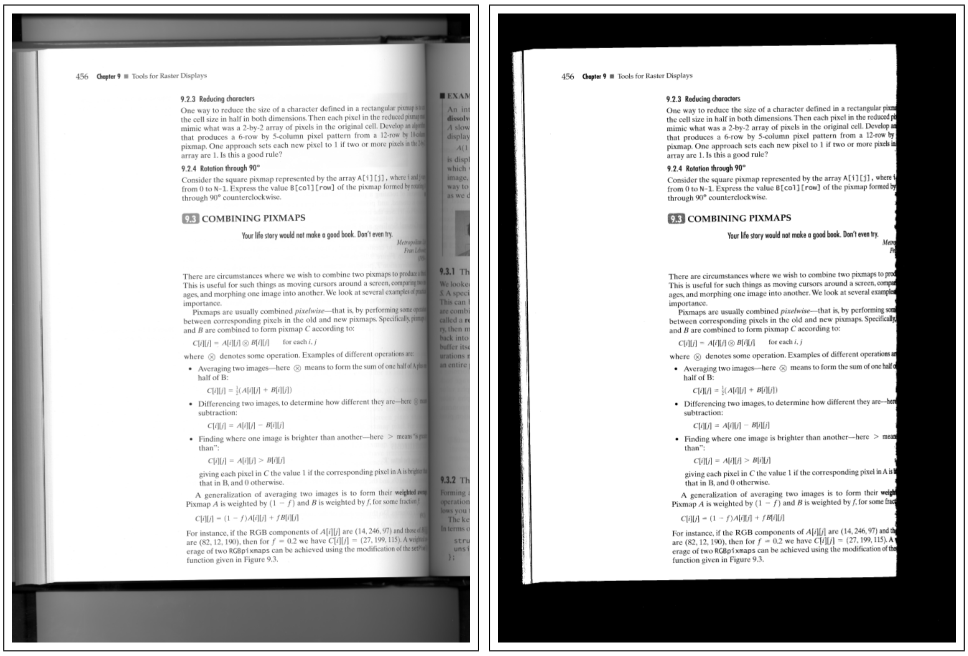
3. Compute the local threshold $t(x, y)$ for each pixel using some state-of-the-art thresholding method (e.g. Equation 2.10) in another pass through the image.
4. Obtain the output binarized document image using Equation 2.7.

An important hint from implementation point of view is that the values of the squared integral image tend to get very large, so overflow problems might occur if 32-bit integers are used. An implementation of the presented adaptive thresholding algorithm can be found in the OCRopus open source OCR system [Bre08].

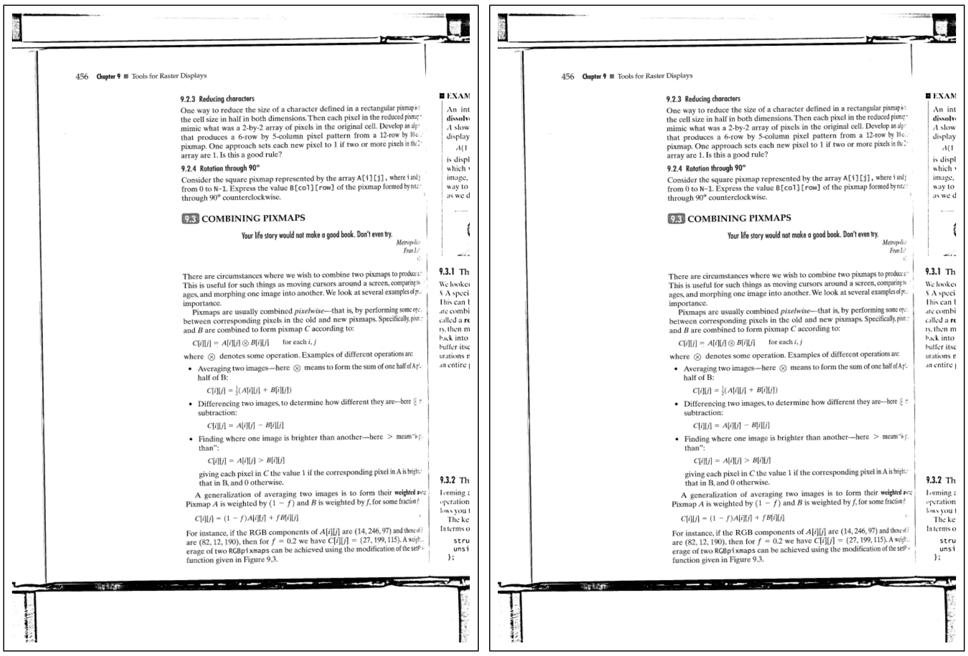
2.5 Experiments and Results

The result of running the Otsu and the Sauvola binarization algorithms on a scanned book page are shown in Figure 2.5. Illumination over the scanned greyscale image (Figure 2.5(a)) is not uniform. Hence the grey value of the pixels near the right border of the scanned page is lower than the global threshold computed by the Otsu's method, resulting in a thick black bar near the right side of the image loosing all the information in that region (Figure 2.5(b)). Sauvola's method computes a local threshold for each pixel individually taking into account image intensities in the local neighborhood of the pixel (Section 2.3). This results in a much better output (Figure 2.5(c)) at a cost of a 15-fold increase in computation time as compared to Otsu's method. The proposed algorithm (Section 2.4) achieves the same result as that of Sauvola (Figure 2.5(d)) in a time close to that of Otsu.

To measure the gain in speed over a large collection of documents, the approach presented in this chapter was tested on the University of Washington (UW-1) dataset. The UW-I dataset contains 125 greyscale images scanned at a resolution of 300-dpi. The typical dimensions of the images are 2530x3300 with little variations for each image. The experiment was conducted on an AMD Opteron 2.4 GHz machine running Linux. A comparison of mean running times of the Otsu, Sauvola, and the presented algorithm is shown in Figure 2.6. The result shows that by using the proposed algorithm, the execution time of Sauvola binarization came close to that of Otsu's binarization method. Mean running time for Otsu's binarization method was 2.0 secs whereas the presented algorithm took a mean running time of 2.8 secs. Original Sauvola algorithm took 12.6 secs when using a 15×15 local window and 65.5 secs when using a 40×40 local window. This amounts to a 5-fold speed gain in the case of small window size (15×15) and a 20-fold speed gain in the case of large window size (40×40).



(a) Input Image (2550 × 3509) (b) Otsu's result ($t = 2.7$ sec)



(c) Sauvola's result ($t = 39.2$ sec) (d) The proposed algorithm's result ($t = 2.9$ sec)

Figure 2.5: The result of applying Otsu thresholding, Sauvola binarization, and the proposed binarization algorithm on a scanned book page. Note that Sauvola's method achieves better results at a cost of a 15-fold increase in computation time as compared to Otsu's method. The proposed algorithm achieves the same result as that of Sauvola in a time close to that of Otsu.

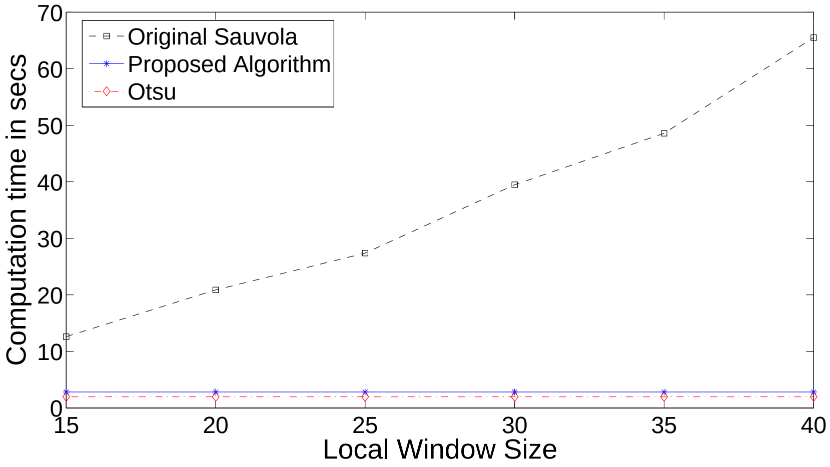


Figure 2.6: A comparison of mean running times of the original Sauvola binarization method [SP00], the proposed algorithm, and the Otsu binarization method [Ots79] on the UW-1 dataset. The graph shows that the proposed local thresholding technique achieves speed close to Otsu's binarization method. The graph also demonstrates that the computation time of the proposed technique is independent of the local window size.

Bradley et al. [BR07] have recently proposed in a parallel work to perform real-time adaptive thresholding using mean of a local window, where local mean is computed using integral image. However, for binarizing document images, local mean alone does not work as good as considering both local mean and local variance [SS04].

2.6 Summary

This chapter presented a novel way of computing thresholds for locally adaptive binarization schemes. Integral images were used to compute mean and variance in local windows, which resulted in an algorithm thats running time does not depend on the local window size. Using the threshold function of Sauvola, the presented technique achieves the same results as those of Sauvola, but in a time close to that of global binarization schemes like Otsu. The proposed technique does not only apply to the Sauvola's local thresholding algorithm, but also to other local thresholding methods, e.g. White's [WR83] or Niblack's [Nib86].

Chapter 3

Page Frame Detection

When a page of a book is scanned or photocopied, textual noise (extraneous symbols from the neighboring page) and/or non-textual noise (black borders, speckles, ...) appear along the border of the document. Let the page frame of a scanned document be defined as the smallest rectangle that encloses all the foreground elements of the document image. The goal of page frame detection is to find the actual page contents area, ignoring marginal noise along the page border. This chapter introduces page frame detection as a new step in document image analysis and demonstrates that using page frame detection as a pre-processing step significantly reduces error rates in practical applications. The main contributions of this chapter are[1]:

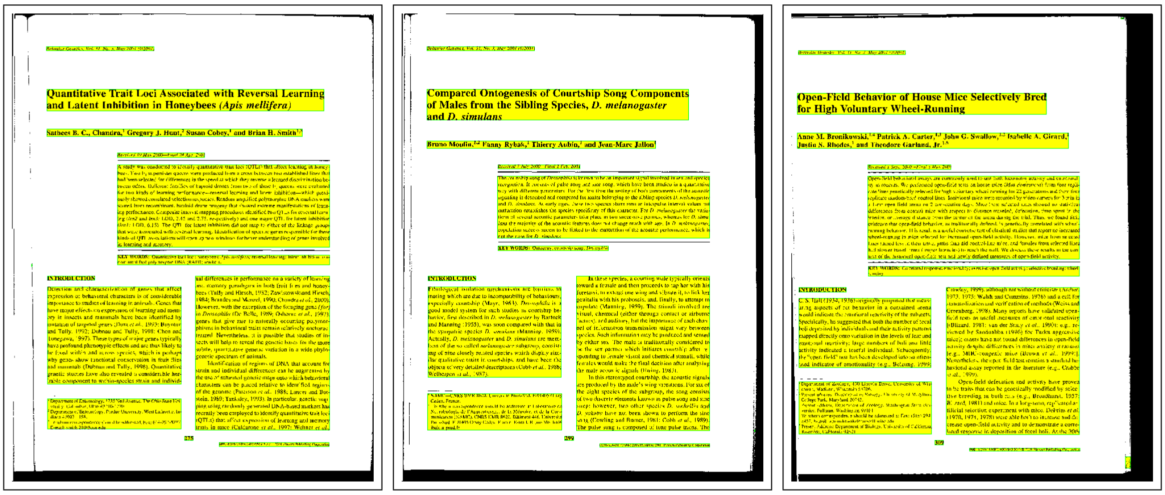
1. Identification of page frame detection as a new step in document image analysis
2. Use of a geometric matching algorithm for page frame detection
3. Demonstration of a reduction in OCR error rate from 4.3% to 1.7% ($n = 1600$) for a commercial OCR system
4. Demonstration of up to 30% lower error rates for layout-based document image retrieval

3.1 Introduction

Paper positioning variations is a class of document degradations that results in a skew and translation of the page contents in the scanned image. Document skew detection

---

[1] This chapter is based on the author's work in [SvBKB07].



Quantitative Trait Loci Associated with Reversal Learning and Latent Inhibition in Honeybees (*Apis mellifera*)

Sathees B. C., Chandra,[1] Gregory J. Hunt,[2] Susan Cobey,[1] and Brian H. Smith[1,3]

Compared Ontogenesis of Courtship Song Components of Males from the Sibling Species, *D. melanogaster* and *D. simulans*

Bruno Moulin,[1,2] Fanny Rybak,[1] Thierry Aubin,[1] and Jean-Marc Jallon[1]

Open-Field Behavior of House Mice Selectively Bred for High Voluntary Wheel-Running

Anne M. Bronikowski,[1,4] Patrick A. Carter,[1,3] John G. Swallow,[1,2] Isabelle A. Girard,[1] Justin S. Rhodes,[1] and Theodore Garland, Jr.[1,5]

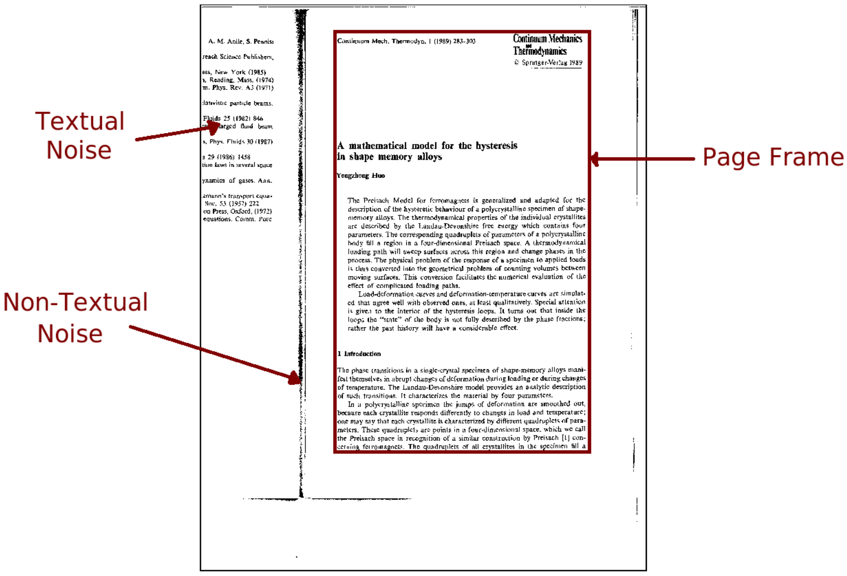
Figure 3.1: Example images showing the results of a page segmentation algorithm on pages with different amounts of global translation. The results show that the algorithm identifies the page blocks quite well in each case irrespective of the translation in the page.

and correction has received a lot of attention in last decades and several skew estimation techniques have been proposed in the literature (for a literature survey, please refer to [CCMM98]). However, estimating the global position of the page has been largely ignored by the document analysis community. This is perhaps due to the fact that most of the layout analysis methods are robust to global translation of the page and would produce the same segmentation of the page for different translations as long as all page contents are visible. Hence the OCR output is usually not affected by global translation of the page. This effect can be seen in Figure 3.1, where a page segmentation algorithm is shown to correctly identify the page segments irrespective of the translation of the page in each image.

Different amount of noise can be present along the border of a document image depending on the position of the paper on the scanner. Figure 3.1 shows the effect of paper positioning variations on the amount of marginal noise in the resulting scanned image. In general, marginal noise along the page border can be classified into two broad categories based on its source:

- non-textual noise (black bars, speckles, . . .) resulting from the binarization process
- textual noise coming from the neighboring page

An example image showing textual and non-textual noise along the page border is shown in Figure 3.2.



A. M. Anile, S. Pennisi
reach Science Publishers,
ess, New York (1985)
n, Reading, Mass. (1974)
m. Phys. Rev. A5 (1971)
lativistic particle beams.
Fluids 25 (1982) 846
arged fluid beam
s, Phys. Fluids 30 (1987)
s 29 (1986) 1458
tion laws in several space
ynamics of gases. Ann.
zmann's transport equa-
Soc. 53 (1957) 222
on Press, Oxford, (1972)
equations. Comm. Pure

Continuum Mech. Thermodyn. 1 (1989) 283-303

Continuum Mechanics and Thermodynamics
© Springer-Verlag 1989

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.

1 Introduction

The phase transitions in a single-crystal specimen of shape-memory alloys manifest themselves in abrupt changes of deformation during loading or during changes of temperature. The Landau-Devonshire model provides an analytic description of such transitions. It characterizes the material by four parameters.

In a polycrystalline specimen the jumps of deformation are smoothed out, because each crystallite responds differently to changes in load and temperature; one may say that each crystallite is characterized by different quadruplets of parameters. These quadruplets are points in a four-dimensional space, which we call the Preisach space in recognition of a similar construction by Preisach [1] concerning ferromagnets. The quadruplets of all crystallites in the specimen fill a

Figure 3.2: Example image showing textual and non-textual noise along the page border

The most common approach to dealing with non-textual noise is to perform document cleaning by filtering out connected components based on their size and aspect ratio [Bai94, Bre02c,O'G93]. This usually works out quite well as pointed out by Nagy, "Additive noise and isolated specks have been filtered and averaged out and are now on the endangered species list" [Nag00]. However, when characters from the adjacent page are also present, they usually cannot be filtered out using these approaches.

State-of-the-art page segmentation algorithms report textual noise regions as text-zones [SKB06b]. Hence, the OCR accuracy decreases in the presence of textual noise, because the OCR system outputs several extra characters in these regions.

Textual noise can be avoided altogether by scanning only the page contents area. Typical desktop scanners come with a graphical user interface to allow the users to conveniently mark the region to be scanned. This allows the user to manually select the page frame during document scanning. The resulting document image is then free of textual noise. However, if a large number of documents have to be scanned, manually defining the page frame for each one of them might become quite cumbersome.

Researchers have also tried to explicitly detect and remove marginal noise in scanned documents. For example, Le et al. [LTW96] have proposed a rule-based algorithm using several heuristics to detect the page borders. The algorithm relies upon the classification of document rows and columns into blank, textual or non-textual classes. Then, an analysis of projection profiles and crossing counts is done to detect the marginal noise. Their approach is based on the assumption that the marginal noise is very close to the edges of the image and borders are separated from image contents by a large whitespace, i.e. the borders do not overlap the edges of an image content area. However, this assumption is often violated when pages from a thick book are scanned. Avila et al. [AL04] and Fan et al. [FWL02] propose techniques for removing non-textual noise overlapping the page content area, but do not consider textual noise removal. Cinque et al. [CLLT02] propose an algorithm for removing both textual and non-textual noise from greyscale images based on image statistics like horizontal/vertical difference vectors and row luminosities. However, their method is not suitable for cleaning binary images. Also, their approach is very sensitive to the amount of noise present in the document image and the error rates increases monotonically with the artifact area.

This chapter presents a new approach for dealing with paper positioning variations in scanned documents. Instead of identifying and removing noisy components themselves, the proposed method focuses on identifying the actual content area. This is accomplished by a geometric matching algorithm. Including page frame detection as a document preprocessing step can help to increase OCR accuracy by removing textual noise from the document. Also in applications like document image retrieval based on layout information [vBKSB06], noise regions result in incorrect matches. Using the page frame to reject zones originating from noise can therefore reduce the retrieval error rates.

The method for page frame detection takes advantage of structure in a printed document to locate its page frame. This is done in several steps. First, a geometric model is built for the page frame of a scanned document. Then, a geometric matching method is used to find the globally optimal page frame with respect to a defined quality function.

The use of geometric matching for page frame detection has several advantages. Instead of devising carefully crafted rules, the page frame detection problem is solved in a more general framework, thus allowing higher performance on a more diverse collection of documents. Additionally, the use of geometric model for page frame detection makes the presented approach very robust to the amount of noise present in a document image and can find the page frame even if noise overlaps some regions of the page content area.

The rest of this chapter is organized as follows. Section 3.2.1 briefly describes the document model used in this work and how the page frame relates to the document model. Section 3.2 describes the method for page frame detection in detail. In Section 3.3, several error measures to evaluate the performance of a page frame detection algorithm are proposed. Section 3.4 presents the experimental protocol and discusses the results obtained, followed by the conclusion in Section 3.5.

3.2 Geometric Matching for Page Frame Detection

3.2.1 Document Model

For structured documents, like technical journals and business letters, the document structure can be described as a hierarchy, where entities at each level of the hierarchy represent a particular level of information, like zones, text-lines, or connected components. Different hierarchical models representing document structure have been proposed in the past [DB89, LPH01, DSC02]. One common shortcoming of these models is that they only model the contents of the document ignoring textual and/or non-textual noise added to the document due to photocopy or scanning process. The definition of the hierarchical document model is extended in this work and another level of hierarchy is added that represents the actual page content area. The definitions of different levels of the hierarchy are as follows.

- A binary *document image* D is defined as the union of the set of the foreground pixels P_f and the background pixels P_b.
- The set of foreground pixels can then be partitioned into *connected components* $C = \{C_1, \cdots, C_M\}$ such that $C_i \cap C_j = \emptyset \;\; \forall i \neq j$ and $\bigcup_{i=1}^{M} C_i = P_f$.
- The set of *text-lines* $L = \{L_1, \cdots, L_N\}$ is viewed as a partitioning of the connected components such that $L_i \subseteq C$, $L_i \cap L_j = \emptyset \;\; \forall i \neq j$ (some connected components may not be included in any text-line).
- The set of *zones* $Z = \{Z_1, \cdots, Z_R\}$ is defined such that each zone $Z_i \subseteq C$ and $Z_i \cap Z_j = \emptyset \;\; \forall i \neq j$, where each zone consists of only one physical layout structure like text, graphics, or pictures.
- The *page frame* F is defined as the minimum rectangle containing all connected components belonging to the actual document.

Note that other levels of the hierarchy are also possible (e.g.word-level, character-level), but the above-mentioned levels are sufficient to describe a document for the purpose of page frame detection.

In order to extract the document structure at different levels of the hierarchy, the page frame detection system uses a different algorithm at each level. A fast labeling algorithm is used to extract connected components from the document image. The constrained text-line finding algorithm [Bre02c] is used to extract text-lines, whereas the Voronoi-diagram based algorithm [KSI98] is used to extract zones from the document. The text-line extraction algorithm was used with a high threshold for the quality of the extracted text-lines to avoid text-lines generated from non-textual noise components.

3.2.2 Page Frame Model

The page frame of a scanned document is parameterized as a rectangle described by five parameters $\vartheta = \{l, t, r, b, \alpha\}$. The parameters $\{l, t, r, b\}$ represent the left, top, right, and bottom coordinates respectively, whereas α represents the skew angle of the page frame. The page frame detection system takes skew corrected documents as input; standard skew correction methods [OPS99, Bre02b] can be used for this purpose. Hence, the page frame is modeled as an axis-aligned rectangle, described by four parameters $\vartheta = \{l, t, r, b\}$. Given the sets of connected components C, text-lines L, and zones Z, the goal of page frame detection is to find the maximizing set of parameters ϑ with respect to the sets C, L, and Z:

$$\hat{\vartheta}(C, L, Z) := \arg\max_{\vartheta \in T} Q(\vartheta, C, L, Z) \tag{3.1}$$

where $Q(\vartheta, C, L, Z)$ is the total quality for a given parameter set, and T is the parameter space. The design of the quality function is described in detail in Section 3.2.3, followed by the description of the algorithm for finding the optimal set of parameters in Section 3.2.4.

3.2.3 Design of Quality Function

The design of the quality function in Equation 3.1 is done by exploiting the text-alignment property of structured documents. In such documents, text-lines are usually printed in justified or left-aligned style. Hence, a large number of connected components are aligned with the page frame of the document. At first glance, it may seem like a good idea to use the number of character bounding boxes touching the page frame as the quality of the page frame. The character bounding boxes could be obtained from C by filtering out noise and non-text components based on their area and aspect ratio. However, such an approach does not work well in practice because:

1. The top and bottom text-lines do not necessarily contain more characters than other text-lines in the page (especially when there is only a page number in the header or footer). Also in some cases, there can be non-text zones (images, logos, etc.) at

the top or bottom of the page. Hence the parameters t and b can not be reliably estimated using character level information.

2. The parameters l and r can only be reliably estimated for justified text.

Therefore, instead of using connected component level information, text-lines can be used. The quality function can then be a function of the number of text-lines that touch the page frame from inside. Based on this idea, the parameters ϑ can be decomposed into two parts: $\vartheta_h = \{l, r\}$ and $\vartheta_v = \{t, b\}$. Although ϑ_h and ϑ_v are not independent, such a decomposition can still be done because of the nature of the problem. First, the parameters ϑ_v are set to their extreme values ($t = 0, b = H$ where H is the page height) and then optimal ϑ_h is searched. This setting ensures that none of the candidate text-lines is lost based on its vertical position in the image. The decomposition not only helps in reducing the dimensionality of the searched parameter space from four to two, but also prior estimates for ϑ_h make the estimation of ϑ_v a trivial task, as will be seen later in sub-section 3.2.5. Hence the optimization problem of Equation (3.1) is reduced to

$$\hat{\vartheta_h}(L) := \arg\max_{\vartheta_h \in T} Q(\vartheta_h, L) \tag{3.2}$$

The total upper bound of the quality Q can be written as the sum of local quality functions

$$Q(\vartheta_h, L) := \sum_{j=1}^{N} q(\vartheta_h, L_j) \tag{3.3}$$

An upper and lower bound for local quality function q is computed. Given a line bounding box $\bar{L} = \{x_0, y_0, x_1, y_1\}$, intervals $d(l, x_i)$ and $d(r, x_i)$ of possible distances of the x_i from the parameter intervals l and r, respectively are determined. The local quality function q for a given line and a parameter range ϑ_h can then be defined as

$$q_1(\vartheta_h, (x_0, x_1)) = \max\big(0, 1 - \frac{d^2(l, x_0)}{\epsilon^2}\big) + \max\big(0, 1 - \frac{d^2(r, x_1)}{\epsilon^2}\big) \tag{3.4}$$

Where ϵ defines the distance up to which a text-line can contribute to the page frame. Text-lines may have variations in their starting and ending positions within a text column depending on text alignment or paragraph indentation. A value of $\epsilon = 150$ pixels is used in this work in order to cope with such variations for documents scanned at 300-dpi. This quality function alone already works well for single column documents, but for multi-column documents it may report a single text-column (with the highest number of text-lines) as the optimal solution. In order to discourage such solutions, a negative weighting for text-lines on the ‘wrong’ side of the page frame is introduced in the form of the quality function

$$q_2(\vartheta_h, (x_0, x_1)) = -\max\big(0, 1 - \frac{d^2(l, x_1)}{(2\epsilon)^2}\big) - \max\big(0, 1 - \frac{d^2(r, x_0)}{(2\epsilon)^2}\big) \tag{3.5}$$

The overall local quality function is then defined as

$$q(\vartheta_h, (x_0, x_1)) = q_1(\vartheta_h, (x_0, x_1)) + q_2(\vartheta_h, (x_0, x_1)) \tag{3.6}$$

The quality function in Equation (3.6) will yield the optimal parameters for ϑ_h even if there are intermediate text-columns with larger number of text-lines. However, if the first or last column contains very few text-lines, the column can possibly be ignored. The search space for the parameters ϑ_h can be limited to certain regions of the document image to solve this problem. In this work the value of the parameter l was constrained to lie within the first half of the page, whereas the value of the parameter r was limited to the second half of the page.

3.2.4 Branch-and-Bound Optimization

The RAST (Recognition by Adaptive Subdivision of Transformation Space) technique by Breuel [Bre01] is employed to perform the maximization in Equation (3.2). RAST is a branch-and-bound algorithm that guarantees to find the globally optimal parameter set by recursively subdividing the parameter space and processing the resulting parameter hyper-rectangles in the order given by an upper bound on the total quality. During the search, each partition of the search space is described by a Cartesian product of intervals for the parameters, i.e.a set of the form $T = [l_0, l_1] \times [r_0, r_1]$. The upper bound on the quality of the page frame with parameters in the rectangular region T is calculated using interval arithmetic [Bre03d]. Given a computation of an upper bound on the quality, the search can be organized as follows (for details see [Bre01, Bre03c]):

1. Pick an initial region of parameter values T.
2. Maintain a priority queue of regions T_i, where the upper bound on the possible values of the global quality function Q for parameters $\vartheta \in T_i$ is used as the quality.
3. Remove a region T_i from the priority queue; if the upper bound of the quality

function associated with the region is too small to be of interest, terminate the algorithm.

4. If the region is small enough to satisfy the accuracy requirements for the dimensions of a region, accept it as a solution.

5. Otherwise, split the region T_i along the dimension furthest from the accuracy constraints and insert the subregions into the queue; then continue the algorithm at Step 3.

This algorithm will return the parameter set that maximizes the quality of the match function in Equation (3.2). To make the approach practical and avoid duplicate computations, a match-list representation [Bre01] is used. That is, with each region kept in the priority queue in the algorithm, a list (the match-list) of all and only those text-lines is maintained that have the possibility to contribute with a non-zero local quality to the global quality. These match-lists shrink with decreasing size of the regions T_i. It is easy to see that the upper bound of a parameter space region T_i is also an upper bound for all subsets of T_i. Hence, when a region is split in Step 5, the text-lines in the children that have already failed to contribute to the quality computation in the parent never have to be reconsidered. Thus the match-lists can be reused in the children thereby allowing a very fast computation of quality for the children.

3.2.5 Parameter Refinement

The RAST algorithm returns the optimal parameters for ϑ_h in terms of mean square error with respect to the quality function in Equation 3.3. However, if the text is not aligned in the justified style or if different paragraphs have different indentation, parameters ϑ_h returned by the RAST algorithm may cut through some text-lines as shown in Figure 3.3. So the parameters are refined to adjust the page frame according to different text alignments. If the bounding box of a text-line overlaps with the page frame by more than half of its area, the page frame parameters ϑ_h are expanded to include the complete text-line.

The use of match-lists gives the list of text-lines bounding boxes which contributed positively to the quality function $Q(\vartheta_h, L)$. All these text-lines are sorted with respect to the top of each text-line's bounding box (y_0). This gives an initial estimate for parameters ϑ_v by simply setting $t = \min(y_{0,j}), j = 1, \ldots, N$ and $b = \max(y_{1,j}), j = 1, \ldots, N$. A page

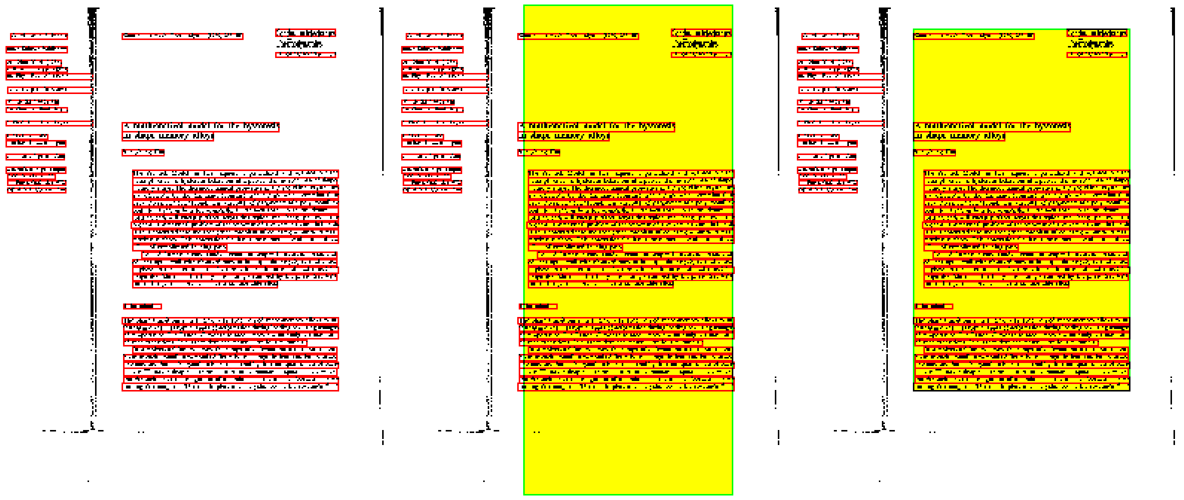


Figure 3.3: Example image demonstrating parameter refinement in order to adapt the parameters to text alignment. The detected text-lines are shown in the leftmost image. Note that some part of the text is indented more to the right as compared to other text on the page. The page frame corresponding to the optimal parameters with respect to Equation 3.6 is shown in the middle image. The image on the right side shows the initial page frame after adjusting the parameters for text alignment. (cp. [SvBKB07])

frame detected in this way is shown in Figure 3.3. Although the detected page frame is correct for most of the documents, it fails in these cases:

1. If there is a non-text zone (images, graphics, logo, . . .) at the top or bottom of the page, it is missed by the page frame.
2. If there is an isolated page number at the top or bottom of the page, and it is missed by the text-line detection, it will not be included in the detected page frame.

An example illustrating these problems is shown in Figure 3.4. In order to estimate the final values for $\vartheta_v = \{t, b\}$, document zones are used as given by the Voronoi algorithm [KSI98]. The Voronoi algorithm performs document cleaning as a part of zoning process and successfully removes most of the non-textual noise. The output of the Voronoi algorithm for an example image is shown in Figure 3.4. Textual noise usually appears only along the left or the right side of the document. Based on this observation, filtering is performed on the zones obtained by the Voronoi algorithm, such that all the zones that lie completely inside or do not overlap horizontally with the detected page frame are removed. Then, all of the remaining zones are included into the page frame. An example result is shown in Figure 3.4.

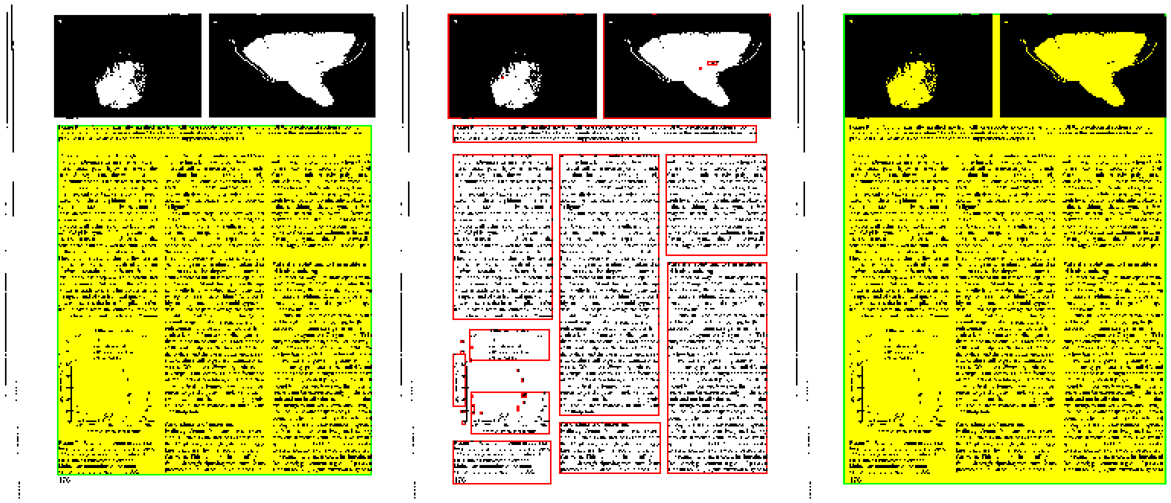


Figure 3.4: Example image demonstrating inclusion of non-text zones into the detected page frame. The initial page frame detected based only on the text-lines is shown on left. Note that the detected page frame does not include the images on the top and the page number at the bottom. The middle image shows the zones detected by the Voronoi algorithm. The rightmost image shows the final page frame obtained by using zone-level information. (cp. [SvBKB07])

3.3 Performance Measures

To determine the accuracy of the presented page frame detection algorithm, performance measures are needed that not only reflect the accuracy of the algorithm, but also quantify its usefulness in practical document analysis systems. Therefore, the error measures are categorized into two parts.

3.3.1 Page Frame Detection Accuracy

The goal of the performance measures in this section is to determine the accuracy with which the page frame is located. Previous approaches for marginal noise removal [LTW96, AL04, FWL02, CLLT02] use manual inspection to decide whether noise regions have been completely removed or not. Then, the error rate is defined as the percentage of documents on which the noise was not completely removed. While these approaches might be useful for small scale experiments, an automated way of evaluating border noise removal is needed for evaluation on a large sized dataset. In the following, performance measures based on area overlap, connected components classification, and ground-truth zone detection are introduced to evaluate different aspects of the presented page frame detection algorithm.

Area Overlap

Let F_g be the ground-truth page frame and F_d be the detected page frame. Then the area overlap between the two page frames can be defined as

$$A = \frac{2|F_g \cap F_d|}{|F_g| + |F_d|} \tag{3.7}$$

The amount of area overlap A will vary between zero and one depending on the overlap between ground-truth and detected page frames. If the two page frames do not overlap at all $A = 0$, and if the two page frames match perfectly i.e. $|F_g \cap F_d| = |F_g| = |F_d|$, then $A = 1$. This gives a good measure of how closely the two page frames match. However, the area overlap A does not give any hints about the errors made by the algorithm. Secondly, a small error like including a noise zone near the top or bottom of the page into the page frame may result in a large error in terms of area overlap. To evaluate the page frame detection algorithm in more detail, a performance measure based on connected component classification is defined.

Connected Components Classification

As mentioned in Section 3.2.1, the page frame partitions the connected components into two sets: C^p and C^n. Based on this property, and defining components detected as lying within the page frame as ‘positive’, the performance of page frame detection can be measured in terms of four quantities: ‘true positive’, ‘false positive’, ‘true negative’, and ‘false negative’. The error rate can then be defined as the ratio of incorrectly classified connected components to the total number of connected components.

The error measure based on classification of connected components gives equal importance to all components, which may not be desired. For instance, if the page number is not included in the detected page frame, the error rate will still be very low because page number comprises a very small fraction (typically about 0.03% to 0.1%) of the total number of connected components in the page frame. However, the page number carries important information for the understanding of the document. To compensate this shortcoming, a performance measure based on detection of ground-truth zones is introduced.

Ground-Truth Zone Detection

For the zone-based performance measure, three different values are determined:

- Totally In: Ground-truth zones lying completely inside the computed page frame
- Partially In: Ground-truth zones lying partially inside the computed page frame
- Totally Out: Ground-truth zones lying totally outside the computed page frame

Using this performance measure, the ‘false negative’ detections are analyzed in more detail. Since, the page numbers are considered an independent zone, missing page numbers will have a higher impact on the error rates in this performance measure.

3.3.2 Performance Gain in Practical Applications

In order to demonstrate the usefulness of page frame detection in practical applications, OCR and layout-based document image retrieval were chosen in this work.

OCR Accuracy

The OCR accuracy is determined by the percentage of characters correctly recognized in a document image. Many extra characters (false alarms) may appear in OCR output if textual noise is present in the document. Current commercial OCR systems have their own noise removal techniques to deal with marginal noise. The edit distance [Lev66] between the OCR output and the ground-truth text is used as the error measure for determining OCR accuracy. Edit distance is the minimum number of point mutations (insertion, deletions, and substitutions) required to convert a given string into a target string. The goal of performance measure based on edit distance is to determine whether the performance of existing OCR systems improves if page frame detection is used as a pre-processing step.

Layout-Based Document Image Retrieval

In layout-based retrieval, the purpose is to query document image databases by layout, in particular by measuring the similarity of different layouts in comparison to a reference or query layout. Blocks originating from marginal noise result in incorrect matches, thereby increasing the error rates of the retrieval system. Different layout analysis or page segmentation algorithms use different methods to deal with noise in a document image. The goal of this performance measure is to determine the decrease in retrieval error rates when page frame detection is used as a pre-processing step.

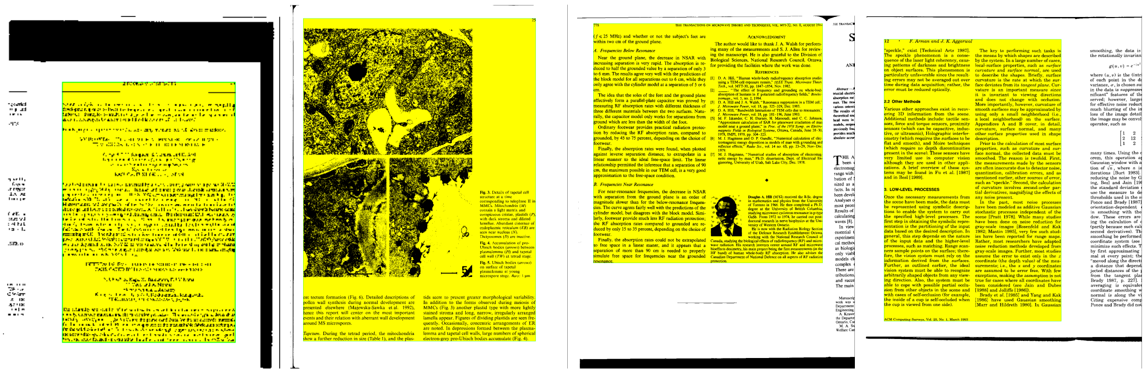
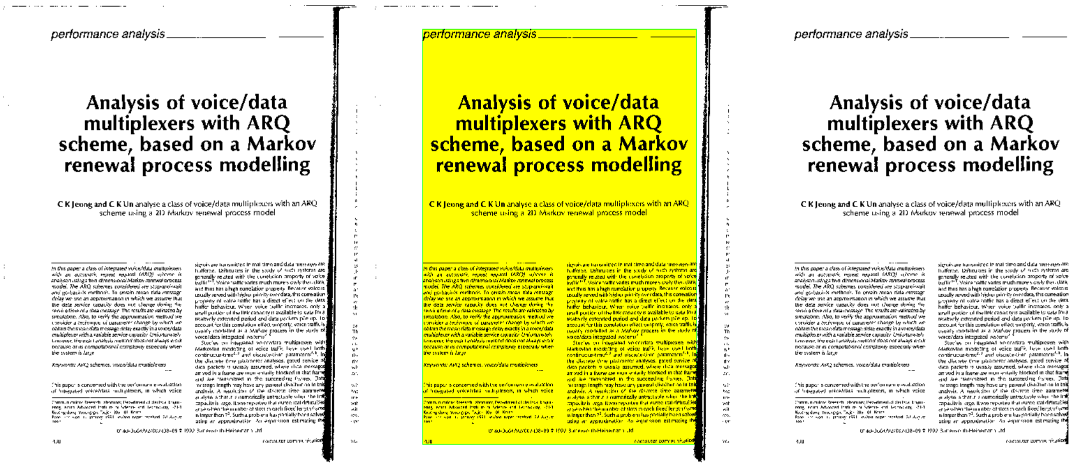


Figure 3.5: Some example images showing the detected page frame in yellow color.



performance analysis

Analysis of voice/data multiplexers with ARQ scheme, based on a Markov renewal process modelling

C K Jeong and C K Un analyse a class of voice/data multiplexers with an ARQ scheme using a 2D Markov renewal process model

performance analysis

Analysis of voice/data multiplexers with ARQ scheme, based on a Markov renewal process modelling

C K Jeong and C K Un analyse a class of voice/data multiplexers with an ARQ scheme using a 2D Markov renewal process model

performance analysis

Analysis of voice/data multiplexers with ARQ scheme, based on a Markov renewal process modelling

C K Jeong and C K Un analyse a class of voice/data multiplexers with an ARQ scheme using a 2D Markov renewal process model

Figure 3.6: An example image showing the result of page frame detection in case of border noise overlapping the page content area. Image on the left shows the original document, the middle image shows the detected page frame, and the right image shows the cleaned image removing both textual and non-textual noise outside the page content area while keeping the page content area intact.

3.4 Experiments and Results

The evaluation of the page frame detection algorithm was done on the University of Washington III (UW-III) database [Phi96]. The dataset was divided into 160 training and 1440 test images. In order to make the results replicable, every 10th image (in alphabetical order) from the dataset was included into the training set. Hence the training set consists of images A00A, A00K, . . . , W1UA. The evaluation was done on the remaining 1440 test images. Some examples of page frame detection for documents from the UW-III dataset are shown in Figure 3.5. Figure 3.6 shows an example where marginal noise overlaps with some text-lines at the bottom of the page. The use of page frame detection successfully

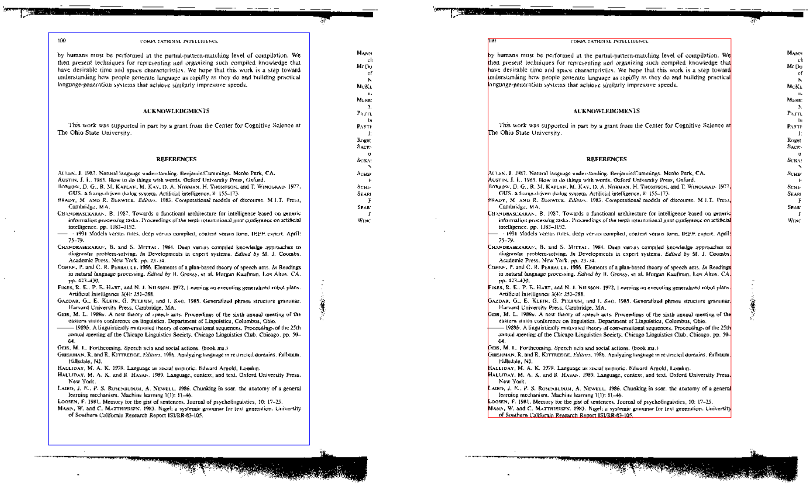


Figure 3.7: The left image shows a document together with its original ground-truth page frame. The right image shows the corrected ground-truth page frame obtained by computing the smallest rectangle including all the ground-truth zones.

detects the page contents region and removes the border noise from the image while keeping the page contents intact.

3.4.1 Page Frame Detection Accuracy

The evaluation of page frame detection on the basis of overlapping area (Equation (3.7)) showed a page frame detection accuracy of 91%. An inspection of the UW3 ground-truth page frame showed that it does not tightly enclose the page contents area as shown in Figure 3.7. Hence, the correct page frame of the documents in the test set was computed by finding the bounding box of all ground-truth zones for each document. Testing with the corrected ground-truth page frame gave an overall mean area overlap of 96%. In the following, when mentioning the ground-truth page frame, this corrected ground-truth page frame is meant.

The result for the connected component based measure is given in Table 3.1. The high percentage of true positives shows that the page frame mostly includes all the ground-truth components. The percentage of true negatives is about 73.5%, which means that a large part of noise components are successfully removed. The results for the Nth generation photocopies show that the percentage of true negatives goes down to 42.8% which may lead to the conclusion, that the computed page frames for this subset are typically bigger than the ground-truth page frame. The total error rate defined as the ratio of 'false' classifications to the total number of connected components is 1.6%. Since

Table 3.1: Results for the connected component based evaluation. The number in brackets gives the number of documents of that class. Error rates in [%].

Document Type	True Positive	False Negative	True Negative	False Positive
Scans (392)	99.84	0.16	76.6	23.4
1Gen (1029)	99.78	0.22	74.0	26.0
NGen (19)	99.93	0.07	42.8	57.2
all (1440)	99.8	0.2	73.5	26.5
total (absolute)	4,399,718	8,753	187,446	67,605

Table 3.2: Results for the zone based performance evaluation. Error rates in [%].

Document Type	Totally In	Partially In	Totally Out
Scans (392)	97.6	0.7	1.7
1Gen (1029)	97.1	1.0	1.9
NGen (19)	97.5	0.0	2.5
all (1440)	97.2	0.9	1.9

the test set contains only 19 images in the NGen category, the total results do not reflect the performance on such severely degraded documents. A detail study of the performance of the proposed page frame detection method on documents with different noise levels is presented later in this section.

The results for the zone based measure are given in Table 3.2. Compared to the number of missed connected components, it can be seen that the percentage of missed zones is slightly higher than the corresponding percentage of false negatives on the connected component level. One conclusion that can be drawn from this observation is that the zones missed do not contain a large number of components, which is typically true for page numbers, headers and footers of documents. These zones have a few components and therefore do not contribute much to the mean false negative errors on the connected component level. In some cases, the text-line finding algorithm merges the text-lines consisting of textual noise to those in the page frame. In such cases, a large portion of textual noise is also included in the page frame.

In order to quantify the amount of marginal noise in a document image, the noise ratio of a document image is defined as

$$\text{Noise ratio} = \frac{n_{pb}}{n_p} \tag{3.8}$$

Where n_{pb} is the number of foreground pixels outside the ground-truth page frame, and

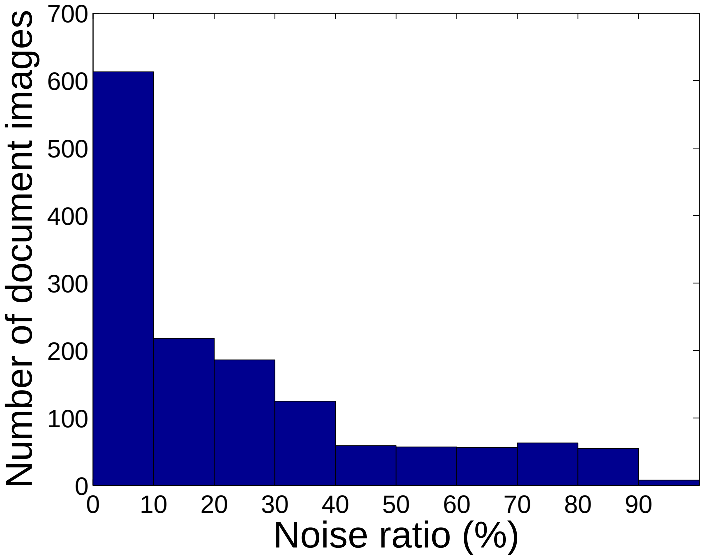


Figure 3.8: Histogram of the noise level of the documents in the test set.

n_p is the total number of foreground pixels in a document image. A histogram of the noise level of the documents in the test set is shown in Figure 3.8. Interestingly, there are many documents with noise levels above 50%. The mean error rate obtained for each of these noise level based document categories is plotted in Figure 3.9. The plot shows that the algorithm works well even on documents with very high amount of noise. The error rates on all three performance measures used are below 10% for noise levels up to 80%.

Some limitations of the presented page frame detection algorithm were also revealed during the course of evaluation. Although the algorithm works very well for most of the layouts even under large amount of noise, yet for a few layouts the algorithm does not give 100% result even for noise-free documents. This happens for documents with very few text-lines beside the margin of the document and there is no text-line that spans across the main content area and the page margin. In this case, these text-lines lie completely outside the computed parameters ϑ_h (Equation 3.2). So the parameter refinement step (Section 3.2.5) fails to include these text-lines into the page frame. To deal with such layouts, the quality function can be modified to include an offset between the page frame parameters and the main content area of the page.

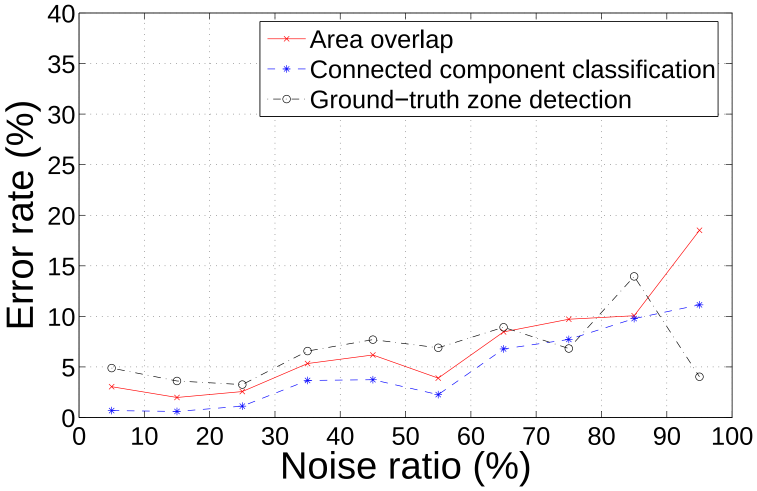


Figure 3.9: Performance of the page frame detection on different documents categorized by their noise level.

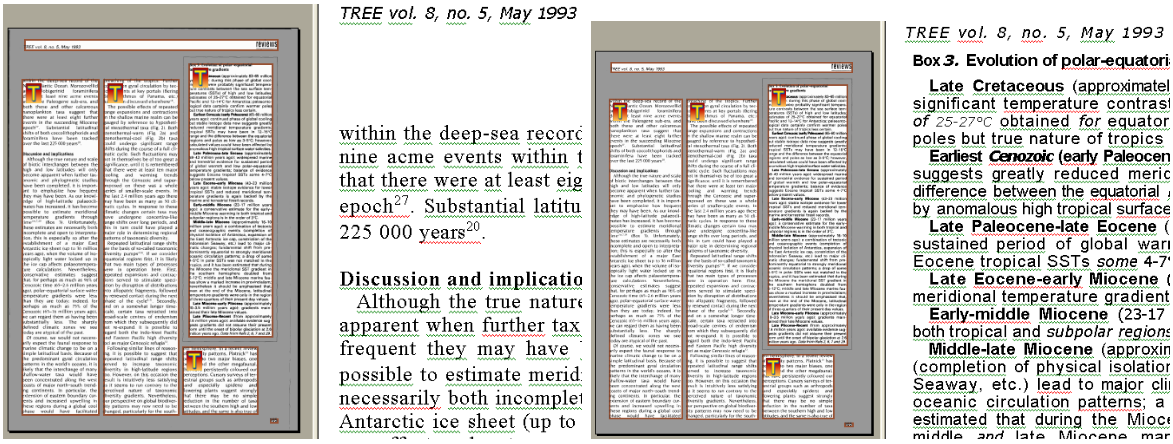
3.4.2 Performance Gain in Practical Applications

The use of page frame detection in an OCR system showed significant improvement in the OCR results. For this purpose Omnipage 14 - a commercial OCR system - was chosen. The ground-truth text provided with the UW-III dataset has several limitations when used to evaluate an OCR system. First, there is no text given for tables. Secondly, the formating of the documents is coded as latex commands. When an OCR system is tested on this ground-truth using error measures like the Edit distance, the error rate is unjustly too high. Also, our emphasis in this work is on the improvement of OCR errors by using page frame detection, and not on the actual errors made by the OCR system. Hence, the UW-III documents are first cleaned using the ground-truth page frame, an then the output of Omnipage on the cleaned images was used as the ground-truth text. This type of ground-truth gives us an upper limit of the performance of a page frame detection algorithm, and if the algorithm works perfectly, it should give 0% error rate, independent of the actual error rate of the OCR engine itself.

First, OCR was performed on the original images and the Edit distance to the estimated ground-truth text was computed. Then, the computed page frame was used to remove marginal noise from the documents, and the experiments was run again. The results (Table 3.3) show that the use of page frame detection for marginal noise removal reduced the OCR error rate from 4.3% to 1.7%. The insertion errors are reduced by a factor of 2.6, which is a clear indication that the page frame detection helped in removing a lot of extra text that were treated previously as part of the document text. There are

Table 3.3: Results for the OCR based evaluation with page frame detection (PFD) and without page frame detection.

	Total Characters	Deletions	Substitutions	Insertions	Total Errors	Error Rate
Without PFD	4831618	34966	29756	140700	205422	4.3%
With PFD	4831618	19544	9828	53610	82982	1.7%



TREE vol. 8, no. 5, May 1993

within the deep-sea recor
nine acme events within
that there were at least eig
epoch^{27}. Substantial latitu
225 000 years^{20}.

Discussion and implicatio
Although the true natur
apparent when further tax
frequent they may have
possible to estimate merid
necessarily both incomple
Antarctic ice sheet (up to

TREE vol. 8, no. 5, May 1993

Box *3.* Evolution of polar-equator
Late Cretaceous (approximate
significant temperature contras
of 25-27°C obtained *for* equator
poles but true nature of tropics
Earliest *Cenozoic* (early Paleocen
suggests greatly reduced meri
difference between the equatorial
by anomalous high tropical surface
Late Paleocene-late Eocene (
sustained period of global war
Eocene tropical SSTs *some* 4-7
Late Eocene-early Miocene
meridional temperature gradient
Early-middle Miocene (23-17
both tropical and *subpolar region*
Middle-late Miocene (approxi
(completion of physical isolatio
Seaway, etc.) lead to major cli
oceanic circulation patterns; a
estimated that during the Mioc
middle *and* late Miocene mar

Figure 3.10: Screenshot of Omnipage 14 showing the recognized text of the original document (left) and the document cleaned using page frame detection (right). Note that the reading order of the text has changed, probably due to the slightly changed geometry. (cp. [SvBKB07])

also some deletion errors, which are a result of the changes in the OCR software's reading order determination. One example is shown in Figure 3.10 for which the reading order changed after document cleaning.

The effect of using page frame detection on the performance of a layout based document image retrieval application showed a significant decrease in retrieval error rates. van Beusekom et al. [vBKSB06] introduced several similarity measures for layout based document image retrieval. They evaluated the performance of these similarity measures on the MARG database [FT03]. The experiments showed that the best distance measure for this task is the overlapping area combined with the Manhattan distance of the corner points as block distance together with the minimum weight edge cover matching for establishing correspondences between the matched layouts. The documents in the MARG database are categorized with respect to 9 different layout types, 59 publishers, and 161 journals. Given a query document, the target is to retrieve a document of the same class based on layout information only. The error rates are then determined as the percentage

Table 3.4: Comparison of the error rates [%] for layout-based document image retrieval with and without using page frame detection.

Segmentation algorithm	Page frame detection	MARG database classes		
		journal	type	publisher
Voronoi	no	31.0	7.5	7.0
	yes	29.7	5.3	5.4
X-Y cut	no	36.3	11.7	13.6
	yes	33.5	8.6	8.0
Docstrum	no	40.9	14.0	14.4
	yes	32.1	7.4	7.1
Whitespace	no	48.3	20.3	24.6
	yes	31.2	7.2	6.1

of correctly retrieved documents using leave-one-out cross validation.

In this work retrieval experiments were performed both with and without using page frame detection. Since each method for page segmentation has a different way of dealing with noise, four well-known page segmentation algorithms were compared for use in layout based retrieval: the X-Y cut [NSV92], Docstrum [O'G93], whitespace analysis [Bai94], and the Voronoi-diagram based approach [KSI98].

In the first experiment, the document images were used directly for page segmentation without any page frame detection. The blocks extracted from the documents were then used for the purpose of layout based retrieval.

In the second experiment, the document images in the database were cleaned by performing page frame detection and removing all the foreground pixels outside the detected page frame. Following the document cleaning, page segments were extracted from the cleaned images and then retrieval experiment was repeated. The decrease in error rates for each of the three subdivisions of the data set (according to type, publisher, and journal) was used as a performance measure.

The use of page frame for layout-based document image retrieval resulted in lower error rates on all three classes of layouts for each algorithm as shown in Table 3.4. These results show that the Voronoi-diagram based approach performs better than other algorithms both with and without page frame detection. The use of page frame detection with the Voronoi algorithm lowers the retrieval error rates by 4% for the correct journal, 30% for the correct type, and 20% for the correct publisher. These results clearly demonstrate the usefulness of page frame detection in practical applications.

3.5 Summary

This chapter presented an algorithm for page frame detection using a geometric matching method. The presented approach does not assume the existence of whitespace between marginal noise and the page frame, and can detect the page frame even if the noise overlaps some regions of the page content area. Several error measures were defined based on area overlap, connected component classification, and ground-truth zone detection accuracy for determining the accuracy of the presented page frame detection algorithm. It was shown that the algorithm performs well on all three performance measures with error rates below 4% in each case. It was also demonstrated that the presented method can handle documents with a very large amount of noise with reasonable accuracy. The error rates on all three performance measures used are below 10% for noise levels up to 80%. The major source of errors was missing isolated page numbers. Locating the page numbers as a separate process and including them in the detected page frame may further decrease the error rates. The benefits of the page frame detection in practical applications were highlighted by using it with an OCR system and a layout-based document image retrieval system, where it showed a significant decrease in the error rates in both applications.

Chapter 4

Page Segmentation

The task of page segmentation is to divide a document image into homogeneous zones, each consisting of only one physical layout structure (text, graphics, pictures, . . .). 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 [CCMM98, Nag00]). This chapter presents the following contributions to the state-of-the-art in page segmentation[1]:

1. Performance evaluation and comparison of six well-known algorithms for page segmentation using a state-of-the-art evaluation methodology [MK01].

2. Identification of a severe flaw in the state-of-the-art evaluation scheme when used for single column documents.

3. A novel, portable, and pixel-accurate representation for arbitrarily shaped page segments.

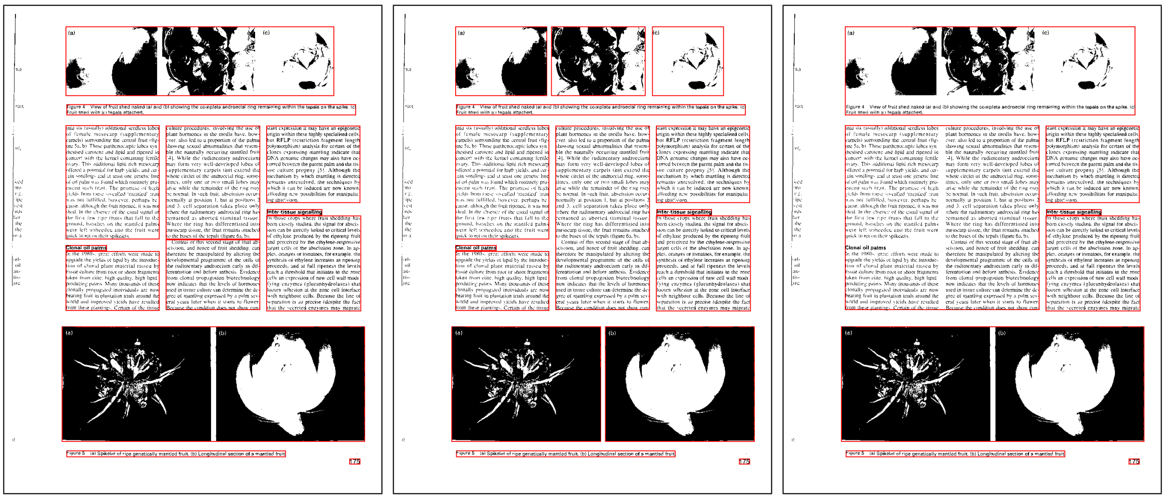
4. A vectorial score to identify and analyze different common classes of segmentation errors made by a page segmentation algorithm.

5. Identification of several specific flaws in widely used page segmentation algorithms.

6. A novel combination of different approaches for page segmentation that results in the lowest mean error rate on the UW-III dataset.

---

[1] This chapter is based on the author's work in [SKB06a], [SKB06b], and [SKB08b].



(a) Segmentation A (b) Segmentation B (c) Segmentation C

Figure 4.1: An example image showing different segmentations of the same document. Segmentations A and B are both correct as they separate text in different columns as well as images from text. Segmentation C is considered to have two errors: 1) text in the first two columns is merged 2) Caption of the top figure is merged with the figure itself.

4.1 Introduction

Page segmentation is a key component of geometric layout analysis. Given a document image, the goal of page segmentation is to perform a decomposition of the document image into smaller zones or segments. The segments thus obtained are classified as containing text or non-text elements. The text segments or zones are then fed to a character recognition module to convert them into electronic format. If a page segmentation algorithm fails to correctly segment text from images, the character recognition module outputs a lot of garbage characters originating from the image parts. Additionally, if the document contains more than one text-column, the page segmentation algorithm should segment all text-columns separately so that the text-lines in different text-columns are not merged together.

Some possibilities of segmenting a document image are shown in Figure 4.1. Note that there are more than one possible ways for correctly segmenting a document image. Figure 4.1(c) shows some common segmentation errors that adversely effect OCR accuracy. The OCR output of the in-correctly segmented zones in Figure 4.1(c) is shown in Figures 4.2 and 4.3. Figure 4.2 shows the case of an image merged with a text segment. The OCR system outputs a large number of garbage characters in an attempt to classify

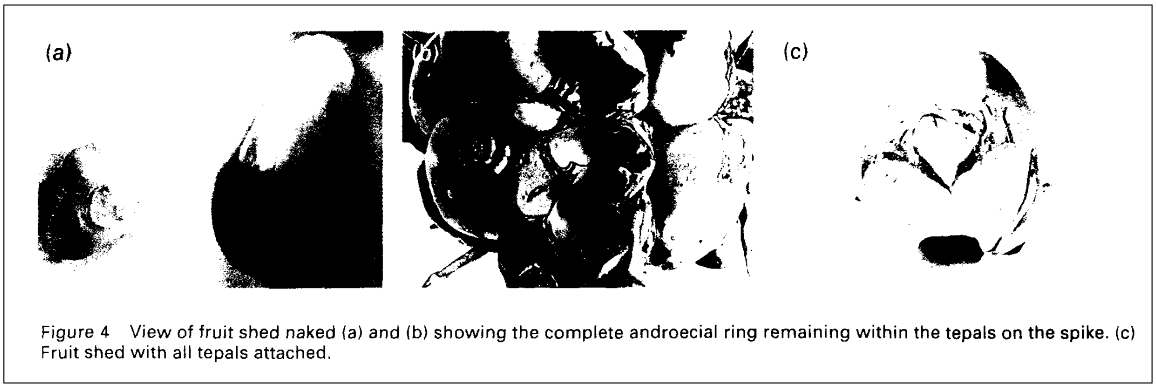
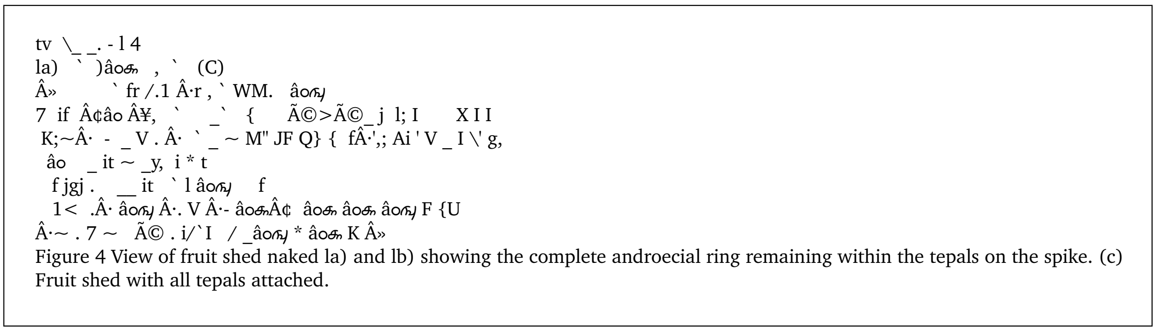


Figure 4 View of fruit shed naked (a) and (b) showing the complete androecial ring remaining within the tepals on the spike. (c) Fruit shed with all tepals attached.

(a) Input page segment



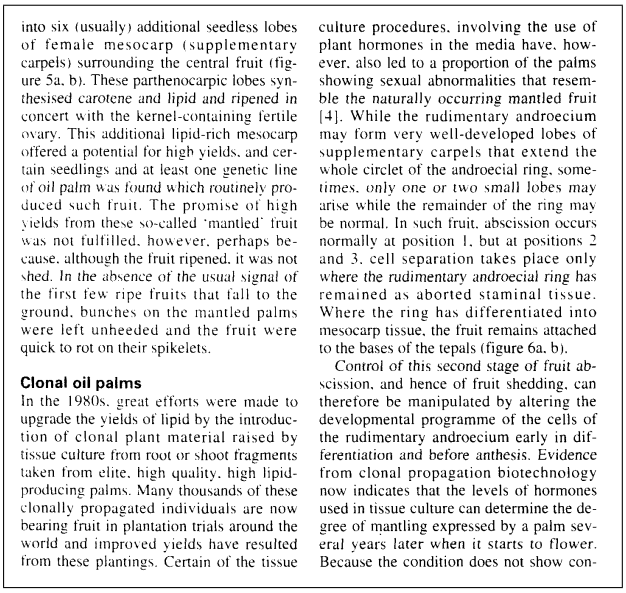
```
tv \_ _. - l 4
la)  `  )â€œ  ,  `  (C)
Â»        ` fr /.1 Â·r , ` WM.  â€™
7  if  Â¢âo Â¥,  `     _`  {     Ã©>Ã©_j  l; I      X I I
 K;~Â·  -  _V . Â·  ` _ ~ M" JF Q} {  fÂ·',; Ai ' V _ I \' g,
  âo    _ it ~ _y,  i * t
   f jgj .    __ it   ` l â€™    f
   1<  .Â· â€™ Â·. V Â·- â€œÂ¢  â€œ â€œ â€™ F {U
Â·~ . 7 ~   Ã© . i/`I  / _â€™ * â€œ K Â»
Figure 4 View of fruit shed naked la) and lb) showing the complete androecial ring remaining within the tepals on the spike. (c)
Fruit shed with all tepals attached.
```

(b) OCR result

Figure 4.2: The OCR result of an in-correctly segmented zone containing both images and text from Figure 4.1(c). The OCR system generates many garbage symbols from the non-text parts of the input page segment.

the image portions as text. Figure 4.3 demonstrates the case in which two text-columns were reported as one segment. The OCR system merged the text from the two columns and completely destroyed the reading order of the document.

The importance of the page segmentation step in geometric layout analysis has triggered a lot of interest in the scientific community over the last three decades and several algorithms for page segmentation have been proposed. Cattoni et al [CCMM98] and Nagy [Nag00] have provided a comprehensive overview of the state-of-the-art in page segmentation. Some of these algorithms have come to wide spread use for analyzing documents in different scripts and languages. In this work, six representative algorithms for page segmentation (Section 4.2) are chosen for a comparative evaluation of their performance (Section 4.3). Then a new algorithm for page segmentation is proposed (Section 4.4) that is based on a novel combination of geometric model of column-separator by Breuel [Bre02c] with the whitespace analysis approach of Baird [Bai94].



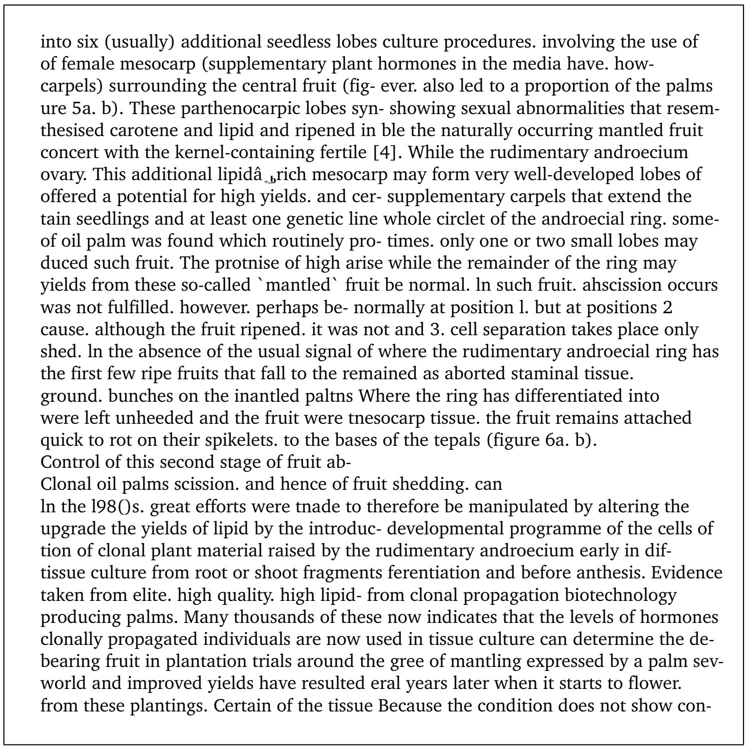
into six (usually) additional seedless lobes of female mesocarp (supplementary carpels) surrounding the central fruit (figure 5a, b). These parthenocarpic lobes synthesised carotene and lipid and ripened in concert with the kernel-containing fertile ovary. This additional lipid-rich mesocarp offered a potential for high yields, and certain seedlings and at least one genetic line of oil palm was found which routinely produced such fruit. The promise of high yields from these so-called 'mantled' fruit was not fulfilled, however, perhaps because, although the fruit ripened, it was not shed. In the absence of the usual signal of the first few ripe fruits that fall to the ground, bunches on the mantled palms were left unheeded and the fruit were quick to rot on their spikelets.

Clonal oil palms

In the 1980s, great efforts were made to upgrade the yields of lipid by the introduction of clonal plant material raised by tissue culture from root or shoot fragments taken from elite, high quality, high lipid-producing palms. Many thousands of these clonally propagated individuals are now bearing fruit in plantation trials around the world and improved yields have resulted from these plantings. Certain of the tissue culture procedures, involving the use of plant hormones in the media have, however, also led to a proportion of the palms showing sexual abnormalities that resemble the naturally occurring mantled fruit [4]. While the rudimentary androecium may form very well-developed lobes of supplementary carpels that extend the whole circlet of the androecial ring, sometimes, only one or two small lobes may arise while the remainder of the ring may be normal. In such fruit, abscission occurs normally at position 1, but at positions 2 and 3, cell separation takes place only where the rudimentary androecial ring has remained as aborted staminal tissue. Where the ring has differentiated into mesocarp tissue, the fruit remains attached to the bases of the tepals (figure 6a, b).

Control of this second stage of fruit abscission, and hence of fruit shedding, can therefore be manipulated by altering the developmental programme of the cells of the rudimentary androecium early in differentiation and before anthesis. Evidence from clonal propagation biotechnology now indicates that the levels of hormones used in tissue culture can determine the degree of mantling expressed by a palm several years later when it starts to flower. Because the condition does not show con-

(a) Input page segment



```
into six (usually) additional seedless lobes culture procedures. involving the use of
of female mesocarp (supplementary plant hormones in the media have. how-
carpels) surrounding the central fruit (fig- ever. also led to a proportion of the palms
ure 5a. b). These parthenocarpic lobes syn- showing sexual abnormalities that resem-
thesised carotene and lipid and ripened in ble the naturally occurring mantled fruit
concert with the kernel-containing fertile [4]. While the rudimentary androecium
ovary. This additional lipidâ¸rich mesocarp may form very well-developed lobes of
offered a potential for high yields. and cer- supplementary carpels that extend the
tain seedlings and at least one genetic line whole circlet of the androecial ring. some-
of oil palm was found which routinely pro- times. only one or two small lobes may
duced such fruit. The protnise of high arise while the remainder of the ring may
yields from these so-called `mantled` fruit be normal. ln such fruit. ahscission occurs
was not fulfilled. however. perhaps be- normally at position l. but at positions 2
cause. although the fruit ripened. it was not and 3. cell separation takes place only
shed. ln the absence of the usual signal of where the rudimentary androecial ring has
the first few ripe fruits that fall to the remained as aborted staminal tissue.
ground. bunches on the inantled paltns Where the ring has differentiated into
were left unheeded and the fruit were tnesocarp tissue. the fruit remains attached
quick to rot on their spikelets. to the bases of the tepals (figure 6a. b).
Control of this second stage of fruit ab-
Clonal oil palms scission. and hence of fruit shedding. can
ln the l98()s. great efforts were tnade to therefore be manipulated by altering the
upgrade the yields of lipid by the introduc- developmental programme of the cells of
tion of clonal plant material raised by the rudimentary androecium early in dif-
tissue culture from root or shoot fragments ferentiation and before anthesis. Evidence
taken from elite. high quality. high lipid- from clonal propagation biotechnology
producing palms. Many thousands of these now indicates that the levels of hormones
clonally propagated individuals are now used in tissue culture can determine the de-
bearing fruit in plantation trials around the gree of mantling expressed by a palm sev-
world and improved yields have resulted eral years later when it starts to flower.
from these plantings. Certain of the tissue Because the condition does not show con-
```

(b) OCR result

Figure 4.3: The OCR result of an in-correctly segmented zone containing multiple text-columns from Figure 4.1(c). Note that text from the two columns in merged in the OCR output such that the reading order is completely wrong.

4.2 Page Segmentation Algorithms

Following algorithms have been selected as representative state-of-the-art page segmentation algorithms due to their wide-spread use in the document analysis community:

1. X-Y Cut by Nagy et al. [NSV92]
2. Smearing by Wong et al. [WCW82]
3. Whitespace Analysis by H. Baird [Bai94]
4. Constrained Text-Line Extraction by T. Breuel [Bre02c]
5. Docstrum by O'Gorman [O'G93]
6. Voronoi by K. Kise [KSI98]

These algorithms can be categorized into two broad classes, namely zone-based algorithm and line-based algorithm. The zone-based algorithms try to extract text zones from a document image that further need to be fragmented into text-lines for OCR. A text block can usually be easily segmented into text-lines using horizontal projection. Examples of such algorithms are the x-y cut, whitespace analysis, docstrum, and Voronoi algorithms. The line-based algorithms, on the other hand, directly extract text-lines from the input document that can be fed to an OCR system. This class of algorithms include the smearing and constrained text-line finding algorithms. A brief description of each algorithm and its parameters is given in turn in the following.

4.2.1 X-Y Cut

The x-y cut segmentation algorithm [NSV92], 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.

4.2.2 Smearing

The run-length smearing algorithm (RLSA) [WCW82] 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

$$R < f_{tr} R_m \qquad \text{and} \qquad H < f_{th} H_m \tag{4.1}$$

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.

4.2.3 Whitespace Analysis

The whitespace analysis algorithm described by Baird [Bai94] 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 [Bre02c] is used in this work for this step. These covers are then sorted with respect to the sort key, $K(c)$:

$$K(c) = \sqrt{\text{area}(c) * W(|\log_2(\text{height}(c)/\text{width}(c))|)} \tag{4.2}$$

where c is the cover and $W(.)$ is a dimensionless weighting function. Baird [Bai94] chose a special weighting function using experiments on a particular dataset. The following approximation of the original weighting function was used in this work:

$$W(x) = \begin{cases} 0.5 & \text{if } x < 3 \\ 1.5 & \text{if } 3 \leq x < 5 \\ 1 & \text{if } x \geq 5 \end{cases} \tag{4.3}$$

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 (4.4) is satisfied:

$$K(s_j) - f_w * j/m \leq t_s \tag{4.4}$$

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, bounding boxes of these uncovered regions are taken as representative of the text segments.

4.2.4 Constrained Text-Line Detection

The layout analysis approach by Breuel [Bre02c] 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 [Bre02c]. 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 [Bre02b]. Then, the bounding box of all the characters making the text-line is computed.

4.2.5 Docstrum

The docstrum algorithm proposed by O'Gorman [O'G93] 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}.

4.2.6 Voronoi-Diagram Based Algorithm

The Voronoi-diagram based segmentation algorithm by Kise et al. [KIDM98,KSI98] 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. Each Voronoi region is then represented by its bounding box.

4.3 Performance Evaluation

Several page segmentation algorithms have been proposed over the last decades, some of which were described briefly in Section 4.2. However, it is hard to predict how well a page segmentation algorithm will perform at a particular task. This problem arises due to a lack of comparative evaluation of page segmentation algorithms. The results of the algorithms as published by their authors can not be directly compared due to the lack of a common dataset, a wide diversity of objectives, and a lack of meaningful quantitative evaluation schemes. Hence, the benchmarking of different page segmentation algorithms is becoming an important issue. Recent page segmentation competitions [AGK03,AGB05,AGB07] aimed at addressing the need of comparative performance evaluation under realistic circumstances. However, each participant in these contests participated with his own methods. Unfortunately, none of the widely used page segmentation algorithms were presented for participation in the contests.

This work presents a comprehensive performance evaluation of well-known page segmentation algorithms and compares their performance on common grounds. The UW-III dataset is used for performance evaluation and benchmarking of the analyzed algorithms with OCR as the objective application. First, a pixel-accurate representation of page segmentation is introduced in Section 4.3.1. Then, some techniques are discussed to generate pixel-level ground-truth for benchmarking purposes in a quick and efficient manner in Section 4.3.2. An overview of the state-of-the-art in evaluation methods for page segmentation is given in Section 4.3.3 followed by the description of a vectorial score based on pixel-level page segmentation information. Finally, a thorough evaluation of the six page segmentation algorithms described in Section 4.2 is presented in Section 4.3.5 based on a state-of-the-art evaluation score as well as the proposed vectorial score.

4.3.1 Representation of Page Segments

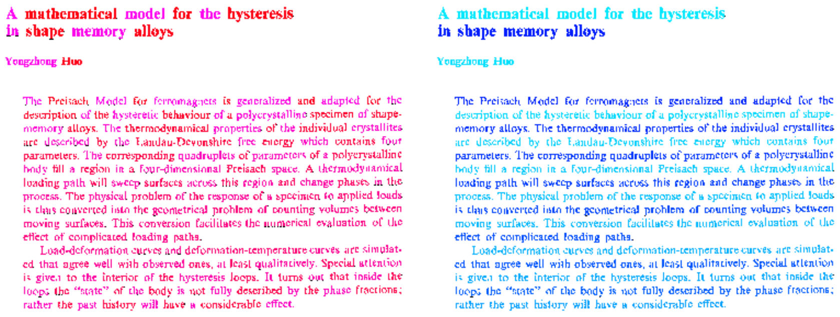
Layouts of a document image are generally categorized into two main classes: *Manhattan* layouts, and *non-Manhattan* layouts [CCMM98]. Manhattan layouts are defined as layouts that can be decomposed into individual segments by vertical and horizontal cuts. For Manhattan layouts, the individual zones 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 [DDS$^+$97] 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 [FT03] 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 [AGK03,AGB05]. 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, a new way of representing page segments in color image format is proposed in this work. Consider a document image decomposed into N homogeneous zones $Z_i, i = 1, \ldots, N$. 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 4.4. Secondly, it is independent of the zone shape and it can be saved and exchanged using any lossless color image format.

4.3.2 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, two alternatives are considered for preparing pixel-level ground-truth.

1. A bounding polygon is drawn for each zone in the page image. The polygon is



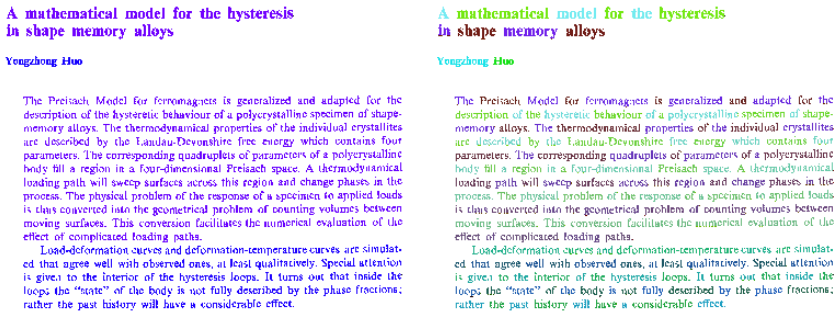
A mathematical model for the hysteresis
in shape memory alloys

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



A mathematical model for the hysteresis
in shape memory alloys

Yongzhong Huo

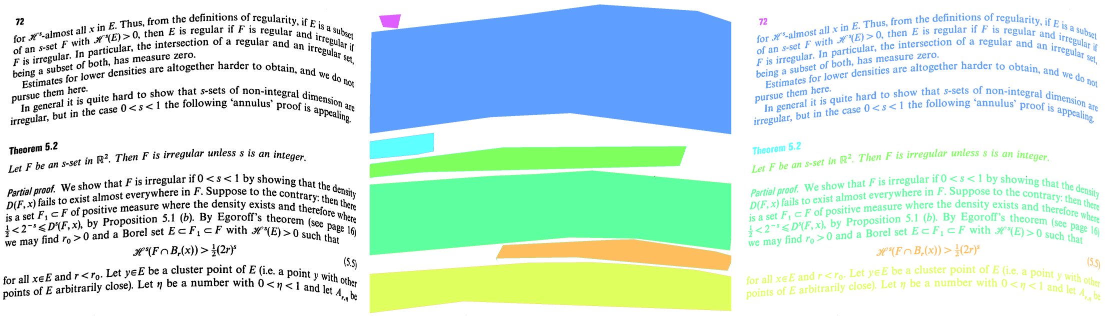
The Preisach Model for ferromagnets is generalized and adapted for the description of the hysteretic behaviour of a polycrystalline specimen of shape-memory alloys. The thermodynamical properties of the individual crystallites are described by the Landau-Devonshire free energy which contains four parameters. 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.

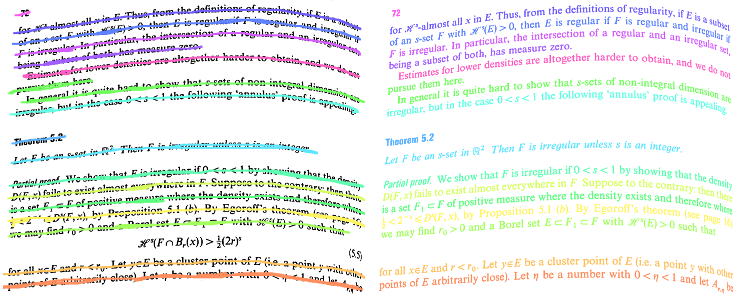
(c) Zone level (d) Multiple layout levels

Figure 4.4: An example image demonstrating 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. (cp. [SKB08b])

filled with a color representing the index of the zone contained inside the polygon (Figure 4.5(b)). Then each foreground pixel is assigned the color of the polygon that contains it (Figure 4.5(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.



(a) Original Image (b) Labeled Zones (c) Generated zone level ground-truth



(d) Labeled text-lines (e) Generated text-line ground-truth

Figure 4.5: An example image demonstrating the process of generating pixel-level ground-truth. The zone-level ground-truth is prepared by first drawing a polygon around each zone (Fig. 4.5(b)) and then transferring the colors to foreground pixels in the zone (Fig. 4.5(c)). The text-line level ground-truth is created by drawing lines (Fig. 4.5(d)) and then labeling connected components touching these lines with the line color (Fig. 4.5(e)). (cp. [SKB08b])

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 4.5(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 4.5(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 the x-height in this 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 [SB07] held with CBDAR 2007.

4.3.3 Evaluation Methods for Page Segmentation

The problem of automatic evaluation of page segmentation algorithms is increasingly becoming an important issue [LHA$^+$04, AKB06]. An approach for measuring the quality of page segmentation algorithms by analyzing the errors in the text recognized by OCR was first proposed in [KNRN93]. Agne et al. [ARR00, ADK03] proposed a benchmarking system for document segmentation based on detecting page segmentation errors by analyzing OCR errors. 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. [YV95] 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. Thulke et al. [TD98, TMD99] presented a quality evaluation scheme for page segmentation based using precision and recall concepts from the information retrieval literature. An advantage of this method is that it can be used to detect segmentation errors at different layout levels. A practical limitation of their approach is that they require the words in the ground-truth segmentation be exactly the same as the words in the segmented output - which is difficult to achieve.

Liang et al. [LPH01] proposed a performance metric for document structure extraction algorithms by finding the correspondences between detected entities and ground-truth. Das et al. [DSC02] 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 [HJBJ$^+$96], and for image segmentation in general [JMIB06]. Mao et al. [MK01] 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. Therefore, this measure was chosen as a representative state-of-the-art methodology for evaluation the performance of page segmentation algorithms.

The performance evaluation measure proposed in [MK01] 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:

$$\rho = \frac{|C \cup S \cup M|}{|G|} \tag{4.5}$$

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 [MK01]. To check the severity of the problem, a dummy segmentation algorithm that returns the whole page as one segment is added into the comparison.

This limitation is overcome is this work by defining a vectorial score (Section 4.3.4) that clearly identifies the common classes of segmentation errors including the under-segmentation problem identified above.

4.3.4 Vectorial Score for Performance Evaluation

Based on the pixel-accurate representation of page segmentation as described in Section 4.3.1, several performance measures are defined to evaluate different aspects of the behavior of a page segmentation algorithm. Consider two segmentations of a document 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, a weighted bipartite graph called *pixel-correspondence graph* [Bre02a] can be constructed as follows. Each color value in H or G is associated with 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 4.3.1) 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, $t_r = 0.1$ is found 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

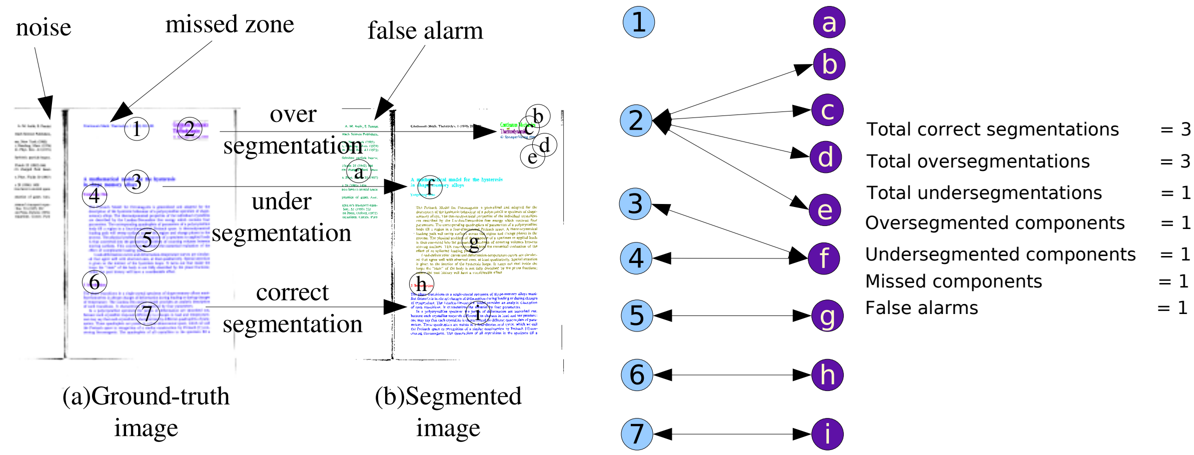


Figure 4.6: Example image illustrating different performance measures. The left image shows two color coded document images. A pixel correspondence graph obtained from these images is shown on the right side. The nodes corresponding to the ground-truth segments are labeled 1-7, whereas the nodes in the segmented image are labeled a-i. Only significant edges are shown in the pixel correspondence graph. Based on the definitions given in Section 4.3.4, the values of each performance measure for this example are given on the right side of the graph. (cp. [SKB08b])

desired geometric accuracy of the evaluation results. For the UW-III document images, $t_a = 500$ pixels was used for zone-level evaluation and $t_a = 100$ pixels was used 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 *under-segmented*, respectively. Using these definitions, several measures for evaluating a page segmentation algorithm are introduced in this work. An illustration of these measures is given in Figure 4.6. 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.

4.3.5 Experiments and Results

The experiments have been divided into two parts:

A. *Benchmarking* of the algorithms based on text-line based measure of block segmentation accuracy given by Equation 4.5.

B. *Performance evaluation* of the algorithms based on the vectorial score defined in Section 4.3.4.

The first experiment augments the work of Mao et al. [MK01] and adds three more algorithms to the comparison. A detailed analysis of the errors is done to show that the limitation of the algorithm as pointed out in Section 4.3.3 is reflected in the results. It is also shown that due to this limitation, the evaluation score gives completely misleading results in certain cases. The second experiments demonstrates the benefits of the proposed vectorial score based evaluation method as compared to the single-score based measure.

Benchmarking

The benchmarking of the page segmentation algorithms was done on a subset of the UW-III database. The 978 images that correspond to the UW-I dataset pages were chosen as in [MK01]. 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 4.1). 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 [NSV92]. The optimized parameter values used for

Table 4.1: Parameter values used for each algorithm in the evaluation given in Table 4.2. 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	$t_{sh} = 300, t_{sv} = 500, t_{sm} = 30, f_{tr} = 3, f_{th} = 3$	
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 4.2: Mean text-line detection error rates (Eq. 4.5) expressed as percentage for different page segmentation algorithms on 100 train images and 878 test images.

	Default parameters		Optimized parameters	
Algorithm	Train ($n = 100$)	Test ($n = 878$)	Train ($n = 100$)	Test ($n = 878$)
Dummy	52.2	48.8	52.2	48.8
X-Y cut	14.7	17.1	14.7	17.1
Smearing	13.4	14.2	13.4	14.2
Whitespace	12.7	12.2	9.1	9.8
Text-line	8.9	8.5	8.9	8.5
Docstrum	8.7	11.2	4.3	6.0
Voronoi	6.8	7.5	4.7	5.5

x-y cut, docstrum, and Voronoi-diagram based algorithms were the same as in [MK01]. For the smearing, whitespace, and constrained text-line finding algorithms, experiments were done with different parameter values and those which gave lowest error rates on the training set were selected.

The page segmentation evaluation toolkit (PSET) [MK02] that implements the training and evaluation scheme by [MK01] was used in this work. The average text-line detection error rate for each algorithm is given in Table 4.2. 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.

Table 4.3 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

Table 4.3: Text-line detection errors [%] for each page segmentation algorithm 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

three-column documents in the test set of 878 pages, respectively. It can be observed 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. This hypothesis was further investigated 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 4.3. It can be observed 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.

Figure 4.7 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. The following details can be observed from the box plot: 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

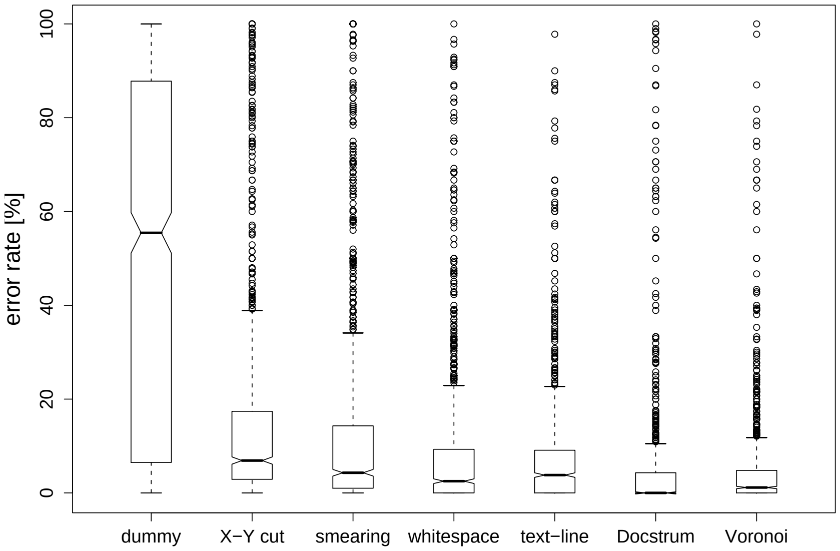


Figure 4.7: A box plot of the results obtained with optimized parameters on the test data. The plot shows that the Docstrum algorithm has the lowest median error, but it fails badly on a larger number of documents than the Voronoi algorithm. (cp. [SKB08b])

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, the correlation of the errors made by each algorithm is plotted in Figure 4.8. 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 4.5 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 4.9. Since there are only a few images in the test set that fall into this category, the experimental

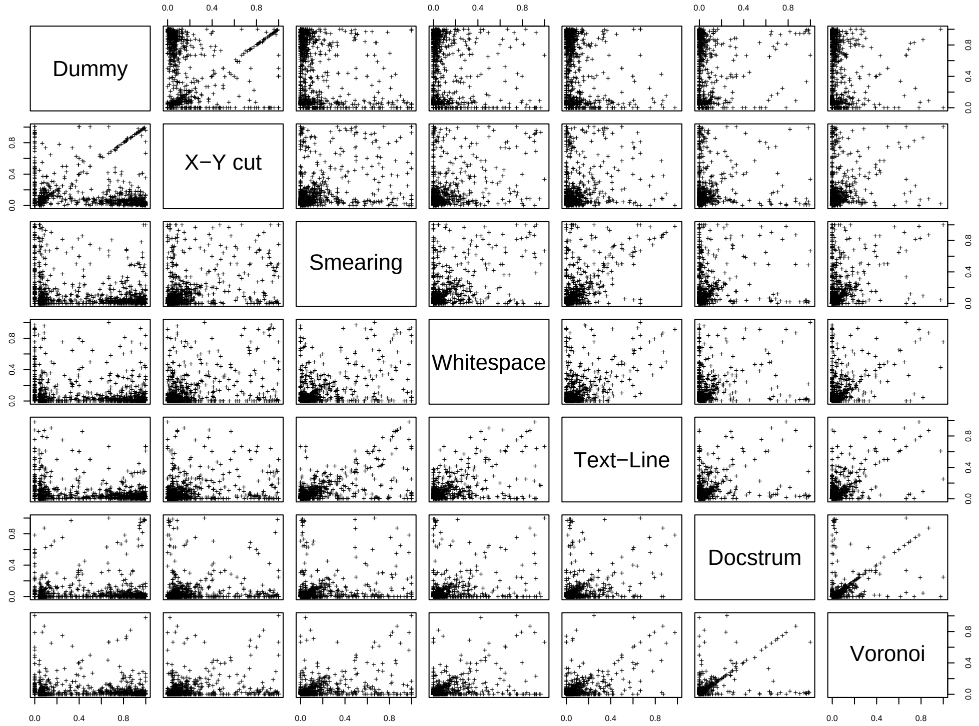
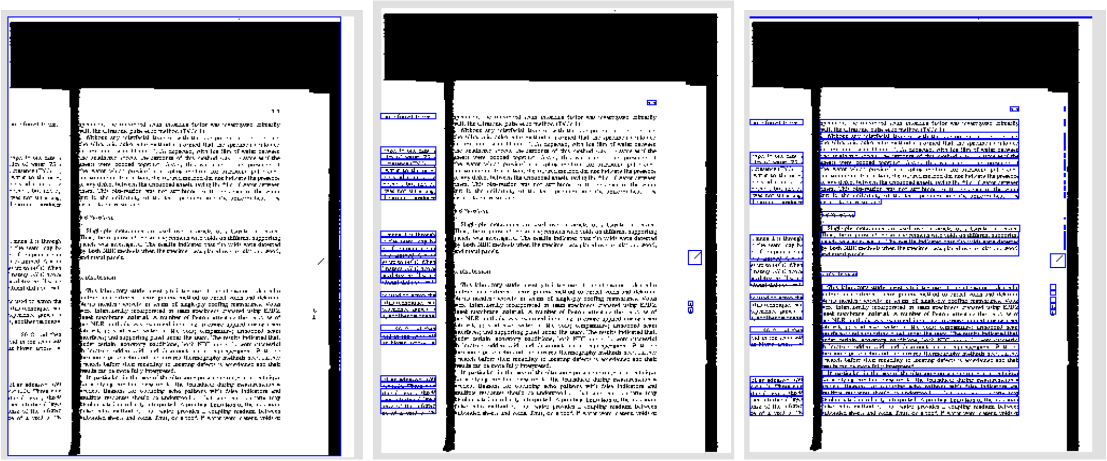


Figure 4.8: Correlation plot of the errors made by each algorithm. Each dot in the correlation plot represents one document image. The horizontal and vertical axis represent the error made by the corresponding algorithms. The plot shows that the Docstrum and the Voronoi algorithms show strong correlation, whereas the x-y cut algorithm has a high correlation with the dummy algorithm. (cp. [SKB08b])

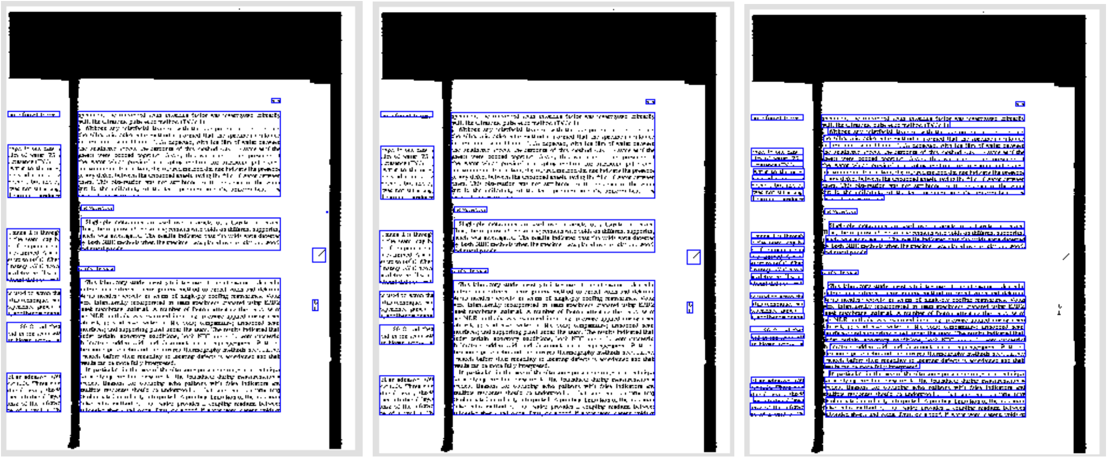
results are still valid. An interesting observation that can be made from the correlation plot is that for each algorithm, there are some documents on which it performs better than all the other algorithms. This indicates that combining the output of more than one algorithm might yield better results.

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 4.3.4. 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.



(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 ($\rho = 0.756$)

Figure 4.9: Segmentation results from applying each algorithm to one page image (D047) in the test set. The figure illustrates that according to the error rates calculated as in [MK01] (Equation 4.5), the x-y cut algorithm performs the best in this case, which is clearly mis-leading (cp. [SKB08b]).

Table 4.4: 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 4.5: Different types of errors made by each algorithm on textline-level ground truth. For a key to column labels please refer to Table 4.4.. 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

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 4.1 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 4.4. The smearing, and the constrained text-line finding algorithms locate text-lines in the given image. So they are evaluated on textline-level ground truth with the results given in Table 4.5.

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

Table 4.6: Different types of errors made by each algorithm on modified zone-level ground truth. For a key to column labels please refer to table 4.4.

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

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 usually 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.

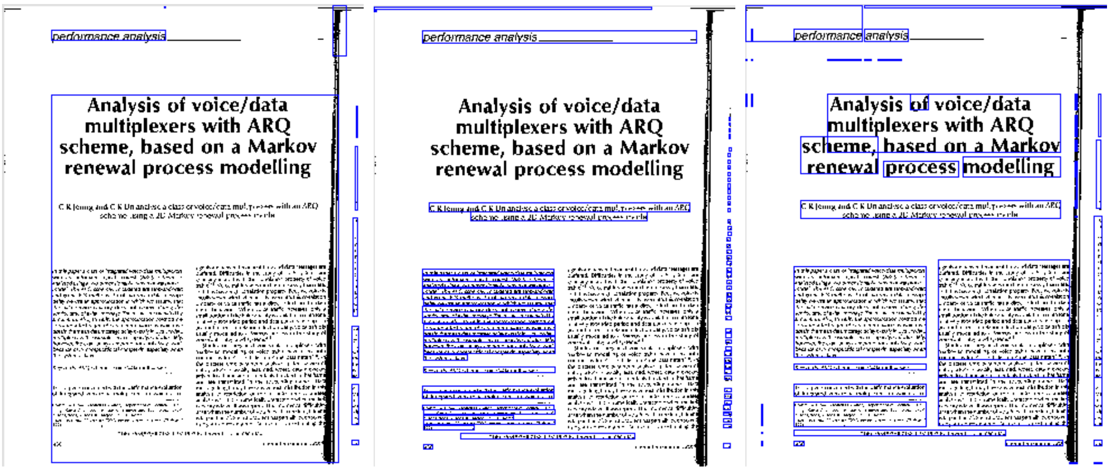
The ground truth for UW-III was modified to get text-zones instead of paragraphs. For this purpose, first a partial order of the text paragraphs was specified based on their spatial relationships, and then a topological sorting algorithm was used to find the reading order as in [Bre03b]. The bounding boxes of two consecutive paragraphs in the reading order were merged if their start and end positions along the horizontal direction were 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 4.6.

The result of applying each algorithm to an example image are shown in Figure 4.10. Based on the results in Tables 4.5 and 4.6, the following observations can be made about each algorithm:

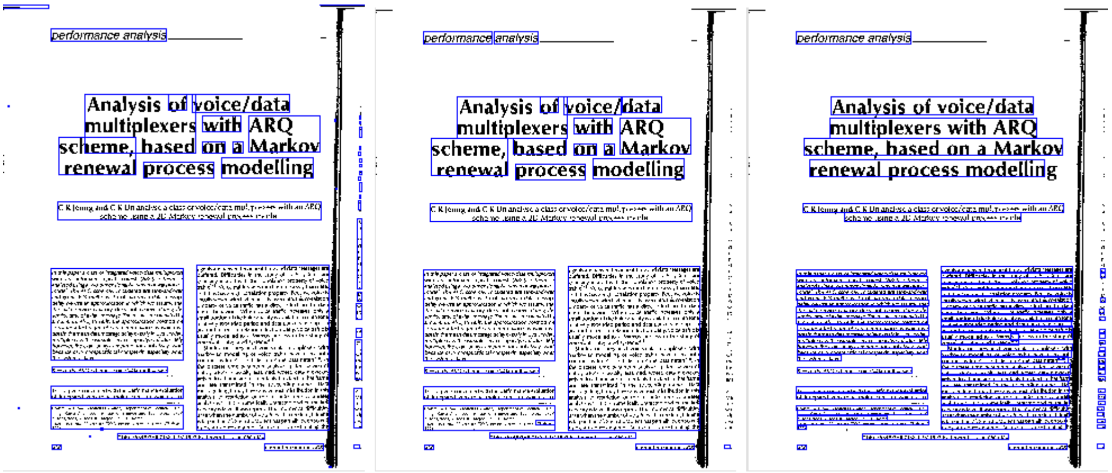
- The dummy algorithm has no correct segmentations and all the components are under-segmented.

- The x-y cut algorithm fails in the presence of noise and tends 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


performance analysis
Analysis of voice/data multiplexers with ARQ scheme, based on a Markov renewal process modelling

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


performance analysis
Analysis of voice/data multiplexers with ARQ scheme, based on a Markov renewal process modelling

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

Figure 4.10: 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. (cp. [SKB08b])

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 4.10) 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 merged. Hence, the bounding box of the lower text-line 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, an additional experiment was conducted. The Voronoi algorithm was chosen as a sample page segmentation algorithm. The goal of the experiment was to observe 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 4.11. 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

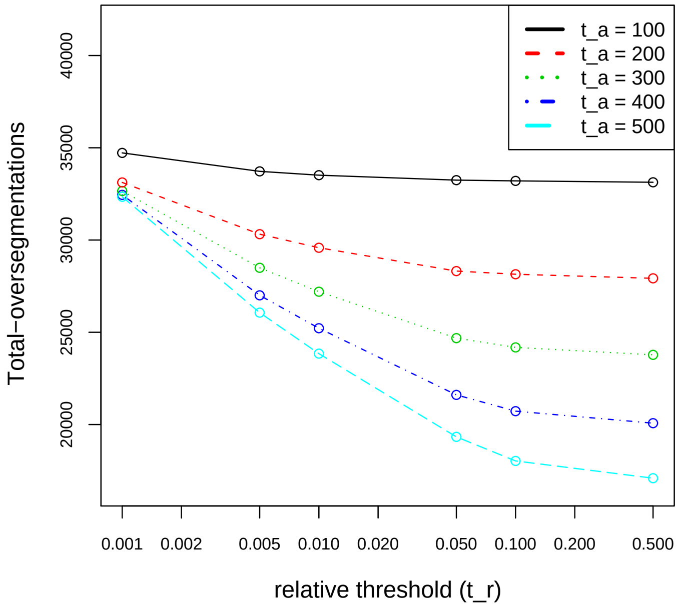


Figure 4.11: A plot of the values of total over-segmentations made by the Voronoi algorithm as the values of thresholds t_r and t_a defining significant edges are changed. The plot shows that setting either t_r or t_a to a very low value makes the performance measure independent of the other threshold.(cp. [SKB08b])

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, both thresholds should be considered 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 4.12. 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

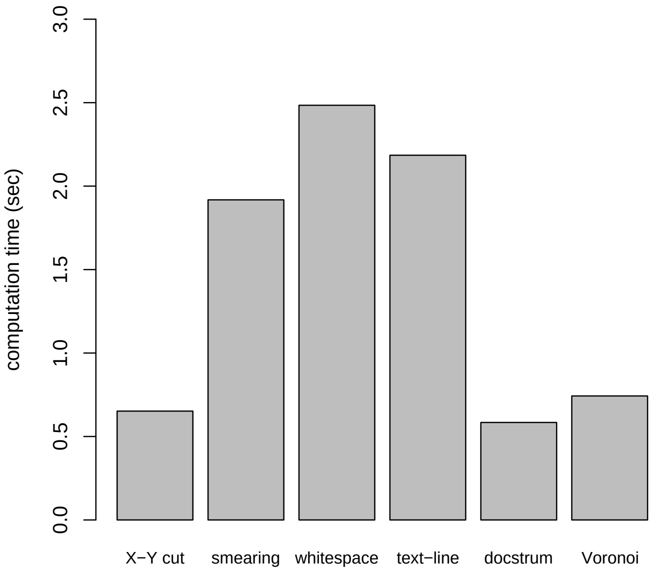


Figure 4.12: Average running time for each algorithm on the UW-III dataset. The experiment was run on an AMD Opteron 2.4 GHz machine running Linux. (cp. [SKB08b])

analysis, and constrained text-line finding algorithms.

4.4 Page Segmentation Using Whitespace Cuts

An analysis of the state-of-the-art page segmentation algorithms presented in Section 4.3.5 showed that zone-based algorithms (whitespace analysis, docstrum, Voronoi) tend to over-segment page title and section headings if their font size is much bigger than the rest of the text on the page. This effect is also evident from Figure 4.10 where the page title is over-segmented by whitespace analysis, docstrum, and Voronoi algorithms. This section presents a novel combination of the column-separator model by Breuel [Bre02c] with the whitespace analysis approach by Baird [Bai94] to solve the over-segmentation problem in Baird's approach. This new algorithm is named as the whitespace-cuts algorithm and is intended for segmenting Manhattan layouts. The main idea behind the algorithm is to first extract the columnar structure of the document based on the column-separator model by Breuel, and then extract horizontal whitespaces that respect this structure.

The whitespace-cuts algorithm proceeds by first extracting the connected components

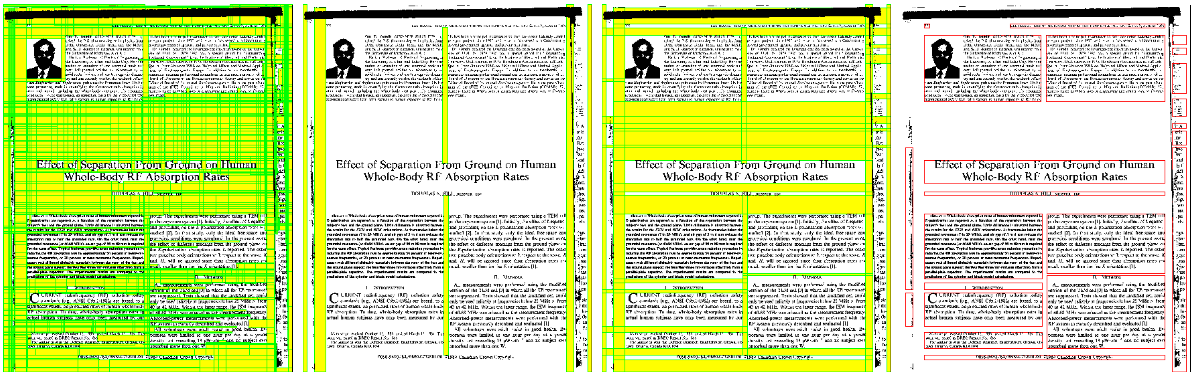


Figure 4.13: An example image illustrating different steps of the whitespace-cuts algorithm. Left to right: whitespace cover of the page background, extracted vertical separators and borders, extracted horizontal separators, extracted page segments.

in a document image using a fast labeling approach based on a union-find data structure. Noise removal is done at this stage by rejecting very small and very large connected components. Then statistics about the document, like inter-word spacing, inter-line spacing etc. are computed. These filtered connected components are used as input in the following steps described below.

4.4.1 Whitespace Cover Computation

The whitespace analysis algorithm described by Breuel [Bre02c] is used for finding a collection of maximal white rectangles (called *covers*), i.e. rectangles that do not overlap with any of the connected components on the page. The key idea behind the algorithm is similar to quicksort or branch-and-bound methods. The details of the algorithm can be found in [Bre02c]. The algorithm returns globally optimal covers in decreasing order with respect to their area, until a minimum area is reached or the maximum specified number of rectangles have been obtained. Usually 300 rectangles are sufficient to completely cover the page background. The result of applying the whitespace analysis to an example image is shown in Figure 4.13.

4.4.2 Extraction of Vertical Separators

The whitespace rectangles belonging to the whitespace cover are evaluated as candidates for vertical separators as in [Bre02c] based on the following constraints:

1. They must have an aspect ratio of at least 1:3

2. They must have a width of at least 1.5 times the mode of the distribution of widths of inter-word spaces.

3. They must be adjacent to a few character-sized connected components on their left and their right side. Such connected components are called the *neighbors* of the whitespace rectangle.

The candidates satisfying these criteria are selected as vertical separators. If there are overlapping vertical separators, the tallest one is selected. Then the vertical separators are filtered based on the text column width defined by two consecutive vertical separators along the horizontal direction. If the text-width is below a threshold, the vertical separator with lesser weight is dropped, where the weight of a whitespace rectangle is the total number of its neighbors multiplied by the normalized height of the whitespace. Some examples of vertical separators found in this way are shown in Figure 4.13.

After finding whitespace candidates for columns separators, the whitespaces representing the vertical borders of the document image are extracted. These borders are simply extracted by first computing a bounding rectangle of all the connected components of the page. Then the vertical page borders are the two vertical whitespace strips that touch the page border on one side and the bounding rectangle on the other side as shown in Figure 4.13. If the bounding rectangle covers the whole page area, two thin rectangles of 10 pixels width each are inserted at both the left and the right side of the page.

4.4.3 Extraction of Horizontal Separators

The selection of horizontal separators is based on the same concept as that for vertical separator. A useful horizontal separator should separate text spanning across multiple columns from text inside individual columns. Segmenting a single-column text into text-lines can be done using horizontal projection. The following constraints are imposed on horizontal separators similar to those in Section 4.4.2:

1. They must have an aspect ratio of at least 2:1

2. They must have a height greater than the mode of the distribution of inter-line spacing.

3. They must be adjacent to a few character-sized connected components on their upper and their lower side.

To restrict the horizontal separators to the columnar structure defined by vertical separators, the horizontal separators are constrained to touch at least two vertical separators. This constraint make sure that isolated horizontal separators are removed. If there are overlapping horizontal separators, the widest one is kept and the others are discarded. At this stage the hanging parts of all horizontal separators are also trimmed such that they do not extend beyond the extreme vertical separators touching them. An example of horizontal separators extracted in this way is shown in Figure 4.13.

4.4.4 Extraction of Page Segments

The set of horizontal and vertical separators segments the page into non-rectangular zones. These zones can be extracted by introducing the vertical and horizontal separators as obstacles and running the maximal empty rectangle extraction algorithm [Bre02c] again. The rectangles returned by this algorithm are the page segments. Then following post-processing steps are performed on these segments to obtain the final segmentation:

1. If there are character-sized connected components that are only partially included in a segment, the segment is enlarged such that these components are completely included.

2. A vertical projection profile of the character-sized connected components in each segment is computed. If the zero-valley in the projection profile is larger than 10 times the mode of the distribution of widths of inter-word spaces, the segment is split into two segments at the middle of the valley. This helps in correctly segmenting header and footer zones.

4.4.5 Experiments and Results

The evaluation of the page segmentation algorithms was done on the 100 training images and 878 test images from the UW-III collection as outlined in Section 4.3.5. The results obtained are shown in Table 4.7. The results show that the mean error rate for the whitespace-cuts algorithm is the lowest among all the compared algorithms. 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. Therefore it is interesting to count the number of documents for each algorithm for which it had the lowest error rate among all the compared algorithms. If more than one algorithm share the lowest error rate on a given document, the document is counted for each of them. A

Table 4.7: Evaluation results for different page segmentation algorithms on the 878 test images in terms of percentage of text-lines detection errors (Eq. 4.5).

	Error rates (%)		
	Train	Test	
Algorithm	Mean	Mean	Stdev
X-Y cut	14.7	17.1	24.4
Smearing	13.4	14.2	23.0
Whitespace	9.1	9.8	18.3
Text-line	5.6	7.0	13.3
Docstrum	4.3	6.0	15.2
Voronoi	4.7	5.5	12.3
Whitespace-cuts	1.7	4.4	11.1

comparison of the algorithms on this basis is shown in Figure 4.14. The results show that the whitespace-cuts algorithm has the best segmentation results for the highest number of documents among all the compared algorithms. However, it should be noted that the smearing and the text-line finding algorithm work on the text-line level, and therefore include the segmentation errors of one additional step, i.e. segmenting a text block into individual text-lines.

An analysis of the errors made by the whitespace-cuts algorithm shows that the algorithm made 2.0% split errors, 2.4% merge errors, and 0.01% missed errors. The main source of split errors was the vertical separators found in lists and references, thereby splitting the text-lines into two parts. The main source of merge errors was the text lines in headers and footers, where the page number was very close to the rest of the text in the header or footer. Also in some cases, column separators were missed thereby resulting in merged text-lines across the two columns. There were only a few missed errors owing mainly to single digit page numbers that were filtered out as noise. The average running time of the whitespace-cuts algorithm is about 1.1 seconds on an AMD Opteron 2.4 GHz machine running Linux.

4.5 Summary

This chapter presented an approach for evaluating page-segmentation algorithms using a color-based representation. The proposed 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 measuring the performance of an

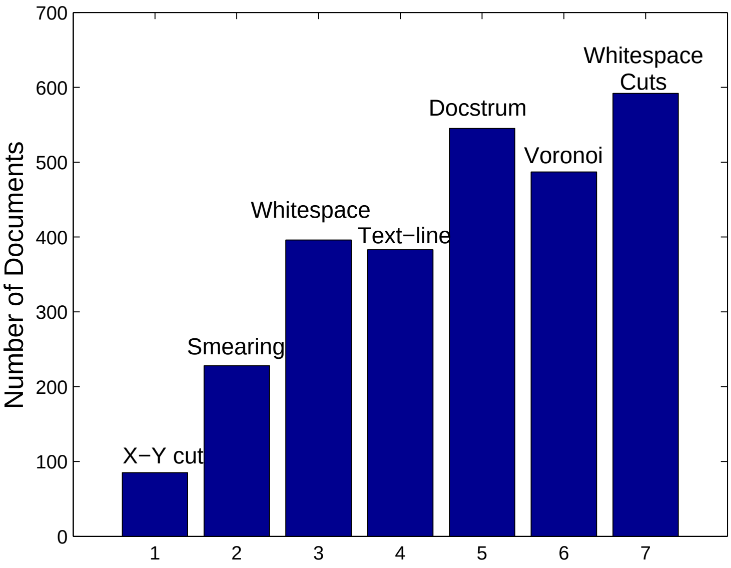


Figure 4.14: A bar plot showing the number of documents for each algorithm on which it had the lowest error rate among all the compared algorithms.

algorithm, different aspects of the algorithm are evaluated separately. Depending on the target application, different error measures may be weighed according to their significance in that application. Using these performance measures, the strengths and weaknesses of six well-known page segmentation algorithms were analyzed in this work.

The experiments presented in this work 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 or under-segmentations. The docstrum and the Voronoi algorithms tend to over-segment page title and section headings if the font size is much larger than the 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 these experiments, it can be concluded 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.

Finally, this chapter presented a highly accurate algorithm for page segmentation. The algorithm is based on the concept of whitespace-cuts, which is a modification of the whitespace analysis algorithm by Baird. The algorithm is tested on the UW-III dataset and is compared to six other well-known page segmentation algorithms. The results show that the proposed algorithm has the lowest mean error rate on the test data among all the compared algorithms. However, for each algorithm there are some images on which it works better than all the other images. Ranking the algorithms on the basis of the number of documents on which they have the lowest error rate results in the following order (best to worst): whitespace-cuts, Docstrum, Voronoi, whitespace analysis, text-line finding, smearing, and X-Y cut. Note that the text-line finding and smearing algorithms extract text-lines directly, and therefore include the errors of one additional step, i.e. segmenting a text block into individual text-lines.

Chapter 5

Zone Classification

Document zone classification aims at classifying the blocks detected by the page segmentation step (Chapter 4) of a geometric layout analysis system into one of a set of predefined classes (e.g. text, image, graphics, . . .). Blocks identified as text can then be fed to a character recognition module. Similarly, other actions can be taken for zones of specific types; for instance graphics regions can be sent to a raster to vector conversion program, whereas table zones can be fed to a table understanding system. Several algorithms for document zone classification have been proposed over the years (for a literature survey, please refer to [ODP99, WPH06]). This chapter presents a simple high-performance zone classification system that achieves the same error rate as the lowest reported in literature while using only simple and easy to compute features. This chapter presents the following contributions to the state-of-the-art in document zone classification[1]:

1. A detailed performance comparison of widely used features in document analysis and content-based image retrieval (CBIR) communities.
2. Introduction of the use of histograms for the measurements of connected components and run lengths and show that this leads to a performance increase.
3. Introduction of a new class of blocks containing speckles that has not been considered by other researchers. This typical class of noise is important to detect during the layout analysis especially for images of bound book pages.
4. Identification of the presence of duplicate documents in the UW-III dataset which might have positively influenced the results of other researchers.

---

[1] This chapter is based on the author's work in [KSB07].

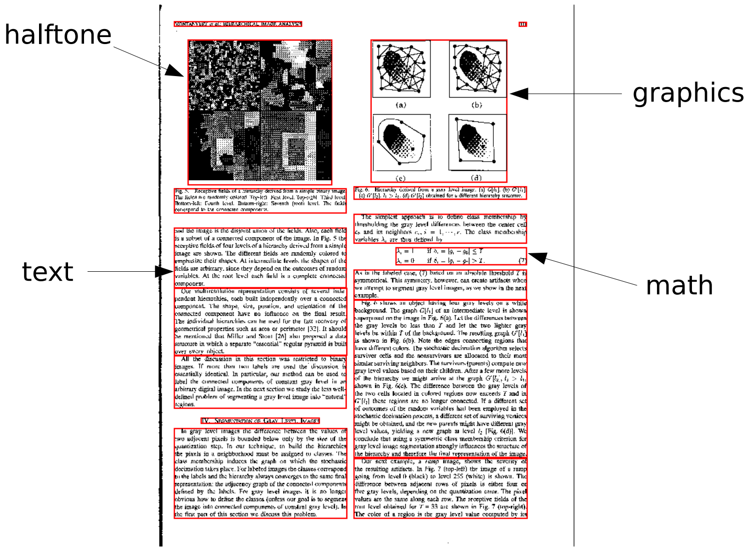


Figure 5.1: An example page of an article containing different types of zones.

5. Achieving an error rate of 1.5% with simple and computationally efficient features on the UW-III dataset without using duplicates, which equals the lowest error rate reported in literature [WPH06] using more complex features.

5.1 Introduction

A document image may contain different types of contents, like text, math, figures, etc. An example document containing different types of zones is shown in Figure 5.1. There is a constant interest in the document image analysis community on document zone content classification problem. Many approaches have been proposed in literature for classifying the contents of a document zone into one of the predefined classes. However, most of these approaches focus on extraction of specific application-dependent zone classes. For instance, Xiao and Yan [XY03] worked on text region extraction problem, Zanibbi et al. [ZBC02] on Mathematics expression recognition, Kieninger and Dengel [KD98a, KD98b, KD01] on table extraction problem, Chen et al. [CLG03] and Pham [Pha03] on logo detection, Li et al. [LNG00] on image (halftone) extraction problem, and Futrelle et al. [FSCG03] on diagram (drawing) extraction and classification problem. Due to a wide diversity of objectives for document understanding, this work focuses on recognition of entities that usually appear in technical journal documents. For this purpose, eight classes of zones are considered in this chapter: math, logo, text, table, drawing, halftone, ruling, and speckles. Sample images of each class are shown in Figure 5.2. Some related work for classifying document zones into different target classes is presented in this section. For a more detailed overview of related work in the field of document zone classification please refer to [ODP99, WPH06].

Inglis and Witten [IW95] present a study of the zone classification problem as a machine learning problem. They use 13831 zones from the UW database and distinguish the three classes text, halftone, and drawing. Using seven features based on connected components and run lengths, the authors apply various machine learning techniques to the problem, of which the C4.5 decision tree performs best at 6.7% error rate.

The review paper by Okun et al. [ODP99] succinctly summarizes the main approaches used for document zone classification in the 1990s. The predominant feature type is based on connected components (see also for example [LPHH96]) and run-length statistics. Other features used include the cross-correlation between scan-lines, vertical projection profiles, wavelet coefficients, learned masks, and the black pixel distribution. The most common classifier used is a neural network.

The most recent and detailed overview of the progress in document zone classification and a very accurate system is presented by Wang et al. [WPH06]. The authors represent each zone with a 25 dimensional feature vector and use an optimized decision tree classifier to classify each zone into one of nine target classes. They use 24177 zones extracted from the UW-III database to evaluate their approach and report an accuracy of 98.45%. The features they use include run-length statistics, spatial features, auto-correlation features, background features, a text glyph feature and a feature based on column width ratio. Additionally, they incorporate contextual constraints in the classification of some zones to get better classification results.

The approaches presented in the literature have one common short-coming: they present the performance of a complete system without highlighting the contribution of each feature to the overall system performance. This work fills this gap by comparing a large set of commonly used features for block classification and includes in the comparison three features that are known to yield good performance in content-based image retrieval (CBIR) and are applicable to binary images [DKN04]. A detailed analysis of the performance of the features used in the document analysis and the CBIR communities enables the development of a zone classification system that is based on very simple and computationally efficient features and yields the same performance as the state-of-the-art block classification system by Wang et al. [WPH06]. An overview of the features used in

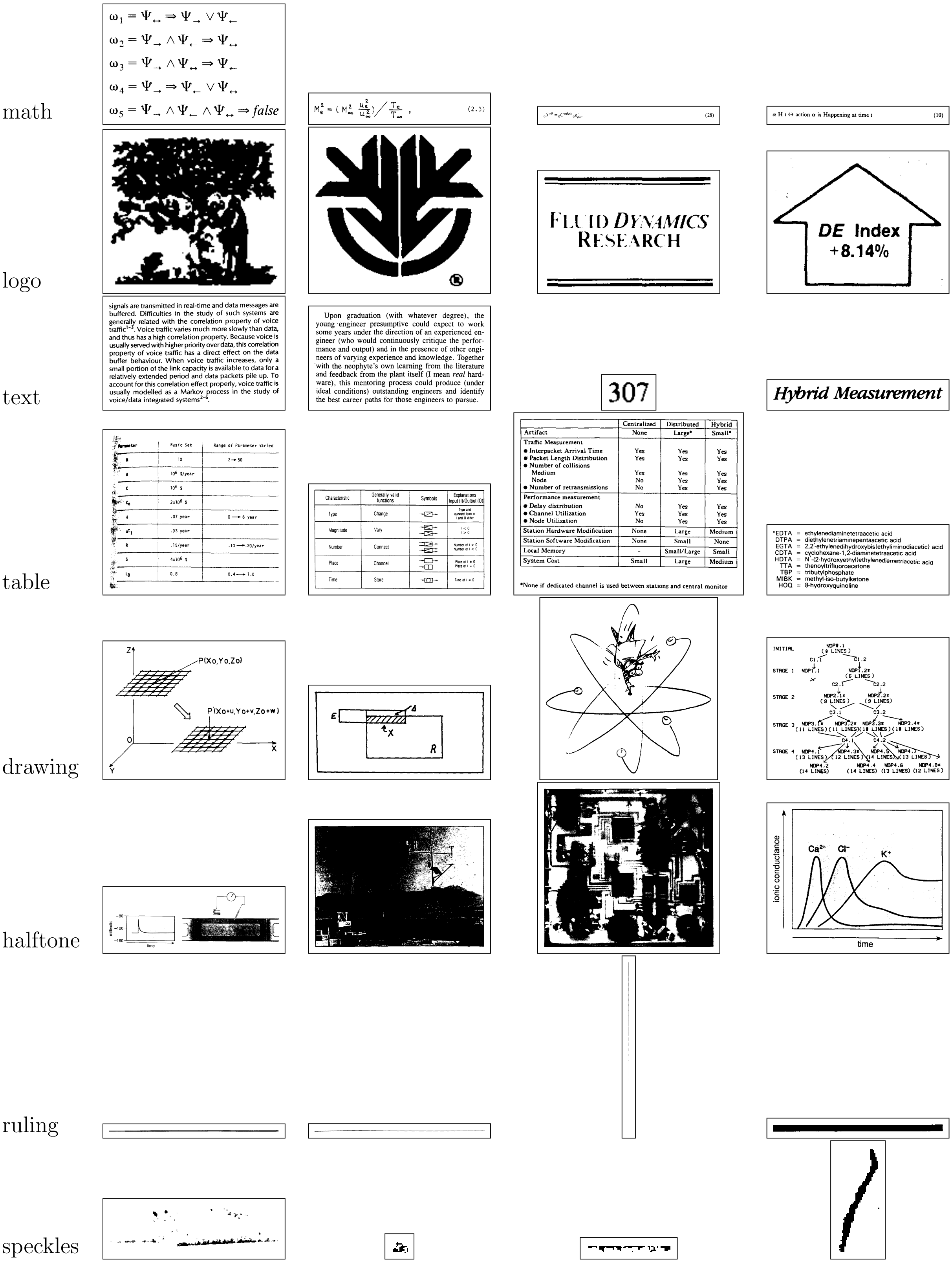


Figure 5.2: Examples of document image block types distinguished in this work.

this work is given in Section 5.2. A simple nearest-neighbor classifier (Section 5.3) was used in this work since the focus was on extracting suitable features for zone classification. The dataset used and the experimental results are discussed in Section 5.4 followed by a conclusion in Section 5.5.

5.2 Feature Extraction

The features used in this work can be grouped into two categories:

1. features known to work well in content-Based image retrieval (CBIR) [DKN04]
2. features commonly used in the document zone classification literature [ODP99, WPH06]

The overall most successful features in CBIR are usually based on color information. This work restricts the analysis for zone classification to those features that are promising for the analysis of binary images. The CBIR features chosen in this work are based on the open source image retrieval system FIRE [DKN04]. These features include Tamura texture features histogram, relational invariant feature histograms, and down-scaled images of size 32×32.

The most widely used features in the document zone classification literature [ODP99, WPH06] are based on connected components and run-length statistics. These features not only yield high performance, but also are simple to implement. Hence these feature types were used as representative from the document zone classification literature. An outline of all the features compared in this work is:

1. Tamura texture features histogram (TTFH)
2. Relational invariant feature histograms (RIFH)
3. Down-scaled images of size 32×32 (DSI)
4. The fill ratio, i.e. the ratio of the number of black pixels in a horizontally smeared [WCW82] image to the area of the image (FR)
5. Run-length histograms of black and white pixels along horizontal, vertical, main diagonal, and side diagonal directions; each histogram uses eight bins, spaced apart as powers of 2, i.e. counting runs of length $\leq 1, 3, 7, 15, 31, 63, 127$ and ≥ 128 (RL{B,W}{X,Y,M,S}H)

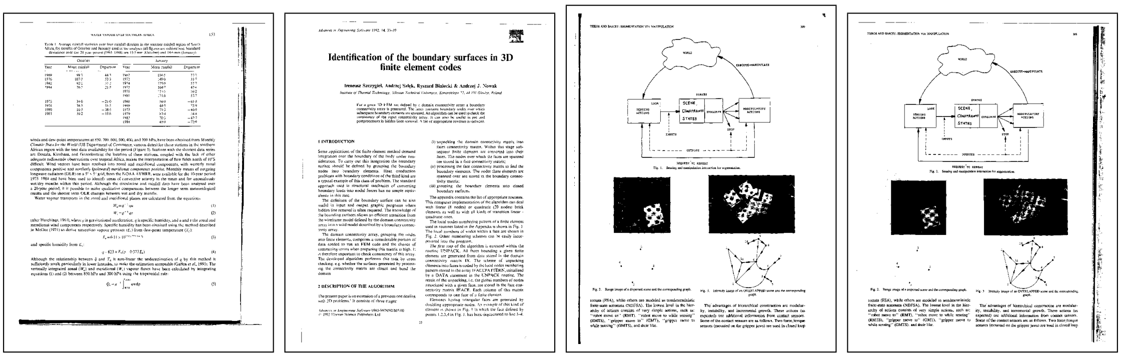
6. The vector formed by the total number, mean, and variance of the runs of black and white pixels along the horizontal, vertical, main diagonal, and side diagonal directions as used in [WPH06] (RL{B,W}{X,Y,M,S}V)
7. Histograms (as in 5) of the widths and heights of connected components (CCXH, CCYH)
8. The joint distribution of the widths and heights of connected components as a 2-dimensional 64-bin histogram (CCXYH)
9. The histogram of the distances between a connected component and its nearest neighbor component (CCNNH)

5.3 Classification

To evaluate the various features, a simple nearest neighbor classifier is used. A test sample is classified into the class the closest training sample belongs to. The distance measures used in this work are the Jensen-Shannon divergence for histograms and the Euclidean distance for all other features [DKN04]. When different feature sets are combined, the overall distance is calculated as the weighted sum of the individual normalized distances. The weights are set proportional to the inverse of the error rate of a particular feature. No tuning with respect to these weights or with respect to the distance measures was performed. Although a k-nearest-neighbor approach gives better results in many cases only the 1-nearest-neighbor classifier was evaluated in this work. The nearest neighbor error rates were determined using leave-one-out cross-validation. The nearest neighbor classifier serves as a good baseline classifier, although in many cases a more suitable classifier can be found for a given task. As this work concentrates on feature extraction and analysis a detailed study of suitable classifier was not performed. An important shortcoming of the nearest neighbor classifier is its requirement on computational resources. Both memory and run-time can be prohibitive for some applications.

5.4 Experiments and Results

To evaluate the presented approach for document zone classification, the University of Washington III (UW-III) database [GHHP97] is used. The database consists of 1600 English document images with bounding boxes of 24177 homogeneous page segments or



(a) E00D (b) C000 (c) W033 (direct scan) (d) E04A (photocopy)

Figure 5.3: Example document pages from the UW-III database. Note that some documents, as shown on the right, occur in different versions. For our experiments, we made sure that no such duplicates were used.

Table 5.1: Summary of UW zone classification error rates from the literature along with the number of pages, zones and block types used. Note that an exact comparison between all error rates is not possible.

reference	# pages	# zones	# types	error [%]
[IW95]	1001	13831	3	6.7
[LPHH96]	979	13726	8	5.4
[SPH$^+$95]	979	13726	9	3.3
[WHP00]	1600	24177	9	2.5
[WPH06]	1600	24177	9	1.5
this work	713	13811	8	1.5

blocks, which are manually labeled into different classes depending on their contents, making the data very suitable for evaluating a block classification system. Table 5.1 shows an overview of related results in zone classification on the UW dataset along with the results obtained in this work.

The documents in the UW-III dataset are categorized based on their degradation type as follows:

1. Direct scans of original English journals
2. Scans of first generation English journal photocopies
3. Scans of second or later generation English journal photocopies

Some documents in the dataset are duplicated and differ sometimes only by the degradation applied to them. This type of collection is useful when one is evaluating a page segmentation algorithm to see how well the algorithm performs when, for instance, photocopy effect degradation is applied to a document. However, the degradation introduced by photocopying a document does not affect the appearance of the contents of a document to a large extent. One such example can be seen in Figure 5.3, where the same document is present in the dataset four times (E04A, W033, S04A, W133, two of them shown here). Although the photocopied documents are usually darker than the corresponding direct scans, the difference is not substantial. This duplication of documents tends to bias the evaluation results towards lower error rates when some of these documents are used in training, while others are used in testing. This effect seems to have been unnoticed previously by other researchers who use the complete dataset for the evaluation of their algorithms.

In this work, a subset of the UW-III dataset was used to avoid using duplicate documents. The documents in the *scans from the first generation photocopies* category were chosen because they were largest in number. These are all the documents with prefixes A0, C0, D0, IG, H0, J0, K0, E0, V0, I03, and I04. There are 713 documents of this type. The ground-truth zones and their labels from each of these 713 documents were extracted. For some of the zone types like "seal", "announcement", "advertisement", etc. there were only a few samples in the dataset. Hence only those classes were considered for evaluation that contained at least ten example images.

One limitation of the UW-III ground-truth zones is that they do not contain any example of noise regions, i.e. regions that emerge from noise introduced during the scanning or photocopy process. These regions mostly consist of speckles and dots present along the border of the document. Since such regions often appear in practice, it is important to detect such regions as noise so that these can be removed from further processing. The UW-III dataset images contain many such regions but these are not labeled. In order to extract examples of such regions the page segmentation algorithm from [KSI98] was used to extract page segments. Then all the segments that did not overlap with any of the ground-truth zones were filtered out as examples of the noise zones. However these contained both textual and non-textual noise. Textual noise appears only along the left or the right side of a document when the facing pages of a book are scanned. Since these extraneous symbols cannot be distinguished from the actual contents of the document based on their appearance alone, textual noise is not considered in this work. Therefore only examples of non-textual noise, i.e. speckles were taken as noise class. The speckles heavily depend on the degradation of the document and vary considerably from the direct scan of a document to its first generation photocopy as can be seen in Figure 5.3.

Table 5.2: Leave-one-out nearest neighbor error rates and extraction run-times for each feature and for combinations.

feature	# features	extr.-time [s]	error [%]
TTFH	512	5.51	**3.4**
RIFH	512	12.59	7.8
DSI	1024	0.01	8.1
FR	1	0.02	27.3
RLBXH	8	0.01	7.9
RLWXH	8	0.01	5.1
RLBYH	8	0.01	8.2
RLWYH	8	0.01	5.6
RLBMH	8	0.01	11.8
RLWMH	8	0.01	6.6
RLBSH	8	0.01	10.5
RLWSH	8	0.01	6.2
RLBXV	3	0.01	12.9
RLWXV	3	0.01	9.7
RLBYV	3	0.01	14.6
RLWYV	3	0.01	12.1
RLBMV	3	0.01	17.2
RLWMV	3	0.01	12.6
RLBSV	3	0.01	16.7
RLWSV	3	0.01	12.2
CCXH	8	0.04	14.5
CCYH	8	0.04	14.9
CCXYH	64	0.04	6.2
CCNNH	8	0.05	19.0
RL**V, constant weight			4.1
RL**H, constant weight			1.8
RL*, CC*, 1/error weight			**1.5**
FR, RL*, CC*, 1/error weight			1.5
TTFH, FR, RL*, CC*, 1/error weight			1.5

Therefore, for the speckles class, examples were extracted from all 1600 documents of the UW-III database. The corresponding number of examples used for each zone type is included in Table 5.3.

Table 5.2 shows the error rates that the nearest neighbor classifier achieves for each single feature along with the dimensionality of the feature vectors and the average time used to compute the feature vector. (All timing information is given for a standard PC with 1.8GHz AMD Athlon processor without special performance tuning of the algorithms.) The last rows show results for combined feature sets.

The following results can be observed from Table 5.2:

- The Tamura texture feature is the single best feature but is more than 100 times slower to compute than most other features.
- The use of histograms as descriptors of the run-lengths distribution leads to much lower error rates than the use of number, mean, and variance. The combination of these histograms alone leads to a very good error rate of 1.8%.
- Interestingly, the use of the white (background) runs for the computation of features consistently leads to better results than the use of black (foreground) runs.
- Among the run-lengths based features, those based on the horizontal runs lead to the best error rates.
- The fill ratio as a single feature does not lead to good results, which is not surprising as it consists only of a single number. However, it is very useful to distinguish drawings from text. This is however also achieved by using the distribution of the white run lengths, such that the FR feature is not part of the best observed feature set.

Table 5.3 shows the frequency of misclassifications between different classes of the best classifier. It can be observed that high recognition accuracy was achieved for the text, ruling, speckles, math, halftone, and drawing classes. However, our system failed to recognize logos correctly, and most of the logos were misclassified as either text, or halftone/drawing. Note that the accuracy rate for type ‘logo’ in [WPH06] is even lower at 0.0%. The reason for this effect is the very small number of samples for this class, which on the other hand implies that it has only a very small influence on the overall system error rate. Similarly, the table detection accuracy was not high, and about 21% of the tables were misclassified as text.

Table 5.3: Contingency table showing the distribution of the classification of zones of a particular type in percent. (The total number of errors equals 201 within 13811 tests.) The labels M, L, T, A, D, H, R, S correspond to the types math, logo, text, table, drawing, halftone, ruling, and speckles, respectively.

	M	L	T	A	D	H	R	S	error [%]	# samples
M	90.8	0.0	8.6	0.0	0.0	0.6	0.0	0.0	9.2	476
L	9.1	27.3	36.4	0.0	9.1	9.1	0.0	9.1	72.7	11
T	0.1	0.0	99.8	0.0	0.0	0.0	0.0	0.0	0.2	10450
A	0.8	0.0	20.7	68.6	9.9	0.8	0.0	0.0	31.4	121
D	1.5	0.3	3.0	5.5	86.0	3.5	0.0	0.3	14.0	401
H	0.0	0.9	0.0	0.0	9.7	86.7	0.9	1.8	13.3	113
R	0.4	0.0	1.3	0.0	0.4	0.0	96.1	2.2	3.9	232
S	0.1	0.0	0.5	0.0	0.1	0.1	0.0	99.4	0.6	2007

To visualize the errors made, the nearest-neighbor images for each misclassified block were investigated. Figures 5.4, and 5.5 shows some typical examples of misclassified zones. It can be seen that some of these images cannot be simply classified correctly by using the block content alone, and even humans are likely to make errors if they are asked to classify these images.

The error rates achieved in this work are competitive to the best error rate reported in literature by Wang et al. [WPH06]. For a comparison it should be noted that at most 0.2% (53/24177) of the error rate Wang et al. present is caused by their distinction between the text classes of different font-sizes and the class “other” with the remaining classes. On the other hand, a new class “speckles” is added in this work, which is related to 0.15% (21/13811) error. Additionally the presence of duplicates in the UW-III dataset was ignored by Wang et al. As they use 9-fold cross-validation to obtain their results, it might be possible that the error rates they present (the best result is an overall error rate of 1.5%) may be influenced positively by this fact, because it is likely that instances of blocks of the same document occur in training and test set. In a similar direction, Wang et al. use one feature that “uses a statistical method to classify glyphs and was extensively trained on the UWCDROM-III document image database.” It is not clear if this implies that the glyphs that occur in testing have also been used in the training of the glyph classifier. Finally, in this work zone context modeling is not used, although it is likely that a context model (which can be integrated in a similar way as presented by Wang et al.) would help the overall classification performance.

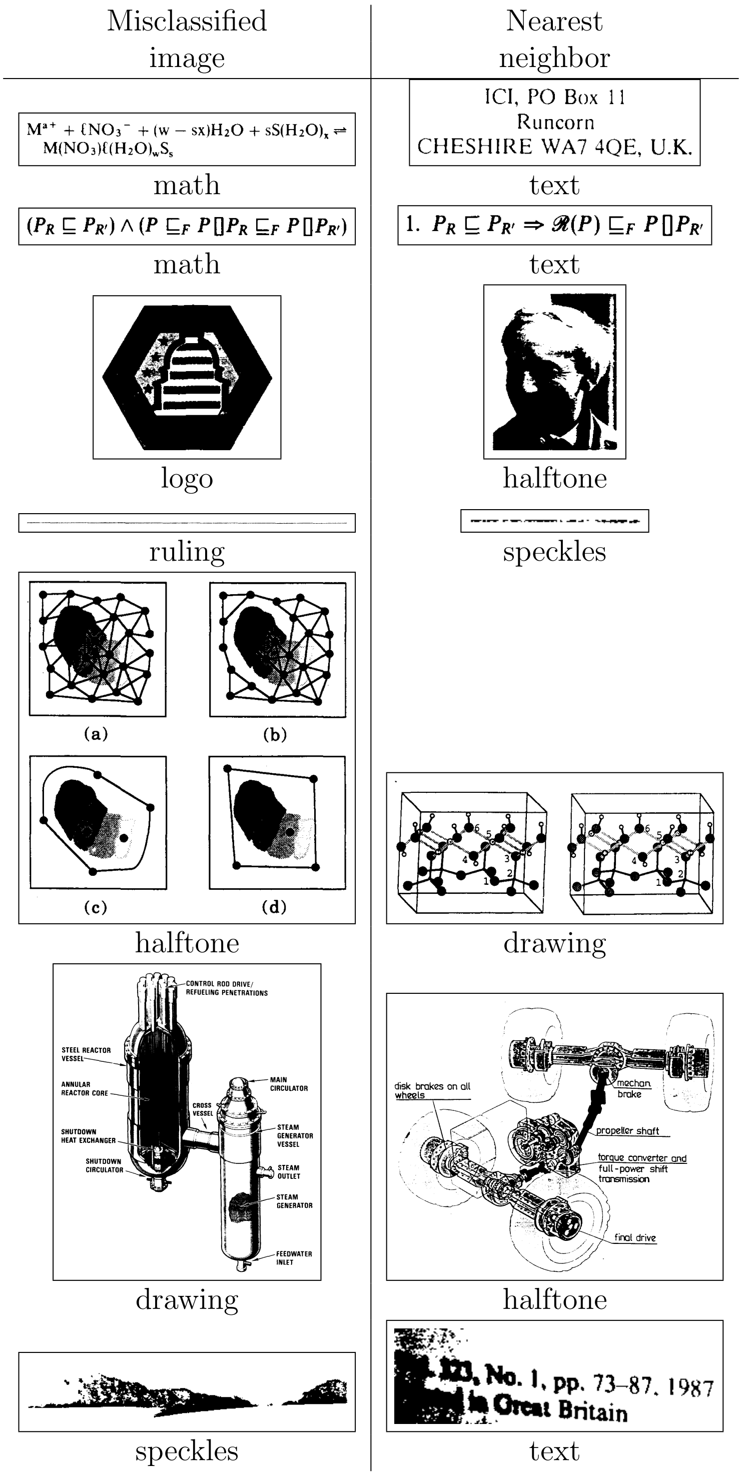
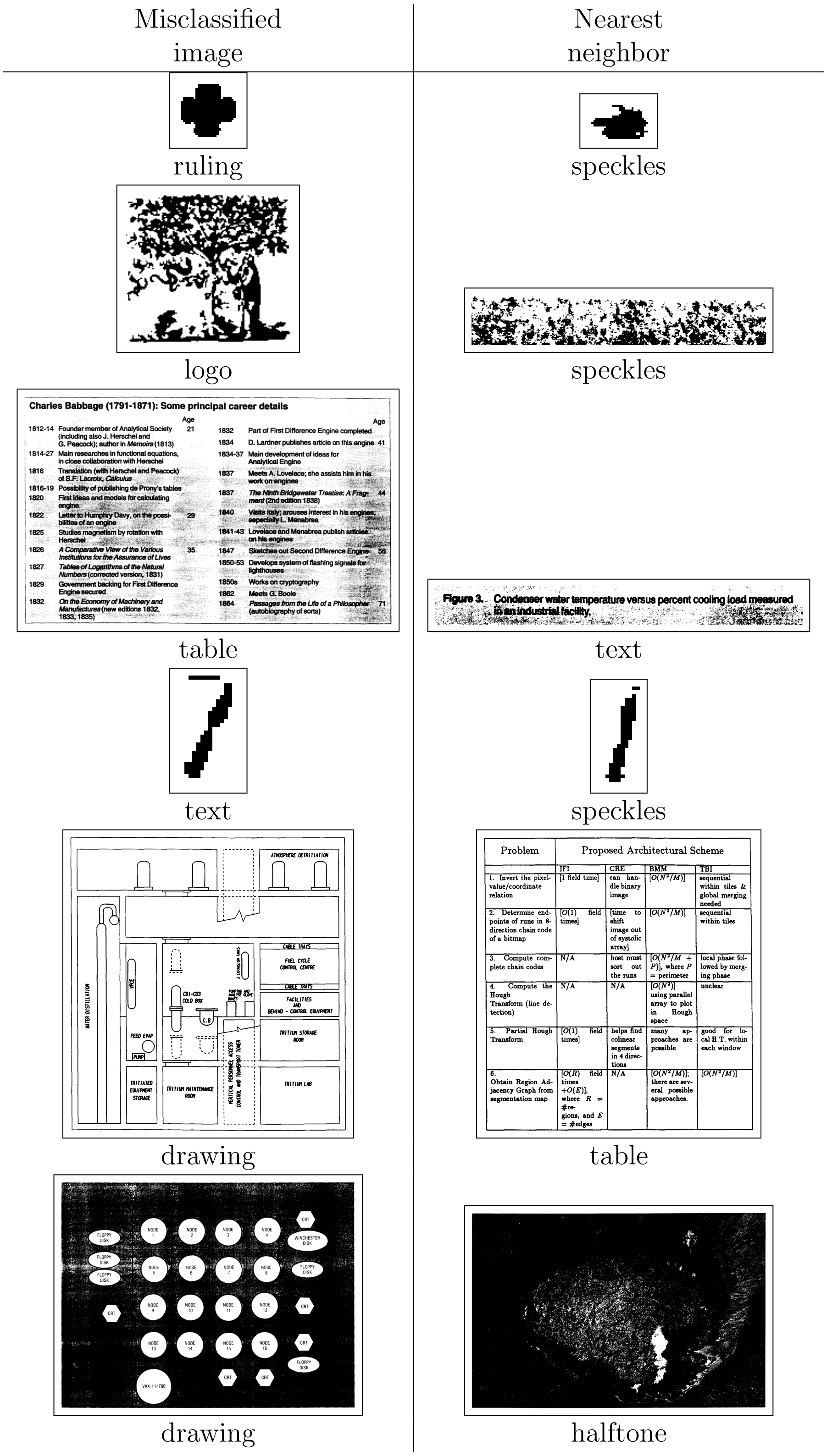


Figure 5.4: Examples of misclassification showing the misclassified image and its nearest neighbor from a different class.



Charles Babbage (1791-1871): Some principal career details

		Age
1812-14	Founder member of Analytical Society (including also J. Herschel and G. Peacock); author in *Memoirs* (1813)	21
1814-27	Main researches in functional equations, in close collaboration with Herschel	
1816	Translation (with Herschel and Peacock) of S.F. Lacroix, *Calculus*	
1816-19	Possibility of publishing de Prony's tables	
1820	First ideas and models for calculating engine	
1822	Letter to Humphry Davy, on the possibilities of an engine	29
1825	Studies magnetism by rotation with Herschel	
1826	*A Comparative View of the Various Institutions for the Assurance of Lives*	35
1827	*Tables of Logarithms of the Natural Numbers* (corrected version, 1831)	
1829	Government backing for First Difference Engine secured	
1832	*On the Economy of Machinery and Manufactures* (new editions 1832, 1833, 1835)	
1832	Part of First Difference Engine completed	
1834	D. Lardner publishes article on this engine	41
1834-37	Main development of ideas for Analytical Engine	
1837	Meets A. Lovelace; she assists him in his work on engines	
1837	*The Ninth Bridgewater Treatise: A Fragment* (2nd edition 1838)	44
1840	Visits Italy; arouses interest in his engines, especially L. Menabrea	
1841-43	Lovelace and Menabrea publish articles on his engines	
1847	Sketches out Second Difference Engine	56
1850-53	Develops system of flashing signals for lighthouses	
1850s	Works on cryptography	
1862	Meets G. Boole	
1864	*Passages from the Life of a Philosopher* (autobiography of sorts)	71

Figure 3. Condenser water temperature versus percent cooling load measured in an industrial facility.

Problem	Proposed Architectural Scheme			
	IFI	CRE	BMM	TBI
1. Invert the pixel-value/coordinate relation	[1 field time]	can handle binary image	$[O(N^2/M)]$	sequential within tiles & global merging needed
2. Determine end-points of runs in 8-direction chain code of a bitmap	$[O(1)$ field times]	[time to shift image out of systolic array]	$[O(N^2/M)]$	sequential within tiles
3. Compute complete chain codes	N/A	host must sort out the runs	$[O(N^2/M + P)]$, where P = perimeter	local phase followed by merging phase
4. Compute the Hough Transform (line detection)	N/A	N/A	$[O(N^2)]$ using parallel array to plot in Hough space	unclear
5. Partial Hough Transform	$[O(1)$ field times]	helps find colinear segments in 4 directions	many approaches are possible	good for local H.T. within each window
6. Obtain Region Adjacency Graph from segmentation map	$[O(R)$ field times $+O(E)]$, where R = #regions, and E = #edges	N/A	$[O(N^2/M)]$; there are several possible approaches.	$[O(N^2/M)]$

Figure 5.5: Some more examples of misclassification showing the misclassified image and its nearest neighbor from a different class.

5.5 Summary

This chapter showed that a very accurate document zone classification system can be constructed based on a feature vector consisting of run-length histograms alone. Run-length features are very easy to implement and fast to extract and thus should be part of any practical zone classification system. Interestingly, the distribution of the background runs is more important for document zone classification than the distribution of the foreground runs. In this work, a very competitive error rate of below 1.5% was obtained on zones extracted form the UW-III database using features based on run-length and connected component statistics only. Examination of the errors made by the system makes it seem likely that further improvements significantly below the error rate reached in this work may be difficult to achieve without a significantly increased effort, for example by using a dedicated sub-classifier to distinguish between text and table zones.

Chapter 6

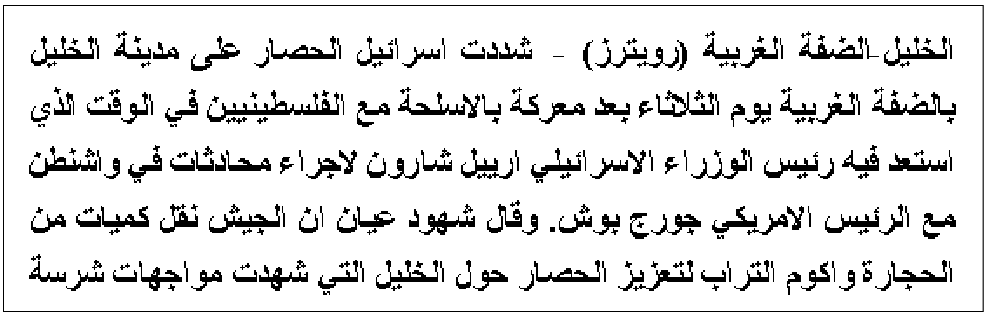
Layout Analysis of Urdu Documents

Automatic recognition of documents in scripts other than Roman like Arabic, Chinese, Japanese etc. has gained a lot of attention in recent years. However, recognition of printed Urdu - the national language of Pakistan - has largely been ignored. To date, no working Urdu OCR system is known to the author. One of the main reasons for the lack of a complete Urdu OCR system is a limited support of the Urdu language in computing environments. Many software do not have an Urdu user interface, and only a few Urdu editors are available. Recent projects like *Urdu Localization Project* [Hus04] aimed at improving Urdu language support in computing environments. These advances in the support of Urdu in computing are leading to an increased interest in digitization of Urdu literature resulting in more demand for an Urdu OCR system. Layout analysis is a crucial part of an OCR system. Kumar et al. [KKJ07] have recently reported that none of the six well-known page segmentation algorithms outlined in Chapter 4 of this thesis work well for segmenting printed Urdu documents. This chapter presents the first high-performance layout analysis system for Urdu documents and demonstrates that the system works well in segmenting documents in different layouts. The following contributions to the state-of-the-art in automatic recognition of printed Urdu documents are presented in this chapter[1]:

1. A geometric model is developed for representing an Urdu text-line.
2. A geometric matching method is used to extract text-lines from printed Urdu documents based on the geometric model of an Urdu text-line.
3. A reading-order determination algorithm for Roman script documents is modified to adapt to Urdu documents.

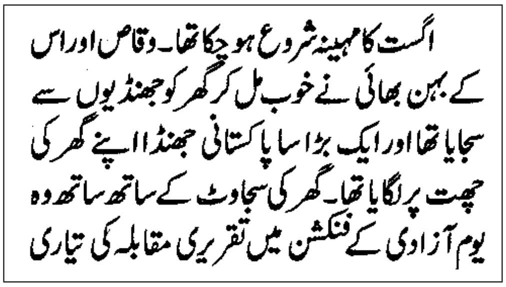
---

[1] This chapter is based on the author's work in [SuHKB06].



الخليل-الضفة الغربية (رويترز) - شددت اسرائيل الحصار على مدينة الخليل
بالضفة الغربية يوم الثلاثاء بعد معركة بالاسلحة مع الفلسطينيين في الوقت الذي
استعد فيه رئيس الوزراء الاسرائيلي ارييل شارون لاجراء محادثات في واشنطن
مع الرئيس الامريكي جورج بوش. وقال شهود عيان ان الجيش نقل كميات من
الحجارة واكوام التراب لتعزيز الحصار حول الخليل التي شهدت مواجهات شرسة

(a) Sample Arabic document written in Naskh script



اگست کا مہینہ شروع ہو چکا تھا۔ وقاص اور اس
کے بہن بھائی نے خوب مل کر گھر کو جھنڈیوں سے
سجایا تھا اور ایک بڑا سا پاکستانی جھنڈا اپنے گھر کی
چھت پر لگایا تھا۔ گھر کی سجاوٹ کے ساتھ ساتھ وہ
یوم آزادی کے فنکشن میں تقریری مقابلہ کی تیاری

(b) Sample Urdu document written in Nastaliq script

Figure 6.1: An example of printed Arabic and Urdu text. Nastaliq script is derived from Naskh script but has a more complicated nature. Text-lines in Nastaliq script are taller and have very little spacing between them.

6.1 Introduction

Urdu is the national language of Pakistan with more than 150 million speakers. Urdu alphabet is written in *Nastaliq* script, which is also used to write other languages like Persian and Pashto. Although Nastaliq script is sometimes used to write Arabic text as well, the pre-dominant script used for writing Arabic is *Naskh*. On the other hand, a small fraction of Urdu documents are also printed in Naskh script. An example of printed Urdu and Arabic texts is shown in Figure 6.1, where Urdu text is written in Nastaliq script and Arabic text is written in Naskh script. Nastaliq script has many differences to Naskh script, the most important of which from layout analysis point of view are very small inter-line spacing, and tall ascenders and descenders that penetrate into adjacent text-lines. An example of this case can be seen in Figure 6.1(b), where ascender from the second text-line gets merged with the dots belonging to the first text-line.

There has been very little work in the area of Urdu document analysis. Husain et al. [HA02] proposed an Urdu character recognition system for the Nastaliq script. It is a cursive script, i.e. individual characters are usually combined to form ligatures. Urdu is written in Nastaliq script using more than 20,000 ligatures [Wik]. Husain et al. skipped the layout analysis step to concentrate more on the OCR part. They chose only some frequently used characters and ligatures for their experiments. They trained a neural network with samples of these ligatures and tested its performance. However, the neural network based approach is not suitable for recognizing printed Urdu because collecting a large number of samples for each of the 20,000 ligatures becomes prohibitive.

Pal et al. [PS03] present an approach for recognizing printed Urdu documents. First, they perform skew correction of the document using Hough transform. Text-lines in the skew corrected document are then segmented by horizontal projection. Finally, each extracted line is segmented into individual ligatures by vertical projection and connected component labeling. Once ligatures belonging to a text-line are obtained, they are fed to a two-stage character recognition system. Pal et al. focused on the recognition of basic characters and numerals and did not consider the recognition of compound characters. In the first stage the characters are grouped into a few subsets by a tree classifier using simple features. In the second stage more sophisticated features are used to recognize similar characters belonging to the leaf nodes of the classification tree.

Research on Arabic and Persian OCR has primarily been focused on word recognition, and very few approaches have been proposed for text-line extraction. Since Arabic is generally written in Naskh script, text-line segmentation using horizontal projections works quite well due to large inter-line spacing [Kho02]. A similar approach is used by Jelodar et al. [JFMF05] to extract text-lines from printed Persian documents. Segmentation of a page image into individual lines by horizontal projection is a primitive approach and works only for clean, single-column documents with large inter-line spacing. To handle multi-column documents, either the x-y cut method [NSV92] is used, or morphological operations are used to get text blocks [SSMSSS06] which can then be further subdivided into individual text-lines by horizontal projection.

Over the last two decades, several layout analysis algorithms have been proposed in the literature (for a literature survey, please refer to [CCMM98, Nag00]) that work for different layouts and are quite robust to the presence of noise in the document. Many of these algorithms have come to wide-spread use for analyzing document images in different scripts. Kumar et al. [KKJ07] have evaluated the performance of six algorithms for page segmentation on Nastaliq script: the x-y cut [NSV92], the smearing algorithm [WCW82], whitespace analysis [Bai94], the constrained text-line finding algorithm [Bre02c], Docstrum [O'G93], and the Voronoi-diagram based approach [KSI98]. These algorithms work very well in segmenting documents in Roman script as shown in Chapter 4. However, when Kumar et al. applied these algorithms to segment Nastaliq script documents, none of these algorithms were able to achieve an accuracy of more than 70% on their test data which had simple book layouts with no font size variations within each page.

This chapter proposes a two-descender model of Urdu text-lines and based on this geometric model presents an algorithm for extracting Urdu text-lines from a scanned documents. This work is based on the layout analysis system described in [Bre03b] that uses globally optimal geometric algorithms, combined with robust statistical models, to model and analyze the layouts of Roman script documents. A particular advantage of this system is that it is nearly parameter-free. A detailed illustration of the presented system is given in Section 6.2. To evaluate the performance of this system in segmenting real-world documents, a dataset of Urdu documents is prepared. As a control experiment, the documents in the test set are also segmented using projection profiles. The recursive x-y cut algorithm (Section 4.2.1) is used for this purpose. The dataset, the experimental setup, and the results obtained are described in Section 6.3 followed by a conclusion in Section 6.4.

6.2 Urdu Document Layout Analysis

The document layout analysis system presented in [Bre03b] - that is adapted to Nastaliq script in this work - consists of the following main steps:

1. Empty whitespace rectangles that completely cover the page background are found.
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. Text-Lines that respect the columnar structure of the document are extracted using a geometric matching algorithm.
4. The reading order of the text-lines is determined using constraints on the geometric arrangement of text-line segments on a page.

To adapt the layout analysis system of Breuel to Nastaliq script, a text-line model is developed for Urdu text-lines written in Nastaliq script. This model is used in the geometric matching algorithm at step 3 of the layout analysis system outlined above. Then, the reading order algorithm in step 4 is modified to adapt to Urdu documents.

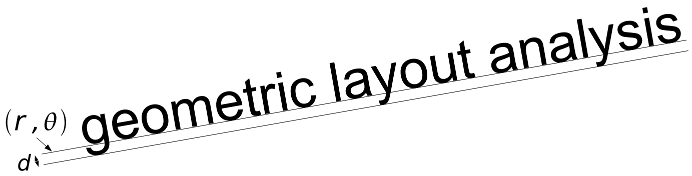


Figure 6.2: An illustration of Roman script text-line model proposed by Breuel [Bre02b]. The baseline is modeled as a straight line with parameters (r, θ), and the descender line is modeled as a line parallel to the baseline at a distance d below the baseline.

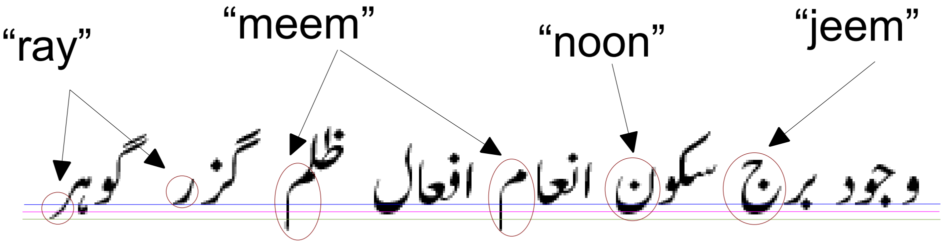


Figure 6.3: An example image showing the descender distance of different characters in Nastaliq script. The characters like "noon", "jeem" etc. make a line of descenders whether they appear alone or appear at the end of a compound character. On the other hand, characters like "meem" descend much lower than other characters. Characters like "ray" may either lie on the baseline, on first descender line, or on the second descender line depending on their position in the compound character, and other characters making the compound character.

6.2.1 Geometric Model for an Urdu Text-Line

The presented geometric text-line model for Urdu text is based on Roman script text-line model by Breuel [Bre02b]. Breuel proposed a parameterized model for a text-line with parameters (r, θ, d), where r is the distance of the baseline from the origin, θ is the angle of the baseline from the horizontal axis, and d is the distance of the line of descenders from the baseline. This model is illustrated in Figure 6.2. The advantage of explicitly modeling the line of descenders is that it removes the ambiguities in baseline detection caused by the presence of descenders.

In Nastaliq script, the descender distance of all compound and individual characters is not the same as illustrated in Figure 6.3. Some individual characters are highlighted in Figure 6.3 both in isolated form and as a part of a ligature. An important characteristic of the Nastaliq script is that a character descends below the baseline only in its isolated

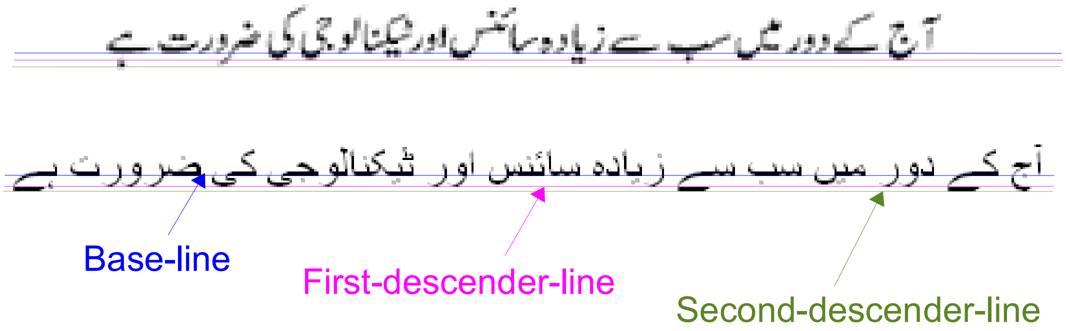

آج کے دور میں سب سے زیادہ سائنس اور ٹیکنالوجی کی ضرورت ہے
آج کے دور میں سب سے زیادہ سائنس اور ٹیکنالوجی کی ضرورت ہے
Base-line
First-descender-line
Second-descender-line

Figure 6.4: Modeling an Urdu text-line using two descender lines. The text-line at the top shows a line of text in Nastaliq script, whereas the text-line at the bottom shows the same piece of text in Naskh script. Although different characters appear at different heights in both scripts, the resulting text-lines can be accurately modeled using two descender lines in each script.

form or if it is the last character in a compound ligature. Another source of variation comes from the script, for example the character “bari yay” (the left-most character of the text-lines shown in Figure 6.4) lies on the baseline in Nastaliq script, whereas it lies on the second descender line in Naskh script. Hence the single-descender model used for Roman script is not suitable for modeling text-lines in Nastaliq script.

A careful observation of printed Urdu text reveals that characters descending from the baseline can be grouped into two categories based on their descending distance. One group including characters like “noon”, “jeem” etc. descends to a constant depth below the baseline, while another group including characters like “ray”, “wao” etc. descends to a different constant depth below the baseline. All descending characters can be assigned to one of these groups with the exception of the character “meem” which descends lower than all other characters. Therefore, a new text-line model is developed for Nastaliq script using two descenders as shown in Figure 6.4. These lines are parameterized by parameters (r, θ, d_1, d_2), where d_1 is the distance of the first descender line and d_2 is the distance of the second descender line from the baseline.

6.2.2 Extraction of Urdu Text-Lines

Breuel [Bre02b] used geometric matching to extract text-lines from scanned documents based on the text-line model described in Section 6.2.1. A quality function is defined which gives the quality of matching the text-line model to a given set of points. The goal is to find a collection of parameters (r, θ, d) for each text-line in the document image that maximizes the number of bounding boxes matching the model and that minimizes the distance of each reference point from the baseline in a robust least square sense. The RAST algorithm [Bre93] is used to find the parameters of all text-lines in a document image. The algorithm is run in a greedy fashion such that it returns text-lines in decreasing order of quality.

Consider a set of reference points $\{x_1, x_2, \cdots, x_n\}$ obtained by taking the middle of the bottom line of the bounding boxes of the connected components in a document image. The goal of text-line detection is to find the maximizing set of parameters $\vartheta = (r, \theta, d)$ with respect to the reference points $\{x_1, x_2, \cdots, x_n\}$:

$$\hat{\vartheta} := \arg\max_{\vartheta} Q_{x_1^n}(\vartheta) \tag{6.1}$$

The quality function used in [Bre02b] is:

$$Q_{x_1^n}(\vartheta) = Q_{x_1^n}(r, \theta, d) = \sum_{i=1}^{n} \max(q_{(r,\theta)}(x_i), \alpha q_{(r-d,\theta)}(x_i)) \tag{6.2}$$

where

$$q_{(r,\theta)}(x) = \max\left(0, 1 - \frac{d^2_{(r,\theta)}(x)}{\epsilon^2}\right) \tag{6.3}$$

The first term in the summation of Equation 6.2 calculates the contribution of a reference point x_i to baseline, whereas the second term calculates the contribution of a reference point x_i to the descender line. Since a point can either lie on the baseline or the descender line, maximum of the two contributions is taken in the summation. Typically the value of α is set to 0.75, and its role is to compensate for the inequality of priors for baseline and descender such that a reference point has more chances to match with the baseline as compared to the descender line. The contribution of a reference point to a line is measured using Equation 6.3 and its value lies in the interval $[0, 1]$. The contribution $q_{(r,\theta)}(x)$ is zero for all reference points for which $d_{(r,\theta)}(x) \geq \epsilon$. These points are considered as outliers and hence do not belong to the line with parameters (r, θ). In practice, $\epsilon = 5$ proves to be a good choice for documents scanned at 300-dpi. The contribution $q_{(r,\theta)}(x) = 1$ if $d_{(r,\theta)}(x) = 0$ which means the contribution of a point to a line is one if and only if the point lies exactly on the line.

The RAST algorithm is used to extract the text-line with maximum quality as given by Equation 6.1. Then all reference points that contributed with a non-zero quality to the extract text-line are removed from the list of reference points and the algorithm is run again. In this way, the algorithm returns text-lines in decreasing order of quality until all text-lines have been extracted from the document image.

To adapt the quality function in Equation 6.2 to the extraction of Urdu text-lines, the contribution of a reference point to both descender lines is used:

$$Q_{x_1^n}(\vartheta) = Q_{x_1^n}(r, \theta, d_1, d_2) = \sum_{i=1}^{n} \max(q_{(r,\theta)}(x_i), \alpha q_{(r-d_1,\theta)}(x_i), \alpha q_{(r-d_2,\theta)}(x_i)) \tag{6.4}$$

The local quality function $q_{(r,\theta)}(x)$ is the same as in Equation 6.3. The quality function in Equation 6.4 can be maximized in Equation 6.1 to extract Urdu text-lines from a scanned document.

6.2.3 Reading Order Determination

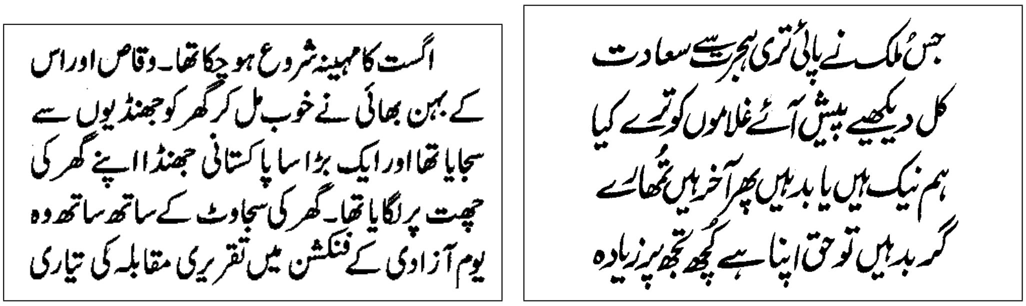
Reading direction of Urdu text is from right to left, which is opposite to the Roman script. Therefore the algorithm for determining the reading order in [Bre03b] is modified as follows:

1. Text-Line a comes before text-line b if their ranges of x-coordinates overlap and if text-line a is above text-line b on the page.
2. Text-Line a comes before text-line b if a is entirely to the *right* of b and if there does not exist a text-line c whose y-coordinates are between a and b and whose range of x-coordinates overlaps both a and b.

Applying these two ordering criteria is sufficient to define a partial order of text-line segments in a document. This partial order is then extended to a total order of all elements using a topological sorting algorithm [CLR90].

6.3 Experiments, Results, and Discussion

In order to evaluate the performance of the described layout analysis system, 25 Urdu documents from different sources were scanned. The scanned images are categorized into five classes: *book*, *poetry*, *digest*, *magazine*, and *newspaper*. There are 5 images of each class in the dataset. The dataset is made publicly available and can be downloaded freely from http://www.iupr.org/demos_downloads/. The Urdu documents that are in common use today are both typeset documents and documents handwritten by calligraphers. Samples of typeset and handwritten Urdu text are shown in Figure 6.5. Due to



اگست کا مہینہ شروع ہو چکا تھا۔ وقاص اور اس
کے بہن بھائی نے خوب مل کر گھر کو جھنڈیوں سے
سجایا تھا اور ایک بڑا سا پاکستانی جھنڈا اپنے گھر کی
چھت پر لگایا تھا۔ گھر کی سجاوٹ کے ساتھ ساتھ وہ
یوم آزادی کے فنکشن میں تقریری مقابلہ کی تیاری

(a) Typeset Urdu text

جس ملک نے پائی تری ہجرت سے سعادت
کل دیکھیے پیش آئے غلاموں کو ترے کیا
ہم نیک ہیں یا بد ہیں پھر آخر ہیں تمہارے
گر بد ہیں تو حق اپنا ہے کچھ تجھ پر زیادہ

(b) Urdu text written by a Calligrapher

Figure 6.5: An example of typeset and handwritten Urdu text. The text written by calligraphers closely resembles typeset text and shares the same properties.

the complex nature of Nastaliq script and the availability of calligraphers, the first Urdu typesetter was developed as late as early 1980s [Wik]. Hence to date many printed books are available only in handwritten form. As a representative sample of Urdu text, the documents in the *book* and *poetry* categories are scanned from text written by calligraphers, whereas the documents in the *digest*, and *magazine* categories are scanned from typeset sources. The *newspaper* documents contain both handwritten text by calligraphers and typeset text in the same page. Important headlines are written by calligraphers in large text, whereas the news text itself is typeset. Although the dataset is quite small in size, it contains many types of layouts. Hence it can be used to evaluate the performance of a layout analysis algorithm for Urdu documents. However, for a thorough evaluation, a larger dataset may be required.

The evaluation of the algorithm is done in two parts: the first part analyzes the errors made in text-line detection and the second part evaluates the reading order algorithm.

6.3.1 Text-Line Detection Accuracy

The text-line detection accuracy measures how many text-lines were correctly detected. A text-line is said to be **correctly detected** if it does not fall into any of the three types of error defined below.

Oversegmented text-lines: the number of text-lines that are either split into more than one text-line, or partially detected.

Undersegmented text-lines: the number of text-lines merged with some other text-line.

Missed text-lines: the number of text-lines that were not found by the algorithm.

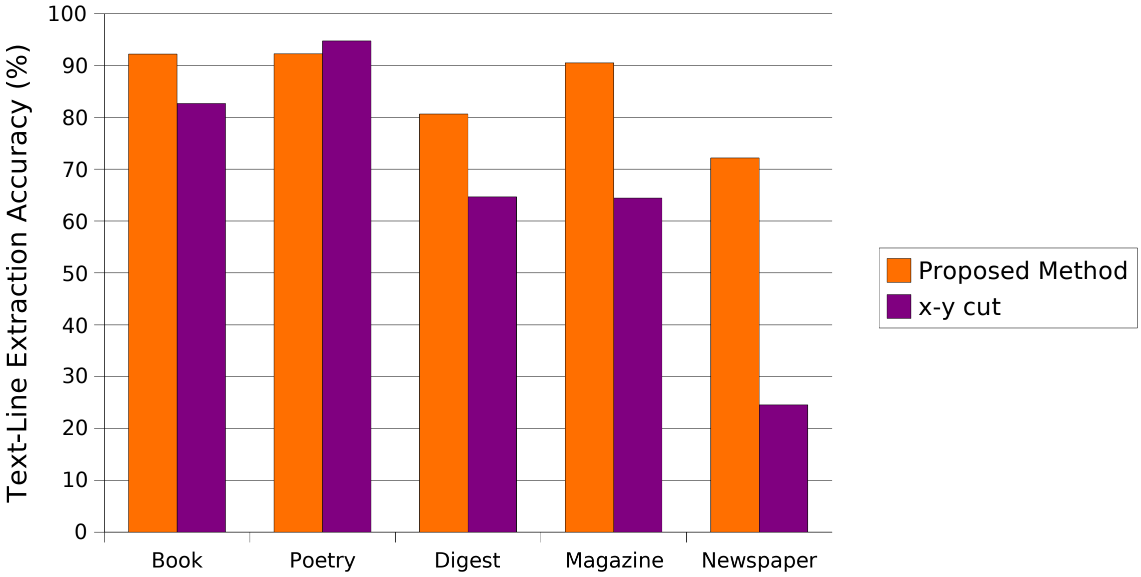


Figure 6.6: A comparison of text-line segmentation accuracies of the proposed technique and the x-y cut algorithm.

The documents in the dataset were segmented using the x-y cut algorithm and the text-line extraction algorithm described in Section 6.2. The x-y cut algorithm was chosen as a representative state-of-the-art algorithm for segmenting Nastaliq script documents. A comparison of the percentage of correctly segmented text-lines by both methods is shown in Figure 6.6.

Horizontal projection of sample paragraphs in Nastaliq script is shown in Figure 6.7. In both paragraphs, there are no zero-valleys in the projection profile. To segment Urdu text with x-y cut, noise thresholds are set to a high value so that the algorithm can find the main body of a text-line. The parts of a text-line cut out at this stage can be assigned to the text-line by simple post-processing steps. Hence, text-lines as shown in Figure 6.8 are considered correctly segmented. The results of segmentation using the x-y cut algorithm are shown in Table 6.1. The values of the parameters for the x-y cut algorithm used were: $t_x = 30, t_y = 2, t_x^n = 20, t_y^n = 250$ (see Section 4.2.1 for an explanation of these parameters).

The results show that the algorithm is able to segment *book*, and *poetry* documents with high accuracy owing to relatively large inter-line spacing. However, it fails to segment *digest*, *magazine*, and *newspaper* layouts due to smaller inter-line gaps and presence of multiple columns.

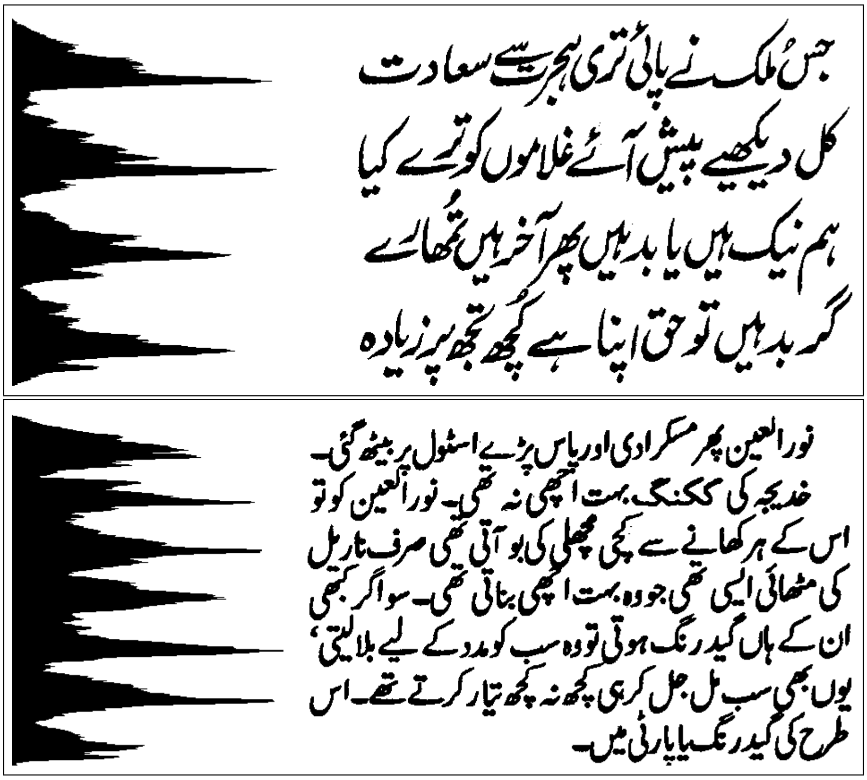


Figure 6.7: Horizontal projection of Urdu text samples. The top figure shows the case of larger inter-line spacing, the bottom figure shows the case of small inter-line spacing. Note that in both cases, there are no between-line zero-valleys in the projection profile.

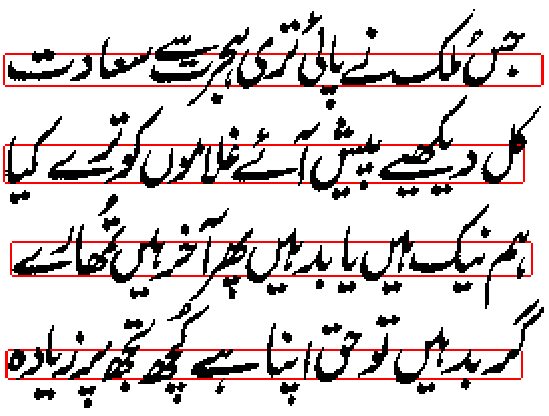

جس ملک نے پائی تری ہجرت سے سعادت
کل دیکھیے پیش آئے غلاموں کو ترے کیا
ہم نیک ہیں یا بد ہیں پھر آخر ہیں تمھارے
گر بد ہیں تو حق اپنا ہے کچھ تجھ پہ زیادہ

Figure 6.8: Results of segmenting an Urdu piece of text using the x-y cut algorithm. The resulting text-lines are considered correct as the body of each text-line is correctly identified. The ascenders and descenders can be assigned to the text-line body using simple post-processing steps.

Table 6.1: The performance of the x-y cut algorithm on the Urdu documents dataset.

Document Type ($n =$)	Correctly Detected Text-Lines (%)	Over-Segmented Text-Lines (%)	Under-Segmented Text-Lines (%)	Missed Text-Lines (%)
Book (231)	82.68	5.19	6.93	5.19
Poetry (284)	94.72	1.76	0.00	3.52
Digest (702)	64.67	0.57	29.91	4.84
Magazine (1156)	64.45	0.17	33.82	1.56
Newspaper (819)	24.54	0.85	60.2	2.20

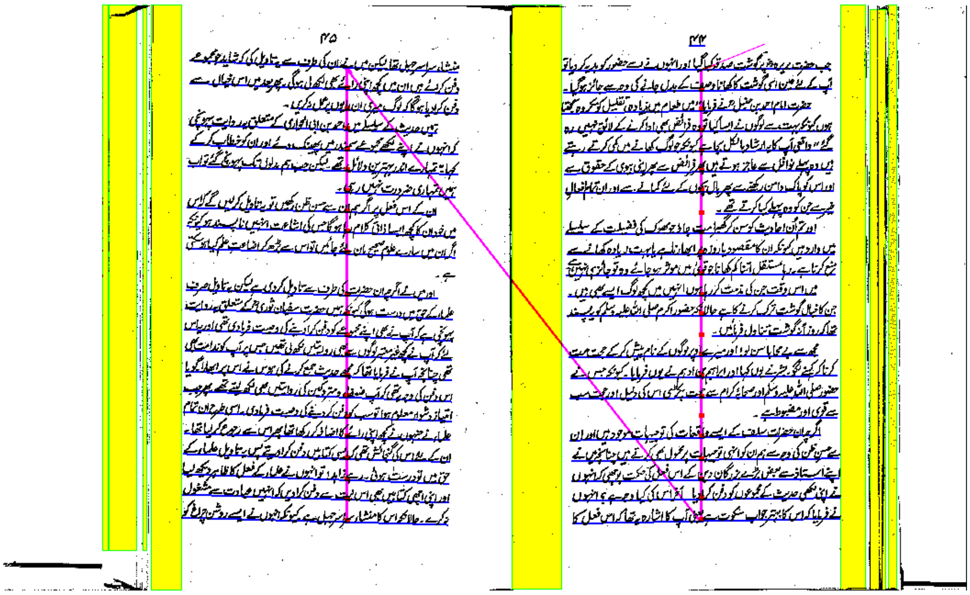
Table 6.2: The performance of the presented text-line extraction algorithm on each category of the dataset.

Document Type ($n =$)	Correctly Detected Text-Lines (%)	Over-Segmented Text-Lines (%)	Under-Segmented Text-Lines (%)	Missed Text-Lines (%)
Book (231)	92.21	5.63	0.00	2.16
Poetry (284)	92.25	4.58	0.00	3.17
Digest (702)	80.63	11.54	0.00	7.83
Magazine (1156)	90.48	4.15	0.87	4.33
Newspaper (819)	72.16	7.81	4.15	15.87

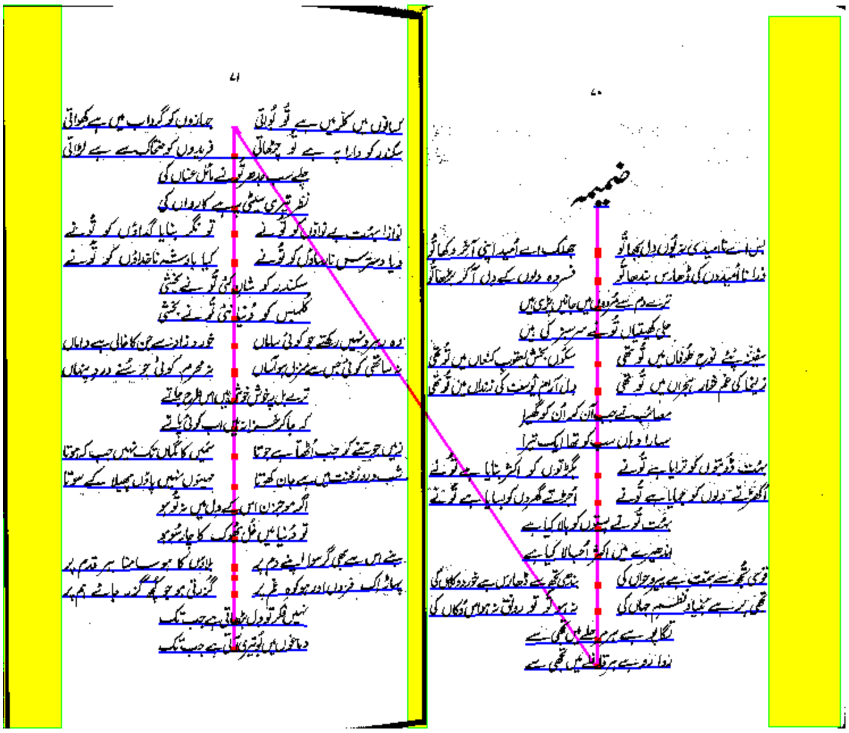
Table 6.2 summarizes the results obtained by applying the presented layout analysis algorithm to the test data, and manually inspecting the resulting text-lines detected by the system. Example images showing the result of the algorithm are shown in Figure 6.9 and 6.10. Note that the presented system achieved better results than the x-y cut approach on all categories of documents except poetry documents. The presented algorithm made more split errors than the x-y cut algorithms. These errors appeared as a result of partially detected text-lines in which the first or the last word of the line was not vertically aligned with the rest of the text-line and hence was ignored by the text-line extraction algorithm.

Based on the results presented in Table 6.2, the following observations can be made:

- The algorithm works quite well on documents in the *book*, *magazine*, and *poetry* categories. A few number of missed error are those in which the text-line consisted of only one connected component as the text-line detection algorithm needs at least two components to make a line. The over-segmentation errors occur due to page curl in some documents.
- In the *digest* layout, many over-segmentation errors occur due to enumerated lists, in which the enumerator symbol is segmented separately.



(a) Facing pages of a prose book



(b) Facing pages of a poetry book

Figure 6.9: Example images illustrating results of the proposed layout analysis system on a book and a poetry page. The yellow rectangles show the detected vertical gutters. Thin horizontal blue lines indicate detected text-line segments, and the thick magenta lines running down and diagonally across the image indicate reading order.

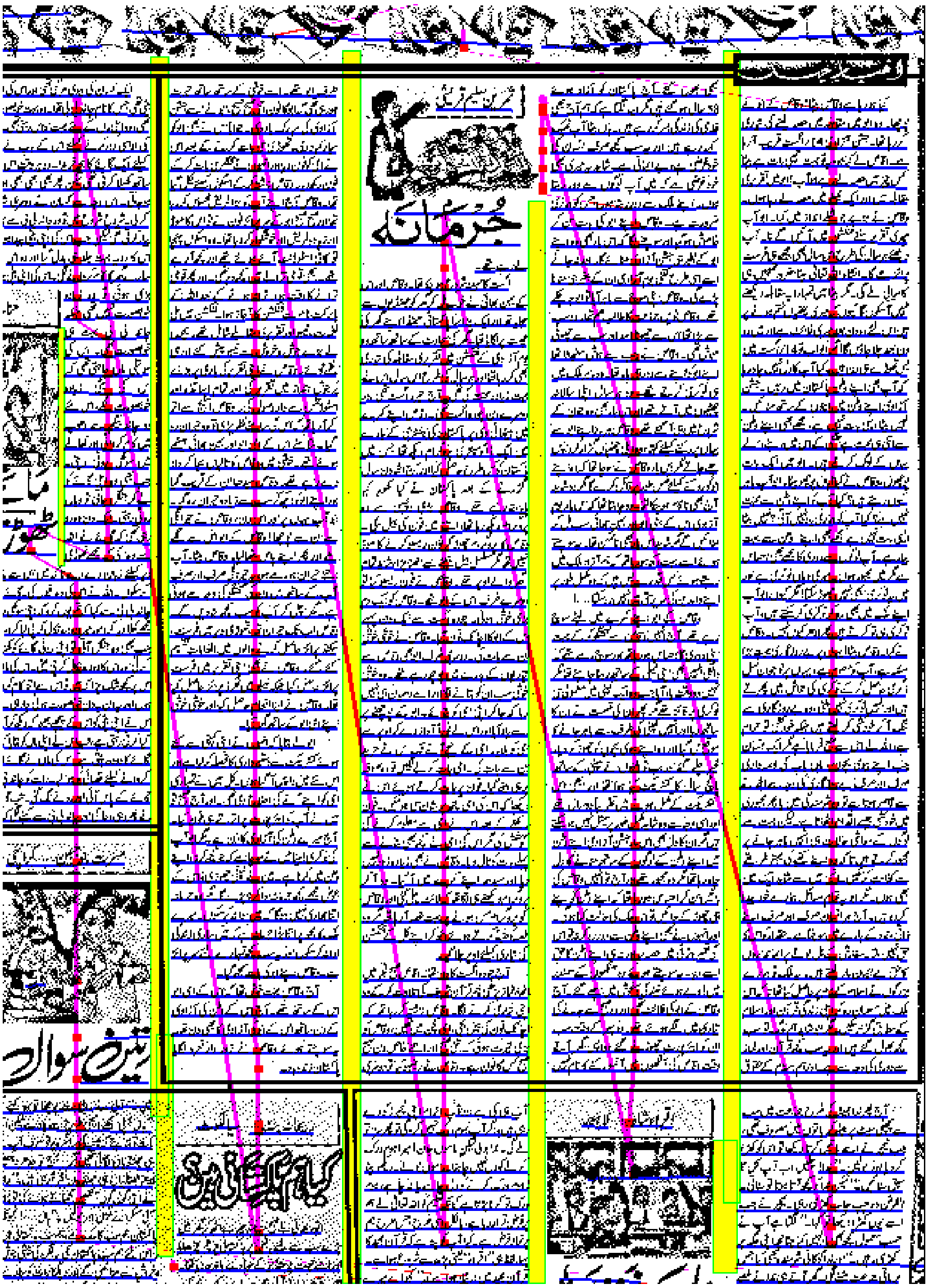
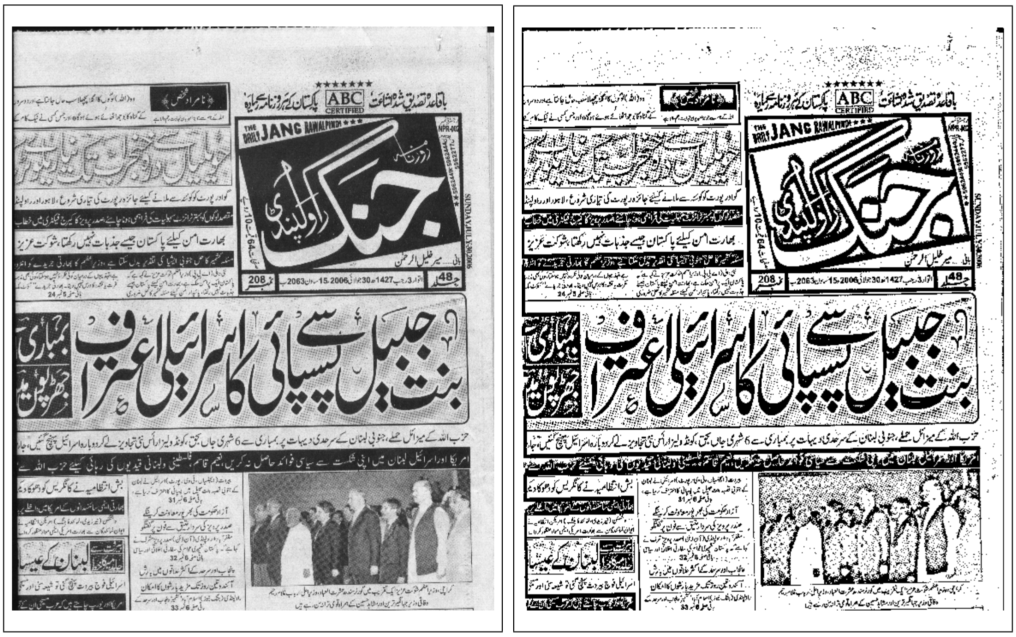


Figure 6.10: Example image illustrating results of the proposed layout analysis system on a magazine page. Note that images and graphics were not removed, so they result in some spurious text-lines.



ABC CERTIFIED

THE DAILY JANG RAWALPINDI

جنگ راولپنڈی

SUNDAY, JULY 30, 2006

بانی...... میر خلیل الرحمٰن

جلد 48

نمبر 208

اتوار 3 رجب 1427ھ 30 جولائی 2006ء 15 ساون 2063ب

گوادر پورٹ کو کوئٹہ سے ملانے کیلئے جائزہ رپورٹ کی تیاری شروع، لاہور اور راولپنڈ

بھارت امن کیلئے پاکستان جیسے جذبات نہیں رکھتا، شوکت عزیز

مسئلہ کشمیر کا حل جنوبی ایشیا کی تقدیر بدل سکتا ہے، وزیراعظم کا بھارتی جریدے کو انٹرو

بنت جبیل سے پسپائی کا اسرائیلی اعتراف

حزب اللہ کے میزائل حملے، جنوبی لبنان کے سرحدی دیہات پر بمباری سے 6 شہری جاں بحق، کونڈولیزا رائس نئی تجاویز لے کر دوبارہ اسرائیل پہنچ گئیں

امریکا اور اسرائیل لبنان میں اپنی شکست سے سیاسی فوائد حاصل نہ کریں، نعیم قاسم، فلسطینی و لبنانی قیدیوں کی رہائی کیلئے حزب اللہ

بش انتظامیہ نے کانگریس کو دھوکا دیا

Figure 6.11: An example image cropped from the front page of a newspaper containing both normal and inverted text in the same image. The binarization step fails to identify the inverted text and makes it a part of the page background (white) as shown in the binarized image. (cp. [SuHKB06])

- In the *newspaper* layout, many text-lines are missed. This is due to very small inter-line spacing between two consecutive text-lines. Due to this small spacing, connected components from two neighboring text-lines sometimes merge together. In such a case the upper text-line is missed by the algorithm. Another source of a missed text-line error is the presence of inverted text resulting in poor binarization of the image as shown in Figure 6.11. A number of false alarms also appear in these layouts due to non-text elements on the page.

6.3.2 Reading Order Determination Accuracy

The performance of the reading order determination algorithm heavily depends on the text-line detection accuracy. Manual inspection of the results showed the following types of errors:

1. If two text-lines from different text columns are merged, they are interpreted as a separator and hence the algorithm gives a wrong reading order.
2. In some cases, the separation between different sections of a multi-column document is not represented by a text-line spanning along the columns, but instead a ruling

(thick horizontal black line) is used. In that case the algorithm fails to detect the start of a new section.

3. The reading order of Urdu poetry in two-column layout is not handled by the current algorithm.

6.3.3 Problems and Future Directions

Based on the experimental results presented in this chapter, the described layout analysis system can possibly be improved in the following ways:

1. Image statistics like x-height (height of a lower case letter 'x') estimation play an important role in automatically adjusting parameters of the algorithm to the resolution and dominant font size of the input document. Incorporating a statistical analysis for Urdu documents to estimate dominant font size may reduce the number of missed lines.

2. If the first or the last word of a text-line is not aligned with the rest of the line, the word is ignored resulting in a split error. This happens if the last character of the word is "meem" which descends lower than both descender lines. Image statistics can be used to solve this problem in a post-processing step.

3. Using a binarization technique that can handle both normal and inverted text on the same image will result in improved performance on the *newspaper* images.

4. A large dataset of scanned Urdu documents with ground-truth should be collected to obtain a more reliable estimate of the performance of the presented layout analysis system.

5. If the skew of a scanned document is so large that axis-aligned gutters cannot be found, an algorithm to extract whitespace rectangles at arbitrary orientations [Bre03a] can be used.

6. The reading order algorithm cannot handle Urdu poetry written in two column format, as it is misinterpreted as a two-column text. Hence a different algorithm must be used to analyze poetry documents in two-column format.

6.4 Summary

This chapter presented a high-performance layout analysis system for Urdu documents. A well-known layout analysis system for Roman script was adapted to analyze Urdu documents. The described system was tested on scanned Urdu documents from different sources like books, novels, magazines, and newspapers. The algorithm achieved an accuracy of above 90% in terms of text-line detection for books, poetry, and magazine images. The accuracy decreased to about 80% for documents from digest class due to small inter-line spacing and presence of enumerated lists. Newspaper documents proved to be the toughest class presenting several challenges like the presence of many font sizes within the same image, small inter-line spacing, inverted text, and poor quality of page resulting in lot of noise. The text-line detection accuracy for the newspaper images was around 72%.

Chapter 7

Statistical Layout Analysis

Current layout analysis methods can handle a variety of simple document layouts with high accuracy as demonstrated in Chapter 4. A key limitation of these methods is that they rely on many hard-coded assumptions about document layouts and can not adapt to new layouts for which the underlying assumptions are not satisfied. Another major drawback of these approaches is that they do not return confidence scores for their outputs. These problems pose major challenges in large scale digitization efforts where a large number of different layouts need to be handled and manual inspection of the results on each individual page is not feasible. This chapter presents a statistical approach to layout analysis that solves the above-mentioned problems. The contributions of this chapter to the state-of-the-art in layout analysis are:

1. A new approach to model known page layouts as a structural mixture model is presented.
2. A probabilistic matching algorithm is presented that gives multiple interpretations of input layout with associated probabilities.
3. An algorithm based on A* search is presented for finding the most likely layout of a page, given its structural layout model.
4. An interactive graphical user interface (GUI) is developed that enables the user to quickly build structural layout models.
5. An EM-like training algorithm is presented that is capable of learning the geometric variability of structural layout models from training data *without* the need for a page segmentation ground-truth.

7.1 Introduction and Related Work

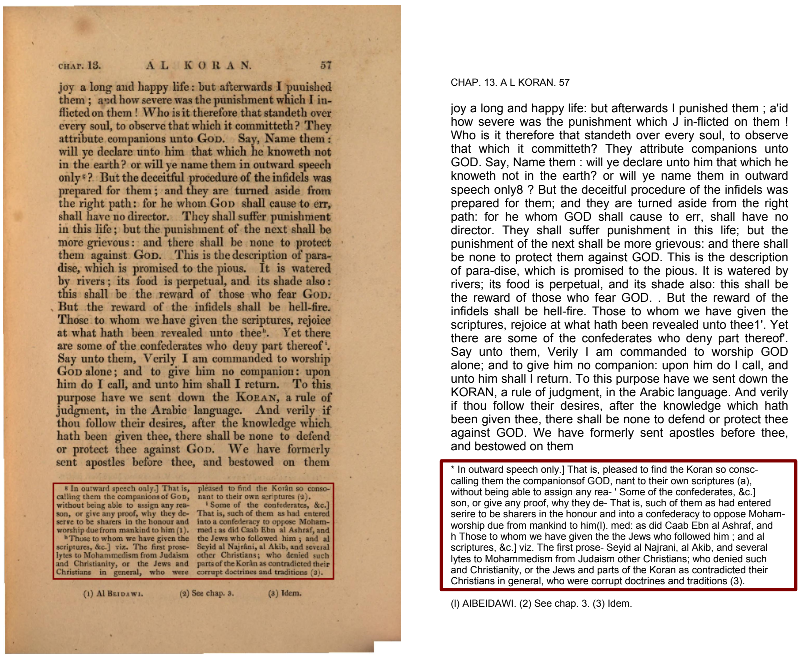
A number of different approaches for geometric layout analysis of scanned documents were presented and evaluated in Chapter 4. These approaches have shown to work quite well on the UW-III dataset which consists of scanned journal articles. A summary of the text-line detection error rates for these approaches is presented in Table 4.7. The table shows that Docstrum, Voronoi, and Whitespace-Cuts algorithms achieve very low error rates on structured journal articles in the UW-III dataset. These algorithms rely on many hard-coded assumptions about document layouts. For instance, a common assumption is that larger structural divisions inside a document are indicated by larger amounts of whitespace. This assumption holds for many simple document layouts, but it may break for more complex or non-stereotypical layouts.

An example of a non-stereotypical layout from the Google 1000 books dataset is shown in Figure 7.1. The dataset was released by Google Inc. in September 2007 and contains 1000 scanned books with hOCR-format [Bre07] ground-truth. It contains scans of old books for which copyrights have expired. Therefore, most of the books have simple one-column page layouts. The ground-truth was generated by an anonymous OCR software. The example image in Figure 7.1 is from a book containing mixed one-two column layout. As shown in the figure, the OCR software failed to segment the page correctly and merged the text from the two columns of the page. The results of applying the state-of-the-art page segmentation algorithms to this image are shown in Figure 7.2. Interestingly, none of the algorithms was able to segment the two-column part of the page correctly.

A closer look at the example image reveals that the gap between the two text columns is not larger than the global inter-word spacing of the document. Hence the basic assumption of most of the research algorithms and commercial systems - that larger structural divisions inside a document are indicated by larger amounts of whitespace - does not hold for this layout. This results in incorrect segmentation of the page both by research algorithms and commercial systems.

Commercial systems address this problem by providing a GUI with which the user can manually define the layout of the page. One such example is shown in Figure 7.3. This solution is suitable for daily office use when a small number of documents have to be dealt with. However, it does not fit the needs for large scale digitization tasks since the user has to manually fix the results for all incorrectly handled pages, which is not feasible due to the scale of the problem.

Research algorithms, on the other hand, usually present several parameters that can



CHAP. 13. A L K O R A N. 57

joy a long and happy life: but afterwards I punished them; and how severe was the punishment which I inflicted on them! Who is it therefore that standeth over every soul, to observe that which it committeth? They attribute companions unto GOD. Say, Name them: will ye declare unto him that which he knoweth not in the earth? or will ye name them in outward speech only[g]? But the deceitful procedure of the infidels was prepared for them; and they are turned aside from the right path: for he whom GOD shall cause to err, shall have no director. They shall suffer punishment in this life; but the punishment of the next shall be more grievous: and there shall be none to protect them against GOD. This is the description of paradise, which is promised to the pious. It is watered by rivers; its food is perpetual, and its shade also: this shall be the reward of those who fear GOD. But the reward of the infidels shall be hell-fire. Those to whom we have given the scriptures, rejoice at what hath been revealed unto thee[h]. Yet there are some of the confederates who deny part thereof[i]. Say unto them, Verily I am commanded to worship GOD alone; and to give him no companion: upon him do I call, and unto him shall I return. To this purpose have we sent down the KORAN, a rule of judgment, in the Arabic language. And verily if thou follow their desires, after the knowledge which hath been given thee, there shall be none to defend or protect thee against GOD. We have formerly sent apostles before thee, and bestowed on them

[g] In outward speech only.] That is, calling them the companions of GOD, without being able to assign any reason, or give any proof, why they deserve to be sharers in the honour and worship due from mankind to him (1).

[h] Those to whom we have given the scriptures, &c.] viz. The first proselytes to Mohammedism from Judaism and Christianity, or the Jews and Christians in general, who were pleased to find the Korân so consonant to their own scriptures (2).

[i] Some of the confederates, &c.] That is, such of them as had entered into a confederacy to oppose Mohammed: as did Caab Ebn al Ashraf, and the Jews who followed him; and al Seyid al Najrâni, al Akib, and several other Christians; who denied such parts of the Korân as contradicted their corrupt doctrines and traditions (3).

(1) Al BEIDAWI. (2) See chap. 3. (3) Idem.

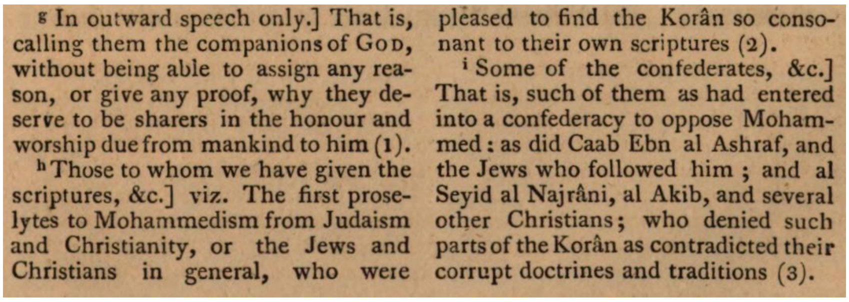
CHAP. 13. A L KORAN. 57

joy a long and happy life: but afterwards I punished them ; a'id how severe was the punishment which J in-flicted on them ! Who is it therefore that standeth over every soul, to observe that which it committeth? They attribute companions unto GOD. Say, Name them : will ye declare unto him that which he knoweth not in the earth? or will ye name them in outward speech only8 ? But the deceitful procedure of the infidels was prepared for them; and they are turned aside from the right path: for he whom GOD shall cause to err, shall have no director. They shall suffer punishment in this life; but the punishment of the next shall be more grievous: and there shall be none to protect them against GOD. This is the description of para-dise, which is promised to the pious. It is watered by rivers; its food is perpetual, and its shade also: this shall be the reward of those who fear GOD. . But the reward of the infidels shall be hell-fire. Those to whom we have given the scriptures, rejoice at what hath been revealed unto thee1'. Yet there are some of the confederates who deny part thereof'. Say unto them, Verily I am commanded to worship GOD alone; and to give him no companion: upon him do I call, and unto him shall I return. To this purpose have we sent down the KORAN, a rule of judgment, in the Arabic language. And verily if thou follow their desires, after the knowledge which hath been given thee, there shall be none to defend or protect thee against GOD. We have formerly sent apostles before thee, and bestowed on them

* In outward speech only.] That is, pleased to find the Koran so consc-
calling them the companionsof GOD, nant to their own scriptures (a),
without being able to assign any rea- ' Some of the confederates, &c.]
son, or give any proof, why they de- That is, such of them as had entered
serire to be sharers in the honour and into a confederacy to oppose Moham-
worship due from mankind to him(l). med: as did Caab Ebn al Ashraf, and
h Those to whom we have given the the Jews who followed him ; and al
scriptures, &c.] viz. The first prose- Seyid al Najrani, al Akib, and several
lytes to Mohammedism from Judaism other Christians; who denied such
and Christianity, or the Jews and parts of the Koran as contradicted their
Christians in general, who were corrupt doctrines and traditions (3).

(l) AIBEIDAWI. (2) See chap. 3. (3) Idem.

(a) sample page (b) text ground-truth

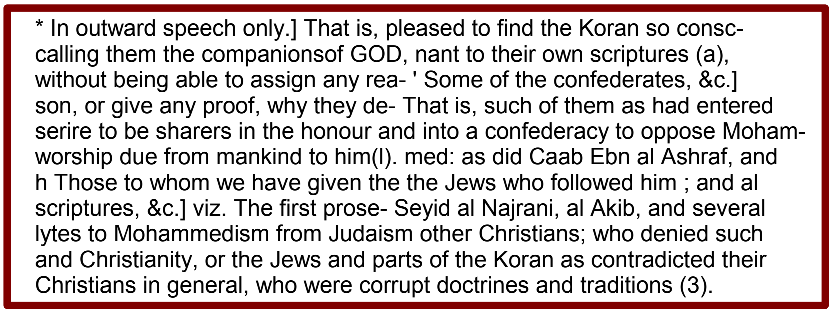


[g] In outward speech only.] That is, calling them the companions of GOD, without being able to assign any reason, or give any proof, why they deserve to be sharers in the honour and worship due from mankind to him (1).

[h] Those to whom we have given the scriptures, &c.] viz. The first proselytes to Mohammedism from Judaism and Christianity, or the Jews and Christians in general, who were pleased to find the Korân so consonant to their own scriptures (2).

[i] Some of the confederates, &c.] That is, such of them as had entered into a confederacy to oppose Mohammed: as did Caab Ebn al Ashraf, and the Jews who followed him; and al Seyid al Najrâni, al Akib, and several other Christians; who denied such parts of the Korân as contradicted their corrupt doctrines and traditions (3).

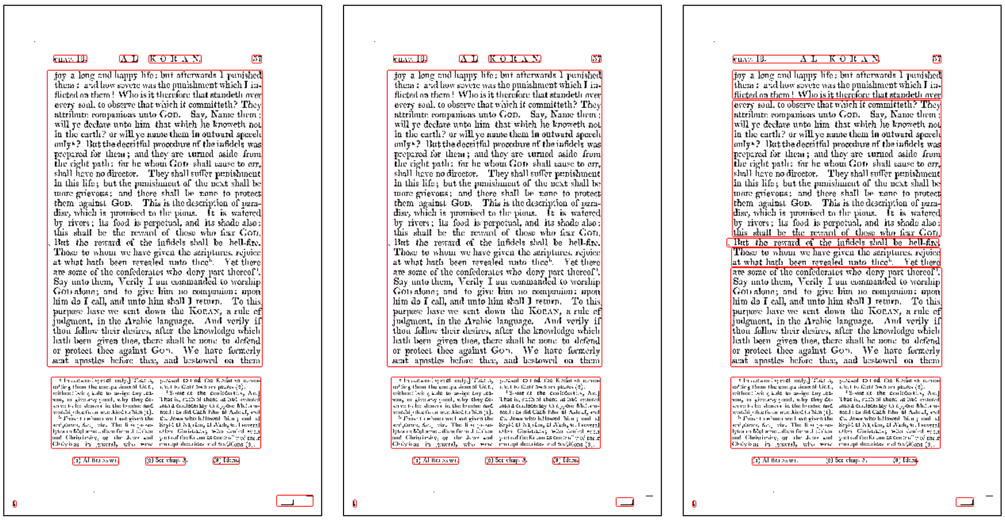
(c) cropped two-column part of the sample page



* In outward speech only.] That is, pleased to find the Koran so consc-
calling them the companionsof GOD, nant to their own scriptures (a),
without being able to assign any rea- ' Some of the confederates, &c.]
son, or give any proof, why they de- That is, such of them as had entered
serire to be sharers in the honour and into a confederacy to oppose Moham-
worship due from mankind to him(l). med: as did Caab Ebn al Ashraf, and
h Those to whom we have given the the Jews who followed him ; and al
scriptures, &c.] viz. The first prose- Seyid al Najrani, al Akib, and several
lytes to Mohammedism from Judaism other Christians; who denied such
and Christianity, or the Jews and parts of the Koran as contradicted their
Christians in general, who were corrupt doctrines and traditions (3).

(d) text ground-truth of the cropped two-column part

Figure 7.1: An example image from Google 1000 books dataset with its ASCII text ground-truth. The software that was used to extract the ground-truth text failed to find the column separator in the two-column part of the page and merged the text across the two columns.



CHAP. 13. AL KORAN. 57

joy a long and happy life; but afterwards I punished them: and how severe was the punishment which I inflicted on them! Who is it therefore that standeth over every soul, to observe that which it committeth? They attribute companions unto God. Say, Name them; will ye declare unto him that which he knoweth not in the earth? or will ye name them in outward speech only[a]? But the deceitful procedure of the infidels was prepared for them; and they are turned aside from the right path: for he whom God shall cause to err, shall have no director. They shall suffer punishment in this life; but the punishment of the next shall be more grievous: and there shall be none to protect them against God. This is the description of paradise, which is promised to the pious. It is watered by rivers; its food is perpetual, and its shade also: this shall be the reward of those who fear God. But the reward of the infidels shall be hell-fire. Those to whom we have given the scriptures rejoice at what hath been revealed unto thee[b]. Yet there are some of the confederates who deny part thereof[c]. Say unto them, Verily I am commanded to worship God alone; and to give him no companion: upon him do I call, and unto him shall I return. To this purpose have we sent down the Koran, a rule of judgment, in the Arabic language. And verily if thou follow their desires, after the knowledge which hath been given thee, there shall be none to defend or protect thee against God. We have formerly sent apostles before thee, and bestowed on them

(1) Al Beidawi. (2) See chap. 3. (3) Idem.

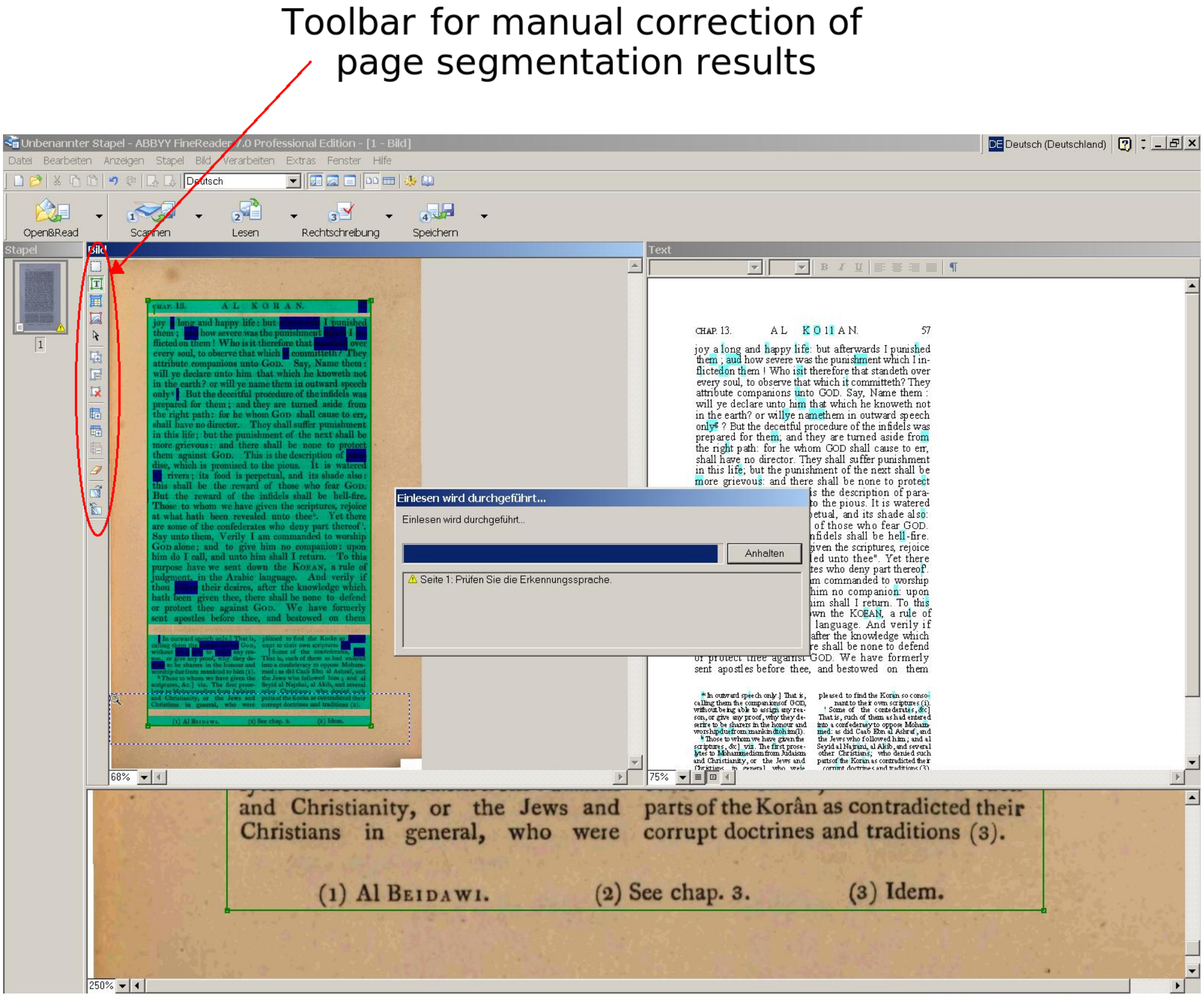
(a) Docstrum (b) Voronoi (c) Whitespace-Cuts

Figure 7.2: Segmentation results of applying state-of-the-art page segmentation algorithms on the example image in Figure 7.1. None of the algorithms segmented the page correctly.

be tuned to correctly segment the target document. However, parameter tuning is not trivial for most of the page segmentation algorithms especially for an end-user. Manually refining parameters of an algorithm requires an in-depth understanding of the algorithm. Automatic parameter tuning using some objective function, on the other hand, requires a large amount of labeled training data. Finally, the assumptions made by an algorithm might prohibit it altogether to segment a particular layout correctly.

Another major issue in large scale digitization projects is to find the documents on which the OCR software failed so that these can be presented to the operator for manual correction. The state-of-the-art layout analysis algorithms and commercial software do not give any confidence of their output. Hence the user has no clue when an algorithm fails to segment a page until he takes a look at the segmentation result of the algorithm. Manual inspection of the results of a segmentation algorithm for each scanned image becomes prohibitive in large scale applications where hundreds of thousands of pages are involved.

This chapter presents a statistical approach to layout analysis aimed at solving these problems. A statistical layout analysis system is based on statistical modeling of layouts. These layout models can be learned from training data and hence can be adapted to segment non-stereotypical layouts. Secondly, the use of statistical layout models to segment a page allows to get the probability of a performed segmentation. This probability can


Toolbar for manual correction of page segmentation results
Unbenannter Stapel - ABBYY FineReader 7.0 Professional Edition - [1 - Bild]
Deutsch (Deutschland)
Datei Bearbeiten Anzeigen Stapel Bild Verarbeiten Extras Fenster Hilfe
Deutsch
Open&Read
Scannen
Lesen
Rechtschreibung
Speichern
Stapel
Bild
Text
68%
75%
250%

Einlesen wird durchgeführt...

Einlesen wird durchgeführt...

Anhalten

Seite 1: Prüfen Sie die Erkennungssprache.

CHAP. 13. AL KO11AN. 57

joy a long and happy life: but afterwards I punished them ; aud how severe was the punishment which I inflictedon them ! Who isit therefore that standeth over every soul, to observe that which it committeth? They attribute companions unto GOD. Say, Name them : will ye declare unto him that which he knoweth not in the earth? or willye namethem in outward speech only? ? But the deceitful procedure of the infidels was prepared for them; and they are turned aside from the right path: for he whom GOD shall cause to err, shall have no director. They shall suffer punishment in this life; but the punishment of the next shall be more grievous: and there shall be none to protect
is the description of para-
to the pious. It is watered
etual, and its shade also:
of those who fear GOD.
nfidels shall be hell-fire.
given the scriptures, rejoice
led unto thee". Yet there
tes who deny part thereof.
m commanded to worship
him no companion: upon
im shall I return. To this
wn the KOEAN, a rule of
language. And verily if
after the knowledge which
re shall be none to defend
or protect thee against GOD. We have formerly sent apostles before thee, and bestowed on them

* In outward speech only.] That is, calling them the companionsof GOD, without being able to assign any reason, or give any proof, why they deserire to be sharers in the honour and worshipduefrom mankindtohim(l).

b Those to whom we have given the scriptures, &c] viz. The first proselytes to Mohammedism from Judaism and Christianity, or the Jews and

pleased to find the Koran so consonant to their own scriptures (i).

' Some of the confederates, &c] That is, such of them as had entered into a confederacy to oppose Mohammed: as did Caab Ebn al Ashraf, and the Jews who followed him; and al Seyid al Najrani, al Akib, and several other Christians; who denied such partsof the Koran as contradicted their

and Christianity, or the Jews and Christians in general, who were

parts of the Korân as contradicted their corrupt doctrines and traditions (3).

(1) Al Beidawi. (2) See chap. 3. (3) Idem.

Figure 7.3: A screen-shot of processing the example document image with ABBYY FineReader 7.0 Professional Edition. Note that FineReader also fails to detect the two columns in the page. The GUI has a toolbox with which user can manually correct page segmentation errors.

be used as a confidence value of the output of the algorithm. Hence, the user can look at the segmentation results of only those documents for which the statistical layout analysis algorithm gives a low probability.

Some attempts to build a trainable layout analysis system have been carried out in the past. One of the first attempts in this direction was made by Gary Kopec et al. [KC94, KK96]. They presented a communication theory approach to document recognition and called it "document image decoding". The key idea of their approach is to view document recognition as consisting of three elements: an image generator, a noisy channel and an image decoder. A document image generator is a Markov source that combines a message source with an imager that converts a one-dimensional message string into an ideal two-dimensional bitmap. The channel transforms the ideal image into a noisy observed image. The decoder estimates the message, given the observed image, by finding the a posteriori most probable path through the combined source and channel models using a Viterbi-like dynamic programming algorithm. The approach was used for direct recognition of text from scanned single-column parts of telephone yellow pages. However, this approach could not come to wide-spread use because it can not handle multi-column layouts.

Most of the efforts made by other researchers towards the development of trainable layout analysis systems have focused on the use of probabilistic grammars, owing to the great success of grammatical modeling in several areas of computer science like natural language processing. Tokuyasu et al. [TC01], inspired by the document image decoding view of Kopec et al., presented a turbo recognition approach for statistical layout analysis. Their key idea in turbo recognition is to use two stochastic regular grammars to describe horizontal and vertical page structure simultaneously, thereby reducing the overall complexity of the system. Kanungo et al. [KM03] present a segmentation algorithm that models the physical structure of a document as a hierarchical structure. Each node in the document hierarchy describes a region of the document using a stochastic regular grammar. The key insight of their approach is to let the user specify the exact form of the hierarchy and the stochastic language. The latest development in the domain of grammatical modeling of document layouts is by Shilman et al. [SLV05]. They use a discriminative grammar to model page layout instead of generative grammars as used in previous work. Their work is inspired by the advances in research on grammars, where it is shown that discriminative models are strictly more powerful than the probabilistic context-free grammars. A common limitation of modeling page layouts using stochastic grammars is that optimal geometric parsing is exponential in the number of terminal symbols. Liang et al. [LNSV05] have proposed a set of efficient geometric constraints that yield near $O(n^3)$ complexity on most types of printed documents, where n is the number of terminal symbols. Using these constraints, Shilman et al. [SLV05] were able to achieve a parsing time of 30 seconds for the task of grouping text-lines into paragraphs and text-columns (80 terminal symbols) on a 1.7GHz Pentium 4 machine. If we consider the task of grouping connected components into text-lines, where the number of terminal symbols is usually around 4000 for a typical A4 document, the running time of the algorithm becomes a major bottleneck.

Other attempts for statistical modeling of page layout include the Markov Random Field (MRF) approach by Liang et al. [LHP99], and the generative zone model approach of Gao et al. [GWHD07]. Liang et al. modeled the statistical relationships between the locations of characters, lines, and paragraphs as a hierarchical MRF. The model is trained by measuring different kinds of distances between terminal and non-terminal symbols on an extensive training set. Gao et al. present the approach of generating overlapping zone hypothesis by using Voronoi diagrams. Then an optimal maximum a posteriori non-overlapping zone combination is obtained using a learned generative zone model. 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.

In this work, page layout is represented as a structural mixture model, where each component in the model is a layout that we are interested in. An individual layout is represented as a hierarchical tree of horizontal or vertical whitespace cuts - that is axis-aligned whitespace rectangles that divide a particular page segment into two parts. A parametric model is built to model the geometric variability in position and size of corresponding whitespace cuts across different documents of the same layout. For each layout, the distribution of parameters is estimated from the training set. These learned models are then matched on the input image and the best matching model is returned with the positions of best fit and the associated probability. The details of the layout model and the matching algorithm are described in Section 7.2. Experimental results are presented in Section 7.3 followed by a conclusion in Section 7.4.

7.2 Statistical Layout Analysis

7.2.1 Statistical Layout Model

The first problem that needs to be addressed for designing a statistical layout analysis algorithm is the representation of page layout. Although different models for page layouts have been proposed in the literature, each model comes with its own problems. The hierarchical MRF model by Liang et al. and the generative zone model by Gao et al. require large amounts of labeled training data to obtain good results. Stochastic grammars, on the other hand, are not a natural representation of page layouts. Page layouts are generated by a number of typesetting rules and hence exhibit a large amount of regularity. However, parsing a page with stochastic grammars might result in page layouts that do not appear in practice.

Instead of trying to model generic page layouts, we take the approach of style-directed

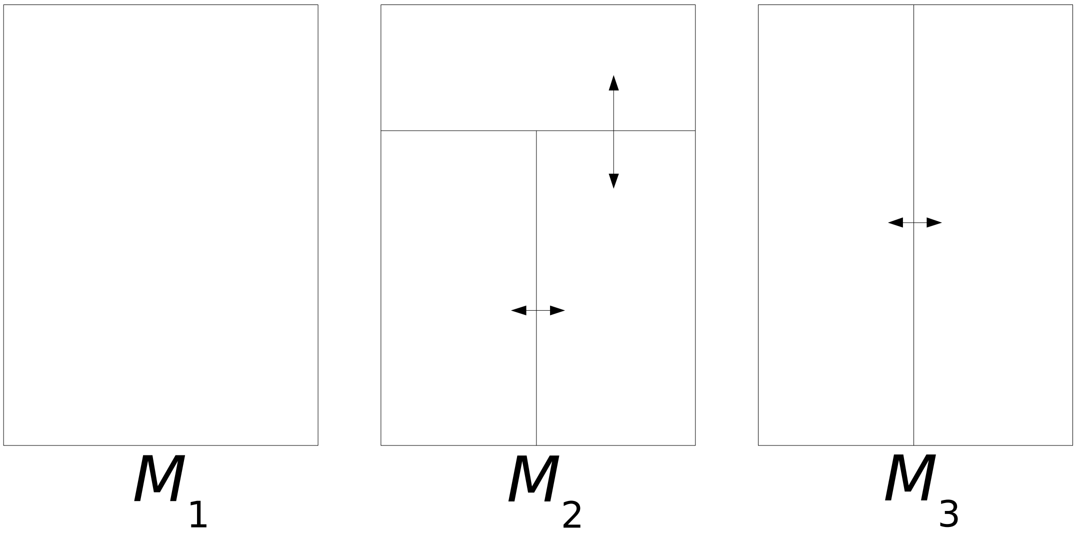


Figure 7.4: An illustration of modeling page layout as a structural mixture model. This example models page layout as a mixture of three layout components. The geometric variability of these components is visualized by the arrows.

layout analysis [DLU87, Spi01, KM03]. A particular 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. In contrast to previous approaches like [DLU87, Spi01] that use rule-based systems 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 7.4. The primary 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.

Let a layout component be modeled as a sequence of rectangles $M = \{m_1, \ldots, m_N\}$, each defined by four parameters describing the center position (x, y), width w, and height h of the corresponding rectangle. These parameters are assumed to have independent Gaussian distributions. The sequence describes the hierarchy of model rectangles. An illustration of such a hierarchical tree is shown in Figure 7.5. Some important features of this model tree are:

- Each model rectangle divides a page segment into two parts. Due to this property a model rectangle will also be referred as a model cut in the work.
- The parameters of model rectangles are relative to the page segment to which they

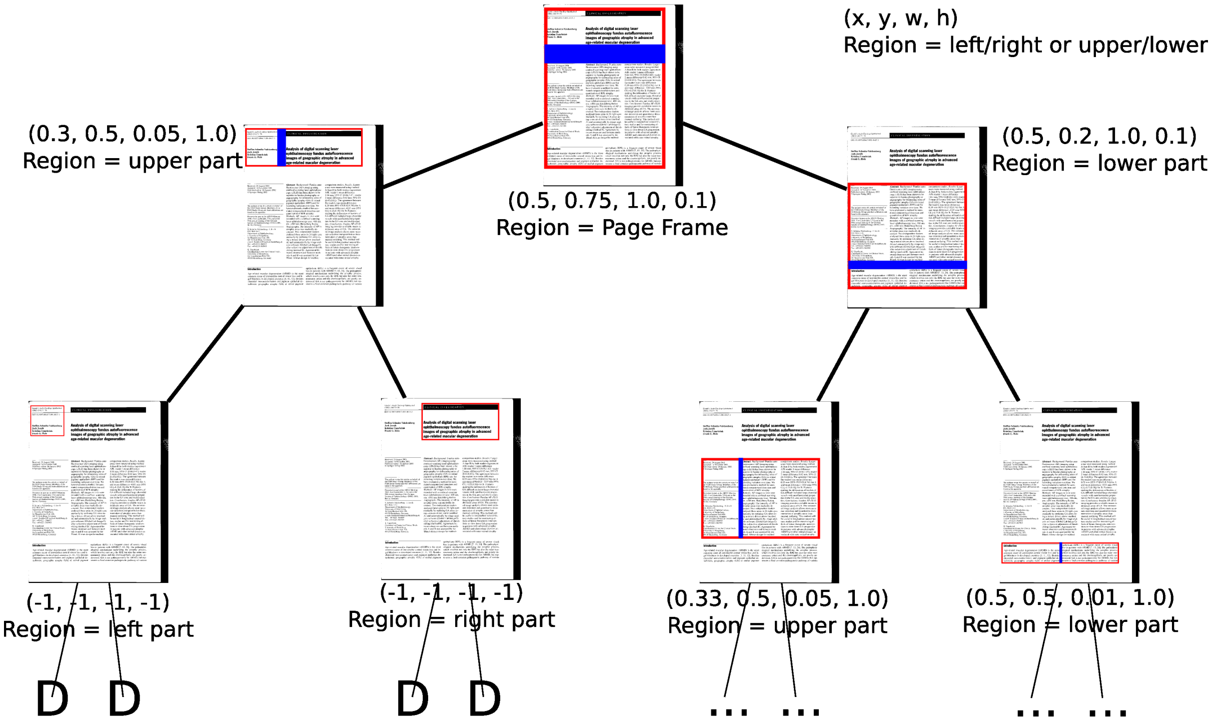


Figure 7.5: An example image illustrating the hierarchical tree model of page layout as used in this work.

are applied.

- The first model cut is applied to the page frame. A horizontal cut divides the page frame into upper and lower parts, whereas a vertical cut divides the page frame into left and right parts. As a result of applying the model cut to the page frame, two new page segments are generated.

- The generated page segments are inserted as children of the root node in a predefined order. Upper or left page segment is inserted as the left child, whereas lower or right page segment is inserted as the right child.

- If a page segment is not further sub-divided, two dummy nodes are added as its children.

For the convenience of implementation, the binary model tree is stored as an array such that the root node of the tree is at index zero, and the children of a node at index i are found at indices $2i + 1$ and $2i + 2$, while its parent (if any) is found at index $\lfloor (i - 1)/2 \rfloor$.

A particular advantage of the ordered hierarchical structure of the layout tree is that the foreground segments can be extracted in reading order by a simple depth-first-search

(DFS) traversal of the layout tree. During the DFS traversal of the tree, as soon as a dummy node is encountered, its corresponding page segment is marked as a final page segment and its children are not explored. The order in which foreground segments are visited corresponds to their reading order as illustrated in Figure 7.6.

7.2.2 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. For this purpose, first a whitespace cover of the page background is extracted using the algorithm described in [Bre02c]. Then, each layout component in the structural mixture model is considered as a candidate that can explain the layout of the target document. We are interested in finding the layout model that best explains the target document and then extracting whitespaces corresponding to that best matching layout model. An illustration of this layout matching algorithm is shown in Figure 7.7.

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

- $W = (w_1, \dots, w_N)$ 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 probable subset of whitespaces:

$$\hat{W} = \arg\max_{W} p(W|M) \tag{7.1}$$

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

$$\begin{aligned} p(W|M) &= p(w_1, w_2, \cdots, w_N | m_1^N) \\ &= p(w_1|m_1^N) p(w_2, \cdots, w_N | w_1, m_1^N) \\ &= p(w_1|m_1^N) p(w_2|w_1, m_1^N) \cdots p(w_N|w_1, \cdots, w_{N-1}, m_1^N) \end{aligned} \tag{7.2}$$

where w_i is the whitespace cover rectangle that m_i has been matched on. Due to the hierarchical structure of our layout models, the probability of observing whitespace w_i does not depend on model cuts that are lower in the hierarchy, i.e. model cuts with indices

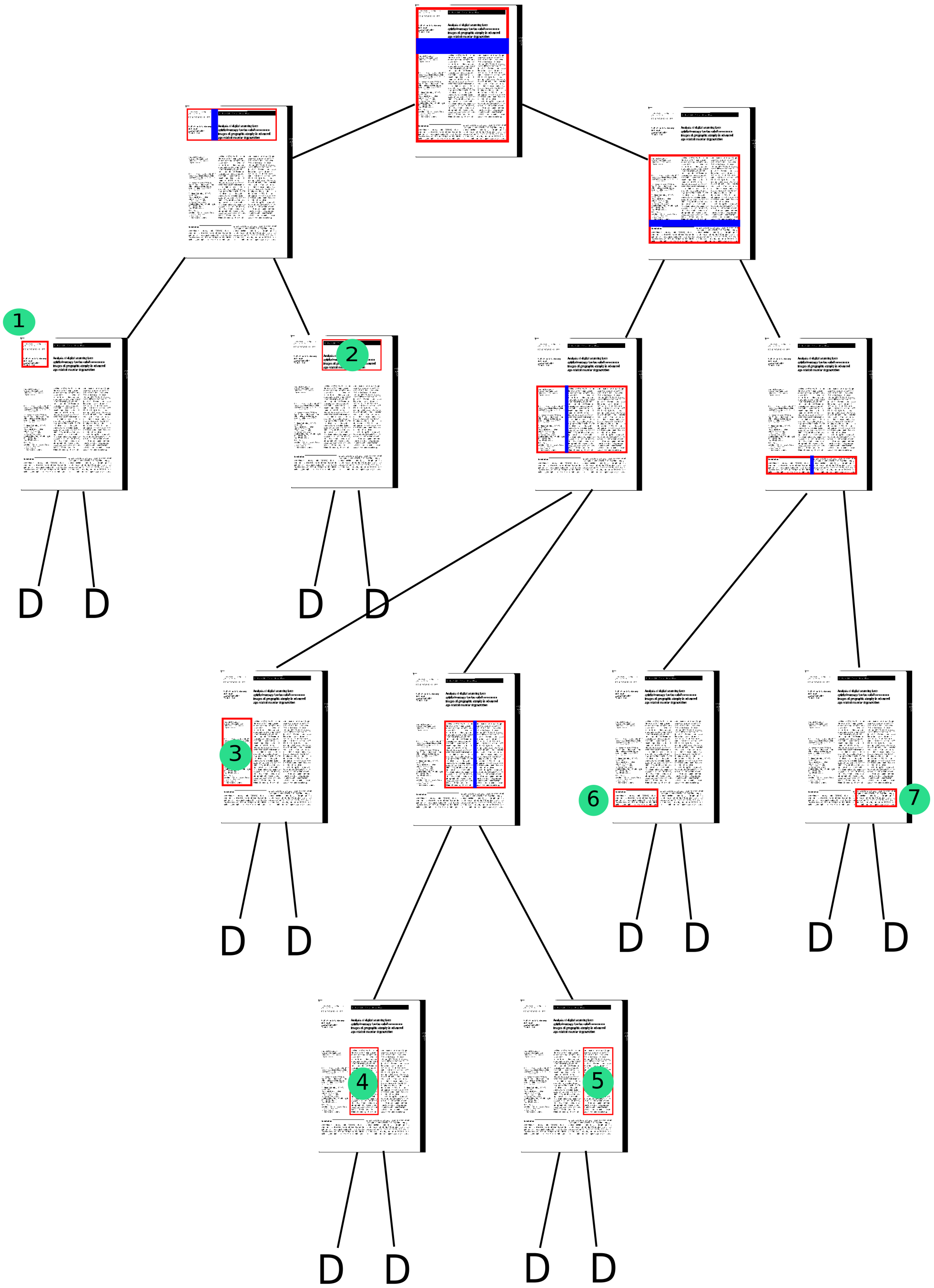


Figure 7.6: Example image illustrating the extraction of page foreground segments from the layout tree using depth-first-search traversal of the tree. The numbers indicate the order in which foreground segments in the final segmentation are visited. Due to the ordered hierarchical structure of the layout tree, this sequence corresponds to their reading order.

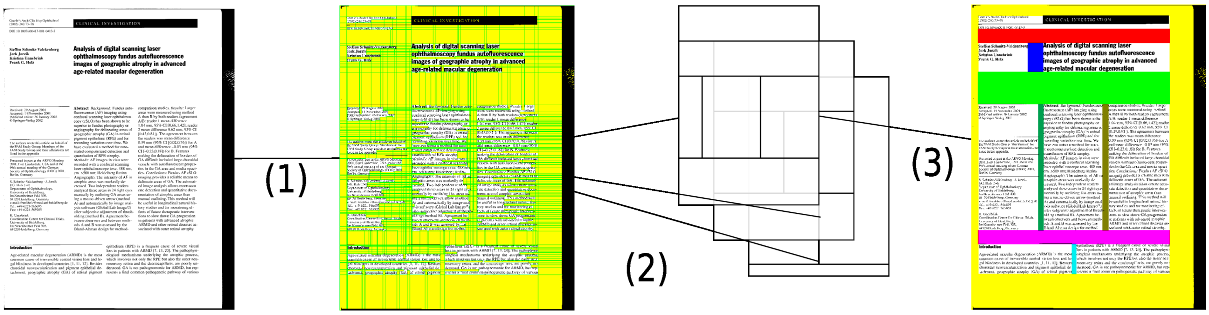


Figure 7.7: An overview of the different steps of the presented statistical layout matching algorithm. First, a whitespace cover of page background is extracted, as depicted by the yellow rectangles. Then, the whitespace rectangles are matched to model rectangles for different layout model components. Finally, the best fitting model and the corresponding whitespaces are extracted.

$i+1$ to N. Hence, the first term on the right hand side of Equation 7.2 - $p(w_1|m_1^N)$ - can be computed as

$$\begin{aligned} p(w_1|m_1^N) &= p(w_1|m_1, m_2, \cdots, m_N) \\ &= p(w_1|m_1) \\ &= \mathcal{N}(x_1;\ \mu_{x_1}, \sigma_{x_1})\mathcal{N}(y_1;\ \mu_{y_1}, \sigma_{y_1})\mathcal{N}(w_1;\ \mu_{w_1}, \sigma_{w_1})\mathcal{N}(h_1;\ \mu_{h_1}, \sigma_{h_1}) \quad (7.3) \end{aligned}$$

Similarly, other terms in Equation 7.2 can be written as:

$$p(w_j|w_1, \cdots, w_{j-1}, m_1^N) = p(w_j|w_1, \cdots, w_{j-1}, m_1^j) \quad (7.4)$$

The dependency between a whitespace cut w_j and its ancestors is modeled 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. Since the coordinates of whitespace cuts are computed relative to the page segment to which they are applied (see Figure 7.5), 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 probability of observing the updated whitespace can then be simply computed by using Equation 7.3.

Using Equations 7.3 and 7.4 in Equation 7.2 gives the probability of matching a particular combination of whitespaces to the layout model. The main challenge then is to find the global maximum in Equation 7.1. The total number of possible solutions for choosing N out of K whitespace rectangles is:

$$P_N^K = \frac{K!}{(K-N)!} \tag{7.5}$$

This implies that for 300 whitespaces and 10 model rectangles, the total number of possible solutions are 5×10^{24}. This is a combinatorial optimization problem and brute-force search to find the globally optimal solution is not practically possible. In this work, A* search is employed to find the globally optimal combination of whitespaces that best matches the layout model.

A* Search Algorithm

A* is a best-first graph search algorithm, which finds the least-cost path from an initial node to a goal node. It maintains a set of partial solutions, i.e. unexpanded leaf nodes of expanded nodes, stored in a priority queue. The priority assigned to a path passing through a node x is determined by the function:

$$f(x) = g(x) + h(x) \tag{7.6}$$

where $g(x)$ is a cost function, which measures the cost it incurred from the initial node to the current node x, and $h(x)$ is a heuristic function estimating the cost to the goal node from x. To ensure the search algorithm finds the optimal solution, $h(x)$ must be admissible, meaning that it never overestimates the actual minimal cost of reaching the goal.

To enable a straightforward application of A* search for finding the global maximum in Equation 7.1, the maximization problem can be reformulated as a minimization problem:

$$\hat{W} = \arg\min_{W} \{-\log p(W|M)\} \tag{7.7}$$

Hence Equation 7.2 can be written as

$$\log p(W|M) = \log p(w_1|m_1^N) + \log p(w_2|w_1, m_1^N) + \cdots + \log p(w_N|w_1, \cdots, w_{N-1}, m_1^N) \tag{7.8}$$

This minimization problem can be formulated as A* search as follows: Let the state variable x be defined as a sequence of rectangles $x = x_1, x_2, \ldots, x_N$ such that each element represents a whitespace from the document that has been matched to the corresponding model rectangle. All elements of the state variable x are assigned a dummy rectangle (a rectangle with parameters $(-1,-1,-1,-1)$) in the start of the search. As the search proceeds, these dummy rectangles are replaced by actual whitespaces that are matched to the corresponding model rectangles. Hence, an initial state will be represented by an N-dimensional sequence of dummy rectangles, whereas a goal state is represented by an N-dimensional sequence of whitespaces from the document. Note that the state variable will either have all dummy rectangles (initial state), all actual whitespaces (goal state), or k consecutive whitespaces ($k < N$) followed by $N-k$ consecutive dummy rectangles since all model cuts are matched in order.

The priority of a node x with k model cuts matched is given by

$$g(x) = -\log p(m_1|w_1^N) - \log p(m_2|m_1, w_1^N) - \cdots - \log p(m_k|m_1, \cdots, m_{k-1}, w_1^N) \quad (7.9)$$

Since the logarithm of the probability of a whitespace to represent a model cut is strictly non-negative, the heuristic function $h(x)$ can be safely set to zero:

$$h(x) = 0 \quad (7.10)$$

Since $h(x)$ defined above can never over-estimate the minimum cost of reaching the goal state, as soon as a goal state reaches the top of the priority queue, it is guaranteed to be the globally optimal solution for Equation 7.1. Using A* search, mean running time of matching one layout model to an image is less than one second on a 2GHz AMD Athlon machine running Linux. A hint from the implementation point of view is that when using double precision floating point numbers, the probability in Equation 7.3 goes to zero when the value inside any of the exponents gets larger than 746. Hence large portions of search space that do not fit the layout model are quickly discarded by the search algorithm.

7.2.3 Learning Model Parameters

An important aspect of the statistical layout analysis approach is that it can be trained without the need for page segmentation ground-truth. Learning layout models from a set of training images can be done in two steps.

In the first step, the goal is to find out 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 the present work, this task is done manually.

First, the user selects documents with the same layout. Then, a structural layout model is built from one document of that layout with the help of an interactive GUI that is specifically designed for that purpose. The GUI uses a whitespace cover computation algorithm to extract whitespaces belonging to page background. Clicking at a particular location on the image selects a whitespace with the highest area at that location. If this is the rectangle desired by the user, he just clicks on another position to select the next model rectangle. However, if the user did not intend to select that highest area rectangle presented to him by the system, he can click inside the selected rectangle. Clicking inside selected rectangles cycles through all rectangles that contain the click position. Some structural layout models built with that GUI are shown in Figure 7.8.

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

$$q = -\sum_{i=1}^{T} \log p_i(\hat{W}|M) \tag{7.11}$$

The training algorithm tries to minimize this quantity iteratively. An outline of the training procedure is as follows:

1. Initialize model parameters to some fixed values. Set mean values to the attributes of corresponding whitespaces selected by the user, and variance to some small arbitrary values.

2. Compute $q^{(0)}$ for training set using initial model by matching the initial model to all documents in the training set.

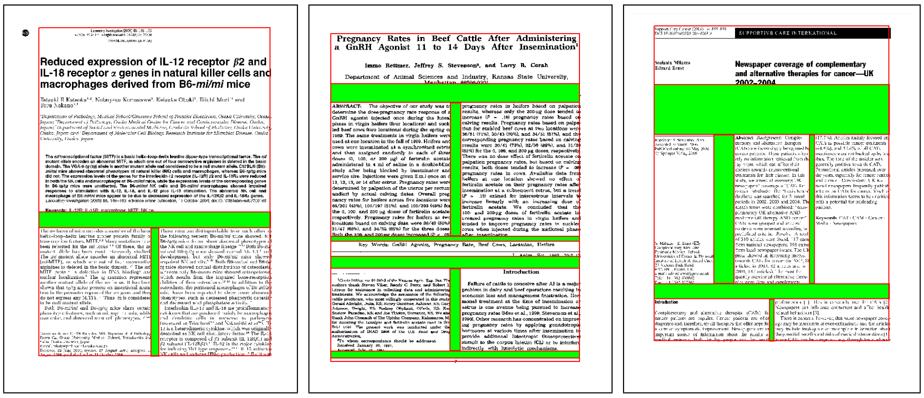
3. The matching result gives a set of whitespace rectangles for each training image that best match the model rectangles. Compute model parameters using maximum likelihood estimation from the obtained whitespaces.

4. Compute $q^{(1)}$ for training set using updated model parameters

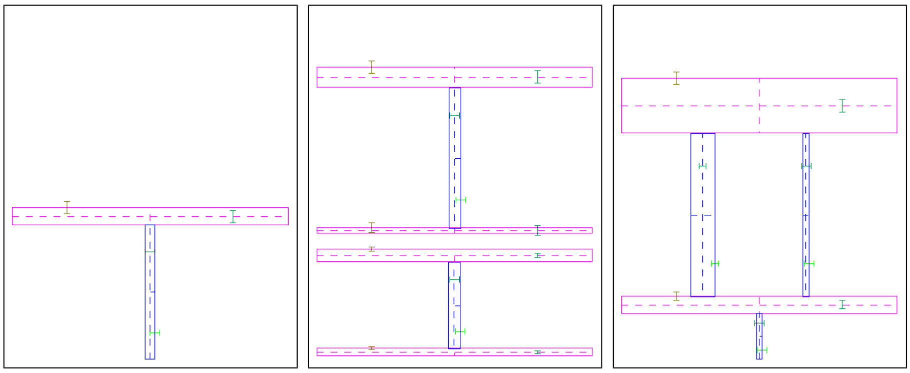
5. If $q^{(t)} \geq q^{(t-1)}$, then terminate; otherwise continue at Step 3

The training algorithm converges very quickly to a local minimum. The convergence of the training algorithm for different partitions of a 5-fold partitioned training set is shown in Figure 7.9. The convergence of each run to similar values suggests that the

(a) Some sample layouts selected by the user



(b) Structural layout model built by the user for each layout



(c) A visualization of initial model parameters for each layout structure

Figure 7.8: Figure illustrating the first step of learning layout models. (a) The user selects some representative layouts from the training data. (b) Structural layout models are built for each of the selected layouts with the help of an interactive GUI. (c) A visualization of initial parameters for each model rectangle. The rectangle itself represents the mean value and the horizontal/vertical bars represent 2σ interval around the mean values. A bar centered at the middle of a rectangle shows variation in position, whereas a bar centered on one side of a rectangle illustrates variation in width/height of the rectangle.

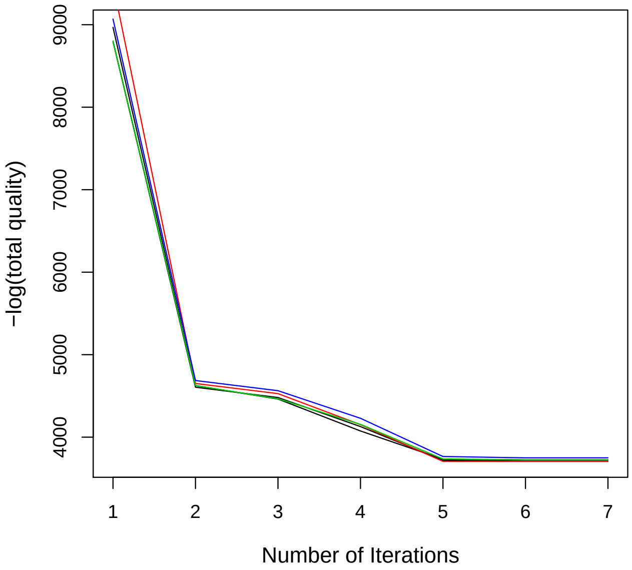


Figure 7.9: A plot showing the convergence of training algorithm. Different curves correspond to training on different partitions of data.

optimal parameters found for each partition correspond closely to the global optimum. A visualization of the model parameters learned in this way is shown in Figure 7.10. Note that the position and width of the column separator at the bottom of the page appears to vary very little over the training data and is reflected by small variances of corresponding model cut. Similarly, the large variance of lower horizontal separator's vertical position depicts that the actual position of this model cut varies quite a lot in the training data.

7.3 Experiments and Results

The statistical layout analysis algorithm presented in this chapter exhibits several key properties that are essential for layout analysis tasks in large scale application. To evaluate the performance of the algorithm, a subset of the publicly available MARG dataset [FT03] was chosen. The MARG dataset is naturally suitable for this purpose since it was developed as a part of the efforts made in digitizing the US National Library of Medicine. Therefore, it contains a large variety of journal layouts with several examples of title pages from each journal. The journal layouts are categorized into nine classes based on the geometric arrangement of logical page blocks (title, author, affiliation, abstract). Since this classification is made based on logical layout elements, layouts from

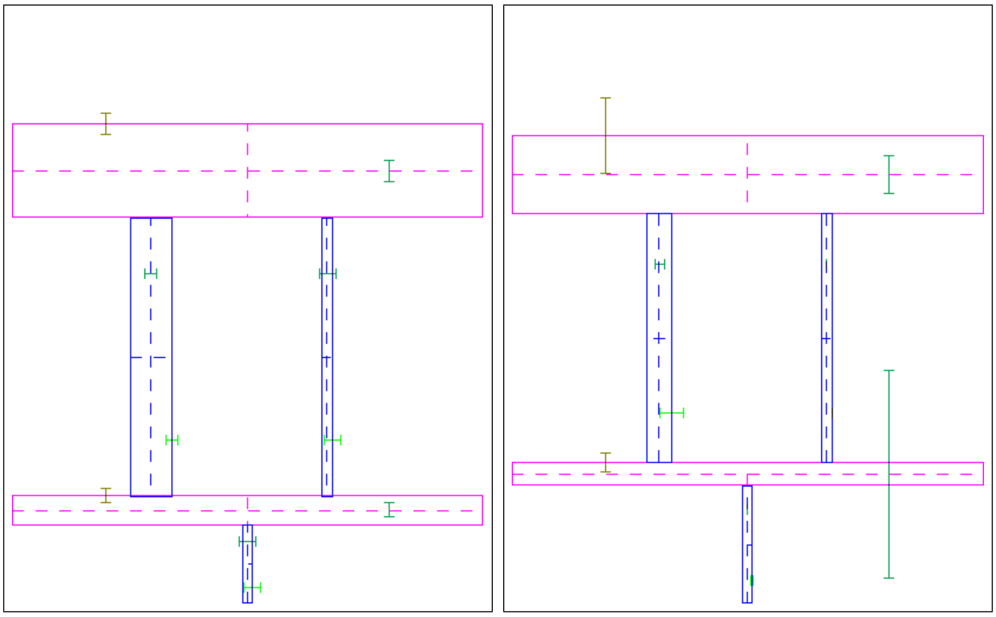


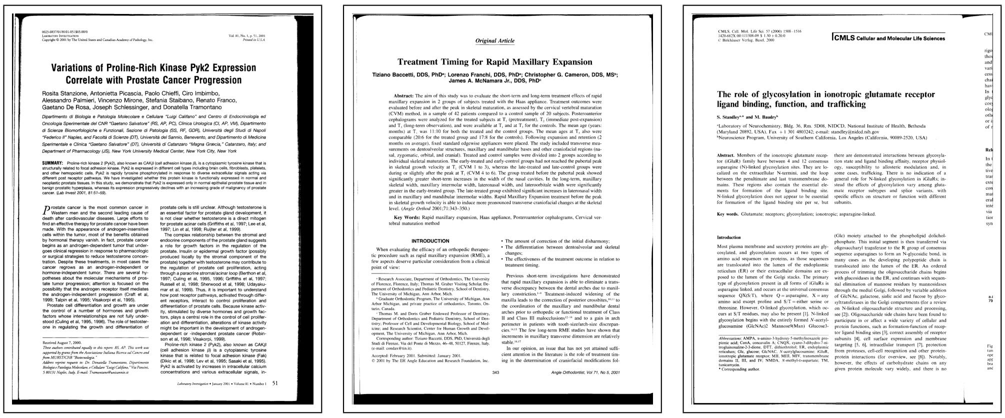
Figure 7.10: An illustration of the result of learning geometric variability of structural layout models. The left image shows a visualization of the initial model, whereas the right image shows a visualization of the learned model.

two different classes might look identical for geometric layout analysis. Secondly, for two journals belonging to the same class, geometric page layout might differ a lot.

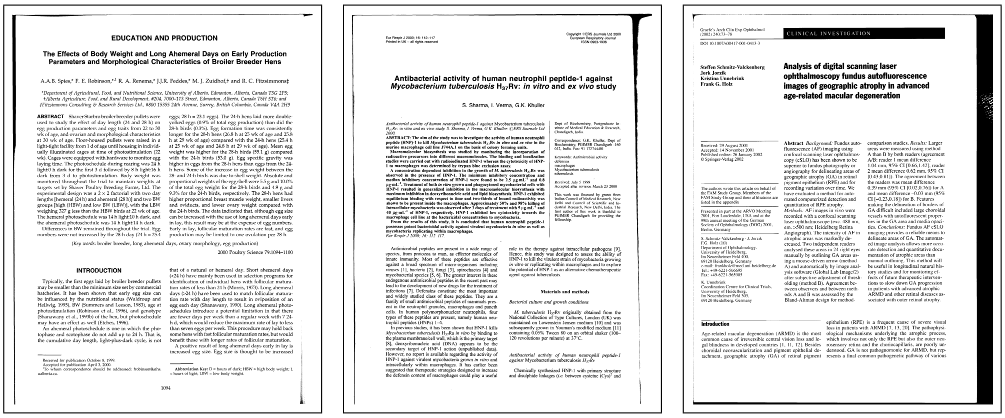
In this work, six journals were chosen from the MARG dataset that had different geometric layouts of the page. These journals are:

- Laboratory Investigations (LabInv)
- Angle Orthodontist (AngOrt)
- Cellular and Molecular Life Sciences (CMLS)
- Poultry Science (PouSci)
- European Respiratory Journal (ERS)
- Supportive Care in Cancer (SCC)

An example image from each of these journals is shown in Figure 7.11. There are 142 images of these journals collectively in the MARG dataset. The details of the number of samples from each journal are shown in Table 7.1. The number of samples per class are too small for a reasonable training of the layout model of that class. Therefore, more samples of each journal were obtained through the central library of Technical University



(a) LabInv (b) AngOrt (c) CMLS



(d) PouSci (e) ERS (f) SCC

Figure 7.11: Example images from each of the journals used in the experiments.

of Kaiserslautern. The number of samples per class thus obtained are also shown in Table 7.1. After obtaining a reasonably sized dataset, it was partitioned into five parts to separate training and test sets. 5-fold cross-validation was then used in all experiments to obtain reliable results.

Four experiments were then designed to evaluate the performance of the algorithm in three different use-cases. In a first experiment, c.f. 7.3.1, the assumption of the parameters being distributed normally is analyzed. The second experiment, described in Section 7.3.2, aims at testing if the obtained page segmentation is correct, given a document image and its layout model. The experiment presented in Section 7.3.3 tests the ability of the proposed approach to find the correct model for an unknown document image type. Finally, Section 7.3.4 shows the results of the new approach on the subset of the MARG dataset.

Journal	No. of samples in MARG dataset	No. of additionally collected samples
Laboratory Investigations	5	127
Angle Orthodontist	14	169
Cellular and Molecular Life Sciences	38	213
Poultry Science	25	224
European Respiratory Journal	49	205
Supportive Care in Cancer	11	220

Table 7.1: The number of samples of each journal used in the experiments.

7.3.1 Experiment 1

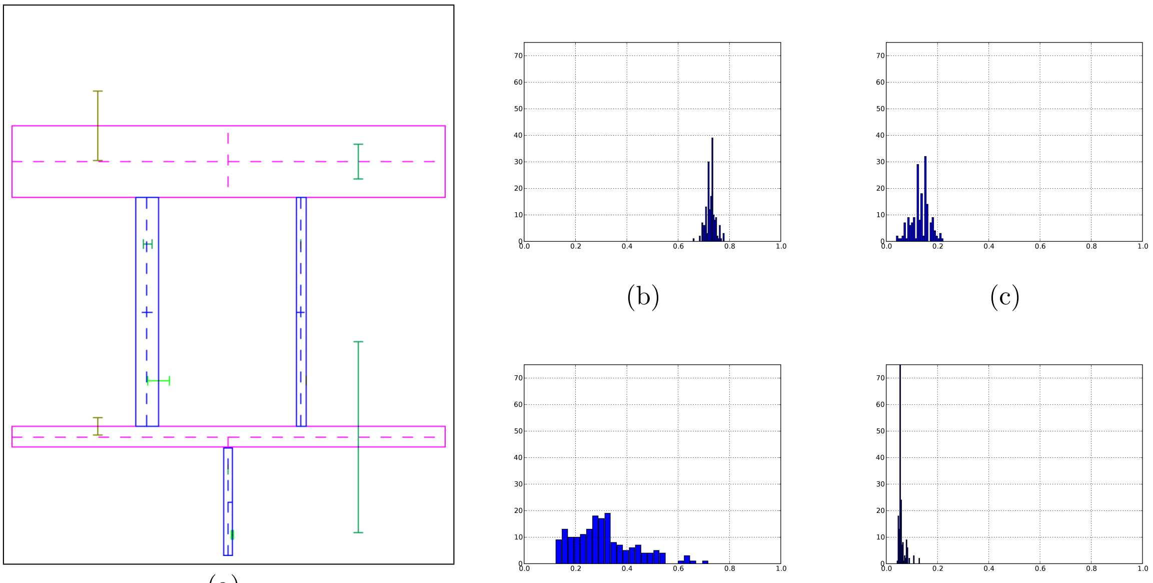
The proposed approach of model matching for page segmentation is partially based on the assumption, that the parameters of the model have a Gaussian distribution. To verify if this assumption is correct, the following experiment has been done: after learning a model for a given type, the model has been matched to all the documents of that type. If the assumption is correct, the parameters should be distributed normally.

In Figure 7.12(a), the histograms of the parameters y position and height of the upper and lower horizontal whitespace cuts are plotted for the shown model. It can be seen that the parameters appear to be distributed normally, which supports the assumption about the distribution of the parameters.

7.3.2 Experiment 2

In this experiment, the performance of the statistical model matching approach is tested on synthetic document images where the model is known. Thus, the method tries to match the correct model to the document image. The most likely segmentation of the page according to the model is obtained. A segmentation is considered correct if the resulting segmentation is the same as the canonical text to block mapping, grouping logical text blocks together. If a segmentation maps text of different blocks together (e.g. text lines from the abstract together with text lines from the title), the segmentation is considered wrong.

Total accuracy for this test is 99.6%. In total 1153 samples of 1158 where segmented correctly. A detailed overview of the number of errors per type can be found in Table 7.2.



(a) (b) (c) (d) (e)

Figure 7.12: In Figure 7.12(a) the model is visualized. Figure 7.12(b) and 7.12(d) show the distribution of the y-positions of the upper and lower horizontal whitespace cut respectively. Figure 7.12(c) and 7.12(e) show the distribution of heights of the upper and lower horizontal whitespace cut.

7.3.3 Experiment 3

This experiment focusses on the ability of the method to find the correct model for an unknown document layout type belonging to one of the trained models. The method finds the most likely model for a given document image. A correct segmentation is again defined as being the canonically correct mapping of the text to the blocks. In this case, matching the wrong model also leads to an incorrect segmentation.

This test yielded 57.5% of correctly matched models. The confusion matrix is shown in Table 7.3. It can be seen that the simple model of LabInv journal matched documents

Journal	samples	correct
LabInv	127	127
AngOrt	169	169
CMLS	213	213
PouSci	224	224
ERS	205	201
SCC	220	219

Table 7.2: Segmentation accuracy of pages of a given journal.

Journal	LabInv	AngOrt	CMLS	PouSci	ERS	SCC
LabInv	127					
AngOrt		169				
CMLS	184		29			
PouSci	21			203		
ERS	129				76	
SCC	158					62

Table 7.3: Confusion matrix for the matching multiple models using log-likelihood as the match quality. It can be clearly seen that simple models, having a smaller number of whitespace cuts, are preferred.

Journal	LabInv	AngOrt	CMLS	PouSci	ERS	SCC
LabInv	124		3			
AngOrt		169				
CMLS			213			
PouSci				224		
ERS			1		204	
SCC			1			219

Table 7.4: Confusion matrix for the matching using the normalized quality of fit (Equation 7.12). Only 5 documents are segmented using the wrong model type.

with more complex layouts. A closer investigation of this problem showed that if a layout model is a subset of another layout model, the simpler model will always fit in the documents of the more complex model. Additionally, 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 7.2). 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 sub-models when both have a good matching score. The quality function thus obtained is:

$$q = -\frac{\log p(\hat{W}|M)}{N^2} \tag{7.12}$$

The use of this quality function increased the total accuracy to 99.6%, which means that 1153 documents out of the 1158 were segmented correctly. The confusion matrix can be found in Table 7.4.

Journal	LabInv	AngOrt	CMLS	PouSci	ERS	SCC
LabInv	0	5				
AngOrt		13	1			
CMLS			38			
PouSci				25		
ERS			1		48	
SCC						11

Table 7.5: Confusion matrix for documents from the MARG dataset. The correct layout was found for 135 out of 142 documents.

7.3.4 Experiment 4

The test setup in this experiment is the same as for the previous one, except for the test data, which in this case consists of document images from MARG dataset. Again the canonical correctness of the text to block mapping is used as correctness measure. The total accuracy for this test is 95.1% using the normalized quality of fit (Equation 7.12). In absolute numbers this means that 135 out of the 142 documents have been successfully segmented. The confusion matrix can be found in Table 7.5.

A closer look at the results showed that the errors are mainly due to complex models being fit to simple layouts. In the case of LabInv, which is a two column layout having a one column title part and a two column footer, the AngOrt model consisting of a one column title, two column main text part and a one column footer fitted well, and was chosen due to the normalization of the quality.

7.4 Summary

This chapter presented a novel statistical approach to layout analysis. The approach is based on top-down modeling of page layouts using a mixture of structural layout models. The geometric variability of individual layout components is modeled as a multi-variate Gaussian distribution, which can be learned from training data without the need for page segmentation ground-truth. An algorithm for finding the globally optimal match of a layout model to a target document was presented. The algorithm returns the probability of match as its confidence score. The trainable nature of the algorithm and its probabilistic output make it suitable for performing geometric layout analysis in large-scale digitization tasks. The application of the algorithm on documents collected by the author and on the MARG dataset showed high accuracy for geometric layout analysis.

Chapter 8

Conclusions and Outlook

Starting from algorithms for efficient preprocessing, accurate page segmentation, and simple and effective block classification, this thesis lead to a high-accuracy, trainable layout analysis system. This thesis made several key contributions to the state-of-the-art in geometric layout analysis, the most important of which are outlined here.

This thesis presented an efficient local thresholding algorithm based on integral images. The time complexity of the presented algorithm is independent of local window size and is of the same order as that of global thresholding algorithms. Hence, the algorithm successfully combines the strengths of local and global thresholding schemes, achieving the same binarization result as that of local thresholding techniques like Sauvola [SP00] in a time close to that of global thresholding schemes like Otsu [Ots79].

Then, this thesis presented page frame detection as a new approach for document image cleanup. A geometric matching algorithm was used in this work to automatically find the optimal page frame of structured documents. It was demonstrated that using page frame detection significantly reduces OCR error rates of commercial OCR systems.

An approach for evaluating page-segmentation algorithms using a color-based representation was proposed in this work. The proposed 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 to evaluate the performance of a page segmentation algorithm, a novel vectorial score was presented. Using this vectorial score, the strengths and weaknesses of six widely used page segmentation algorithms were highlighted. Based on a better understanding of the problems with each algorithm [SKB08b], a modification of the whitespace analysis approach by Baird [Bai94] was proposed. It was shown that with this modification, the whitespace analysis algorithm achieved the lowest mean error rate on the UW-III dataset among all the compared algorithms.

Furthermore, an in-depth study of the performance of different features for block classification was conducted in this thesis. It was concluded that a very accurate classification system can be constructed based on run-length histograms alone. A simple feature vector consisting of run-length histograms and connected component statistics achieved an error rate of 1.5% on the UW-III dataset, which equals the lowest error rate reported in the literature using more complex features.

Based on experiences made in analyzing Roman script documents, the first high-performance layout analysis system for Urdu documents was developed in this work. A geometric model was built to represent an Urdu text-line, and a branch-and-bound algorithm was employed to directly extract text-lines from a scanned document. Results show that the presented system achieves high text-line segmentation accuracy on Urdu documents with different layouts like books, magazines, and newspapers. An interesting future direction would be to apply the Urdu text-line model to Persian and Arabic script documents, since they are written in similar writing styles.

This thesis also presented a statistically motivated trainable layout analysis system that is specifically suitable for use in large scale document analysis tasks. A structural mixture model was proposed for representing known layouts. An EM-like training algorithm was developed that can learn geometric variability of model components from training data without the need for page segmentation ground-truth. Then, a probabilistic matching algorithm was presented that can find multiple interpretations of input layout with associated probabilities. Finally, the A* search algorithm was used to find the most likely layout of a page, given its layout model. Experiments on documents collected by the author and from the MARG dataset show that the system can perform geometric layout analysis of trained layouts with high accuracy. An important extension to this work would be to incorporate foreground information into the layout model and the matching algorithm so that the algorithm can distinguish layouts that are similar in structure, but have different foreground features e.g. different font size and/or font style.

Overall, this thesis presented high performance layout analysis systems for both desktop scanning and large scale document analysis applications. The algorithms presented in this thesis had a significant practical impact and were used to construct different components of the layout analysis module of OCRopus, a state-of-the-art open source OCR system. With these components, OCRopus was able to achieve highest text-line extraction accuracy among existing open source OCR systems [Bre08]. It is expected that the document analysis community will find the open source implementation of this work helpful in using and extending it.

Bibliography

[ADK03] S. Agne, A. Dengel, and B. Klein. Evaluating SEE - a benchmarking system for document page segmentation. In *Proc. Int. Conf. on Document Analysis and Recognition*, pages 634–638, Edinburgh, Scotland, Aug. 2003.

[AGB05] A. Antonacopoulos, B. Gatos, and D. Bridson. ICDAR 2005 page segmentation competition. In *Proc. 8th Int. Conf. on Document Analysis and Recognition*, pages 75–80, Seoul, Korea, Aug. 2005.

[AGB07] A. Antonacopoulos, B. Gatos, and D. Bridson. Page segmentation competition. In *Proc. 9th Int. Conf. on Document Analysis and Recognition*, pages 1279–1283, Curitiba, Brazil, Sep. 2007.

[AGK03] A. Antonacopoulos, B. Gatos, and D. Karatzas. ICDAR 2003 page segmentation competition. In *7th Int. Conf. on Document Analysis and Recognition*, pages 688–692, Edinburgh, UK, Aug. 2003.

[AKB06] A. Antonacopoulos, D. Karatzas, and D. Bridson. Ground truth for layout analysis performance evaluation. In *Proc. Document Analysis Systems*, pages 302–311, Nelson, New Zealand, Feb. 2006.

[AL04] B. T. Avila and R. D. Lins. Efficient removal of noisy borders from monochromatic documents. In *Int. Conf. on Image Analysis and Recognition*, pages 249–256, Porto, Portugal, Sep. 2004.

[ARR00] S. Agne, M. Rogger, and J. Rohrschneider. Benchmarking of document page segmentation. In *Proc. SPIE Document Recognition and Retrieval VII*, pages 165–171, San Jose, CA, USA, Jan. 2000.

[Bai94] H. S. Baird. Background structure in document images. In H. Bunke, P. Wang, and H. S. Baird, editors, *Document Image Analysis*, pages 17–34. World Scientific, Singapore, 1994.

[Bai00] H. S. Baird. State of the art of document image degradation modeling. In *IAPR Workshop on Document Analysis Systems*, Rio de Janeiro, Brazil, Dec. 2000.

[Bai04] H. S. Baird. Difficult and urgent open problems in document image analysis for libraries. In *Proc. 1st Int. Workshop on Document Image Analysis for Libraries*, pages 25–32, Palo Alto, CA, USA, Jan. 2004.

[Ber86] J. Bernsen. Dynamic thresholding of gray level images. In *Proc. Int. Conf. on Pattern Recognition*, pages 1251–1255, Paris, France, 1986.

[BNP06] E. Badekas, N. Nikolaou, and N. Papamarkos. Text binarization in color documents. *Int. Jour. of Imaging Systems and Technology*, 16(6):262–274, 2006.

[BP05] E. Badekas and N. Papamarkos. Automatic evaluation of document binarization results. In *10th Iberoamerican Congress on Pattern Recognition*, pages 1005–1014, Havana, Cuba, Nov. 2005.

[BR07] D. Bradley and G. Roth. Adaptive thresholding using the integral image. *Journal of Graphics Tools*, 12(2):13–21, 2007.

[Bre93] T. M. Breuel. Recognition by Adaptive Subdivision of Transformation Space: practical experiences and comparison with the Hough transform. In *IEE Colloquium on 'Hough Transforms' (Digest No.106)*, pages 71–74, 1993.

[Bre01] T. M. Breuel. A practical, globally optimal algorithm for geometric matching under uncertainty. *Electronic Notes in Theoretical Computer Science*, 46:1–15, 2001.

[Bre02a] T. M. Breuel. Representations and metrics for off-line handwriting segmentation. In *Proc. 8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 428–433, Ontario, Canada, Aug. 2002.

[Bre02b] T. M. Breuel. Robust least square baseline finding using a branch and bound algorithm. In *Proc. SPIE Document Recognition and Retrieval IX*, pages 20–27, San Jose, CA, USA, Jan. 2002.

[Bre02c] T. M. Breuel. Two geometric algorithms for layout analysis. In *Proc. Document Analysis Systems*, pages 188–199, Princeton, NY, USA, Aug. 2002.

[Bre03a] T. M. Breuel. An algorithm for finding maximal whitespace rectangles at arbitrary orientations for document layout analysis. In *7th Int. Conf. on Document Analysis and Recognition*, pages 66–70, Edinburgh, UK, Aug. 2003.

[Bre03b] T. M. Breuel. High performance document layout analysis. In *Symposium on Document Image Understanding Technology*, Greenbelt, MD, USA, April 2003.

[Bre03c] T. M. Breuel. Implementation techniques for geometric branch-and-bound matching methods. *Computer Vision and Image Understanding*, 90(3):258–294, 2003.

[Bre03d] T. M. Breuel. On the use of interval arithmetic in geometric branch-and-bound algorithms. *Pattern Recognition Letters*, 24(9–10):1375–1384, 2003.

[Bre07] T. M. Breuel. The hOCR microformat for OCR workflow and results. In *Proc. Int. Conf. on Document Analysis and Recognition*, pages 1063–1067, Curitiba, Brazil, Sep. 2007.

[Bre08] T. M. Breuel. The OCRopus open source OCR system. In *Proc. SPIE Document Recognition and Retrieval XV*, pages 0F1–0F15, San Jose, CA, USA, Jan. 2008.

[CCMM98] R. Cattoni, T. Coianiz, S. Messelodi, and C. M. Modena. Geometric layout analysis techniques for document image understanding: a review. Technical Report 9703-09, IRST, Trento, Italy, 1998.

[CLG03] J. Chen, M. K. Leung, and Y. Gao. Noisy logo recognition using line segment Hausdorff distance. *Pattern Recognition*, 36(4):943–955, 2003.

[CLLT02] L. Cinque, S. Levialdi, L. Lombardi, and S. Tanimoto. Segmentation of page images having artifacts of photocopying and scanning. *Pattern Recognition*, 35(5):1167–1177, 2002.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*, chapter 23, pages 485–488. MIT Press, Cambridge, MA, 1990.

[Cro84] F. C. Crow. Summed-area tables for texture mapping. *Computer Graphics - Proceedings of SIGGRAPH'84*, 18(3):207–212, 1984.

[DB89] A. Dengel and G. Barth. ANASTASIL: Hybrid knowledge-based system for document image analysis. In *Proc. Int. Joint Conference on Artificial Intelligence*, pages 1249–1254, Detroit, MI, USA, Aug. 1989.

[DDS+97] 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 H. Bunke and P. Wang, editors, *Handbook of character recognition and document image analysis*, pages 421–456. World Scientific, Singapore, 1997.

[DKN04] T. Deselaers, D. Keysers, and H. Ney. Features for image retrieval: A quantitative comparison. In *26th DAGM Symposium on Pattern Recognition*, pages 228–236, Tübingen, Germany, Aug. 2004.

[DLU87] A. Dengel, A. Luhn, and B. Ueberreiter. Model based segmentation and hypothesis generation for the recognition of printed documents. In *Proceedings of the SPIE-87*, pages 89–100, Cannes, France, Nov. 1987.

[DSC02] A. K. Das, S. K. Saha, and B. Chanda. An empirical measure of the performance of a document image segmentation algorithm. *Int. Jour. on Document Analysis and Recognition*, 4(3):183–190, 2002.

[FSCG03] R. P. Futrelle, M. Shao, C. Cieslki, and A. E. Grimes. Extraction, layout analysis and classification of diagrams in PDF documents. In *Proc. Int. Conf. on Document Analysis and Recognition*, pages 1007–1014, Edinburgh, Scotland, Aug. 2003.

[FT03] G. Ford and G. R. Thoma. Ground truth data for document image analysis. In *Symposium on Document Image Understanding and Technology*, pages 199–205, Greenbelt, MD, USA, April 2003.

[FWL02] K. C. Fan, Y. K. Wang, and T. R. Lay. Marginal noise removal of document images. *Pattern Recognition*, 35(11):2593–2611, 2002.

[GHHP97] I. Guyon, R. M. Haralick, J. J. Hull, and I. T. Phillips. Data sets for OCR and document image understanding research. In H. Bunke and P. Wang, editors, *Handbook of character recognition and document image analysis*, pages 779–799. World Scientific, Singapore, 1997.

[GWHD07] D. Gao, Y. Wang, H. Hindi, and M. Do. Decompose document image using integer linear programming. In *Proc. Int. Conf. on Document Analysis and Recognition*, pages 397–401, Curitiba, Brazil, Sep. 2007.

[HA02] S. A. Husain and S. H. Amin. A multi-tier holistic approach for Urdu Nastaliq recognition. In *IEEE Int. Multi-topic Conference*, Karachi, Pakistan, Dec. 2002.

[HJBJ$^+$96] 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*, 18(7):673–689, 1996.

[Hol04] A. Holmes. Publishing trends and practices in the scientific community. *Canadian Journal of Communication*, 29:359–368, 2004.

[Hus04] S. Hussain. Urdu localization project. In *Workshop on Computational Approaches to Arabic Script-based Languages*, pages 80–81, Geneva, Switzerland, Aug. 2004.

[IW95] S. Inglis and I. H. Witten. Document zone classification using machine learning. In *Proc. Digital Image Computing: Techniques and Applications*, pages 631–636, Brisbane, Australia, Dec. 1995.

[JFMF05] M. S. Jelodar, M. J. Fadaeieslam, N. Mozayani, and M. Fazeli. A Persian OCR system using morphological operators. *Proc. of World Academy of Science, Engineering and Technology*, 4:137–140, 2005.

[JMIB06] X. Jiang, C. Marti, C. Irniger, and H. Bunke. Distance measures for image segmentation evaluation. *EURASIP Journal on Applied Signal Processing*, 2006(1):Article ID 35909, 10 pages, 2006.

[KC94] G. E. Kopec and P. A. Chou. Document image decoding using markov source models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(6):602–617, 1994.

[KD98a] T. Kieninger and A. Dengel. A paper-to-HTML table converting system. In *Proc. Document Analysis Systems*, pages 356–365, Nagano, Japan, Nov. 1998.

[KD98b] T. Kieninger and A. Dengel. Table recognition and labeling using intrinsic layout features. In *Proc. Int. Conf. on Advances in Pattern Recognition*, Plymouth, UK, Nov. 1998.

[KD01] T. Kieninger and A. Dengel. Applying the T-RECS table recognition system to the business letter domain. In *Proc. Int. Conf. on Document Analysis and Recognition*, pages 518–522, Seattle, WA, USA, Sep. 2001.

[Kho02] M. S. Khorsheed. Off-Line Arabic character recognition - a review. *Pattern Analysis and Applications*, 5(1):31–45, 2002.

[KIDM98] K. Kise, M. Iwata, A. Dengel, and K. Matsumoto. A computational geometric approach to text-line extraction from binary document images. In *Proc. Document Analysis Systems*, pages 346–355, Nagano, Japan, Nov. 1998.

[KK96] A. C. Kam and G. E. Kopec. Document image decoding by heuristic search. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(9):945–950, 1996.

[KKJ07] K. S. Kumar, S. Kumar, and C. Jawahar. On segmentation of documents in complex scripts. In *9th Int. Conf. on Document Analysis and Recognition*, pages 1243–1247, Curitiba, Brazil, Sep. 2007.

[KKR07] T. Kasar, J. Kumar, and A. G. Ramakrishnan. Font and background color independent text binarization. In *2nd Int. Workshop on Camera-Based Document Analysis and Recognition*, pages 3–9, Curitiba, Brazil, Sep. 2007.

[KM03] 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.

[KNRN93] J. Kanai, T. A. Nartker, S. V. Rice, and G. Nagy. Performance metrics for document understanding systems. In *Proc. 2nd Int. Conf. on Document Analysis and Recognition*, pages 424–427, Tsukuba, Japan, Oct. 1993.

[KSB07] 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*, pages 44–51, Barcelona, Spain, Mar. 2007.

[KSI98] 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.

[Lev66] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[LHA$^+$04] 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.

[LHP99] J. Liang, R. M. Haralick, and I. T. Phillips. A statistically based, highly accurate text-line segmentation method. In *Proc. Int. Conf. on Document Analysis and Recognition*, pages 551–555, Bangelore, India, Sep. 1999.

[LNG00] J. Li, A. Najmi, and R. M. Gray. Image classification by a two dimensional hidden markov model. *IEEE Trans. on Signal Processing*, 48(2):517–533, 2000.

[LNSV05] P. Liang, M. Narasimhan, M. Shilman, and P. Viola. Efficient geometric algorithms for parsing in two dimensions. In *Proc. 8th Int. Conf. on Document Analysis and Recognition*, pages 1172–1177, Seoul, Korea, Aug. 2005.

[LPH01] J. Liang, I. T. Phillips, and R. M. Haralick. Performance evaluation of document structure extraction algorithms. *Computer Vision and Image Understanding*, 84(1):144–159, 2001.

[LPHH96] J. Liang, I. Phillips, J. Ha, and R. Haralick. Document zone classification using the sizes of connected components. In *Proc. SPIE Document Recognition III*, pages 150–157, San Jose, CA, USA, Jan. 1996.

[LTW96] D. X. Le, G. R. Thoma, and H. Wechsler. Automated borders detection and adaptive segmentation for binary document images. In *13th Int. Conf. on Pattern Recognition*, pages 737–741, Vienna, Austria, Aug. 1996.

[MK01] S. Mao and T. Kanungo. Empirical performance evaluation methodology and its application to page segmentation algorithms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(3):242–256, 2001.

[MK02] S. Mao and T. Kanungo. Software architecture of PSET: a page segmentation evaluation toolkit. *Int. Jour. on Document Analysis and Recognition*, 4(3):205–217, 2002.

[Nag00] G. Nagy. Twenty years of document image analysis in PAMI. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(1):38–62, 2000.

[Nib86] W. Niblack. *An Introduction to Image Processing*, pages 115–116. Prentice-Hall, Englewood Cliffs, NJ, 1986.

[NSV92] G. Nagy, S. Seth, and M. Viswanathan. A prototype document image analysis system for technical journals. *Computer*, 7(25):10–22, 1992.

[ODP99] O. Okun, D. Doermann, and M. Pietikainen. Page segmentation and zone classification: the state of the art. Technical Report LAMP-TR-036, CAR-TR-927, CS-TR-4079, University of Maryland, College Park, November 1999.

[O'G93] L. O'Gorman. The document spectrum for page layout analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(11):1162–1173, 1993.

[O'G94] L. O'Gorman. Binarization and multithresholding of document images using connectivity. *Graphical Models and Image Processing*, 56(6):494–506, 1994.

[OPS99] O. Okun, M. Pietikainen, and J. Sauvola. Robust skew estimation on low-resolution document images. In *5th Int. Conf. on Document Analysis and Recognition*, pages 621–624, Bangalore, India, Sep. 1999.

[Ots79] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[Pha03] T. D. Pham. Unconstrained logo detection in document images. *Pattern Recognition*, 36(12):3023–3025, 2003.

[Phi96] I. T. Phillips. User's reference manual for the UW english/technical document image database III. Technical report, Seattle University, Washington, 1996.

[PS03] U. Pal and A. Sarkar. Recognition of printed Urdu script. In *7th Int. Conf. on Document Analysis and Recognition*, pages 1183–1187, Edinburgh, UK, Aug. 2003.

[PT06] F. Porikli and O. Tuzel. Fast construction of covariance matrices for arbitrary size image windows. In *Proc. Int. Conf. on Image Processing*, pages 1581–1584, Atlanta, GA, USA, Oct. 2006.

[SB07] F. Shafait and T. M. Breuel. Document image dewarping contest. In *2nd Int. Workshop on Camera-Based Document Analysis and Recognition*, pages 181–188, Curitiba, Brazil, Sep. 2007.

[SKB06a] F. Shafait, D. Keysers, and T. M. Breuel. Performance comparison of six algorithms for page segmentation. In *7th IAPR Workshop on Document Analysis Systems*, volume 3872 of *Lecture Notes in Computer Science*, pages 368–379, Nelson, New Zealand, Feb. 2006.

[SKB06b] F. Shafait, D. Keysers, and T. M. Breuel. Pixel-accurate representation and evaluation of page segmentation in document images. In *18th Int. Conf. on Pattern Recognition*, pages 872–875, Hong Kong, China, Aug. 2006.

[SKB08a] F. Shafait, D. Keysers, and T. M. Breuel. Efficient implementation of local adaptive thresholding techniques using integral images. In *Proc. SPIE Document Recognition and Retrieval XV*, pages 101–106, San Jose, CA, USA, Jan. 2008.

[SKB08b] 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):941–954, 2008.

[SKPB00] K. Sobottka, H. Kronenberg, T. Perroud, and H. Bunke. Text extraction from colored book and journal covers. *Int. Jour. on Document Analysis and Recognition*, 2(4):163–176, 2000.

[SLV05] M. Shilman, P. Liang, and P. Viola. Learning non-generative grammatical models for document analysis. In *Proc. Int. Conf. on Computer Vision*, pages 962–969, Beijing, China, Oct. 2005.

[SP00] J. Sauvola and M. Pietikainen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, 2000.

[SPH$^+$95] R. Sivaramakrishnan, I. T. Phillips, J. Ha, S. Subramanium, and R. M. Haralick. Zone classification in a document using the method of feature vector generation. In *Proc. 3rd Int. Conf. on Document Analysis and Recognition*, pages 541–544, Montreal, Canada, Aug. 1995.

[Spi01] A. L. Spitz. Style-directed document segmentation. In *Proc. Symp. Document Image Understanding Technology*, pages 195–199, Baltimore, MD, USA, Apr. 2001.

[SS04] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–165, 2004.

[SSMSSS06] S. S.-Shahreza, M. T. M.-Shalmani, and M. H. S.-Shahreza. Page segmentation of Persian/Arabic printed text using ink spread effect. In *SICE-ICASE Int. Joint Conference*, pages 259–262, Busan, Korea, Oct. 2006.

[SuHKB06] F. Shafait, A. ul Hasan, D. Keysers, and T. M. Breuel. Layout analysis of Urdu document images. In *10th IEEE Int. Multi-topic Conference*, pages 293–298, Islamabad, Pakistan, Dec 2006.

[SvBKB07] F. Shafait, J. van Beusekom, D. Keysers, and T. M. Breuel. Page frame detection for marginal noise removal from scanned documents. In *SCIA 2007, Image Analysis, Proceedings*, volume 4522 of *Lecture Notes in Computer Science*, pages 651–660, Aalborg, Denmark, June 2007.

[TC01] T. Tokuyasu and P. A. Chou. Turbo recognition: a statistical approach to layout analysis. In *Proc. SPIE Document Recognition and Retrieval VIII*, pages 123–129, San Jose, CA, USA, Jan. 2001.

[TD98] M. Thulke and A. Dengel. A general approach to quality evaluation of document segmentation results. In *Proc. Document Analysis Systems*, pages 79–88, Nagano, Japan, Nov. 1998.

[TL02] C. M. Tsai and H. J. Lee. Binarization of color document images via luminance and saturation color features. *IEEE Trans. on Image Processing*, 11(4):434–451, 2002.

[TMD99] M. Thulke, V. Maergner, and A. Dengel. Quality evaluation of document segmentation results, precision, and accuracy. In *Proc. Int. Conf. on Document Analysis and Recognition*, pages 450–454, Bangelore, India, Sep. 1999.

[TT95] O. D. Trier and T. Taxt. Evaluation of binarization methods for document images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(3):312–315, 1995.

[vBKSB06] J. van Beusekom, D. Keysers, F. Shafait, and T. M. Breuel. Distance measures for layout-based document image retrieval. In *2nd IEEE Int. Conf. on Document Image Analysis for Libraries*, pages 232–242, Lyon, France, April 2006.

[Vin07] L. Vincent. Google book search: Document understanding on a massive scale. In *9th Int. Conf. on Document Analysis and Recognition*, pages 819–823, Curitiba, Brazil, Sep. 2007.

[VJ04] P. Viola and M. J. Jones. Robust real-time face detection. *Int. Jour. of Computer Vision*, 57(2):137–154, 2004.

[WCW82] K. Y. Wong, R. G. Casey, and F. M. Wahl. Document analysis system. *IBM Jour. of Research and Development*, 26(6):647–656, 1982.

[WHP00] Y. Wang, R. Haralick, and I. Phillips. Improvement of zone content classification by using background analysis. In *Proc. Document Analysis Systems*, Rio de Janeiro, Brazil, Dec. 2000.

[Wik] http://en.wikipedia.org/wiki/Nastaliq_script.

[WPH06] Y. Wang, I. Phillips, and R. Haralick. Document zone content classification and its performance evaluation. *Pattern Recognition*, 39(1):57–73, 2006.

[WR83] J. M. White and G. D. Rohrer. Image thresholding for optical character recognition and other applications requiring character image extraction. *IBM Jour. of Research and Development*, 27(4):400–411, 1983.

[XY03] Y. Xiao and H. Yan. Text region extraction in a document image based on the delaunay tesselation. *Pattern Recognition*, 36(3):799–809, 2003.

[YV95] B. A. Yanikoglu and L. Vincent. Ground-truthing and benchmarking document page segmentation. In *Proc. 3rd Int. Conf. on Document Analysis and Recognition*, pages 601–604, Montreal, Canada, Aug. 1995.

[ZBC02] R. Zanibbi, D. Blostein, and J. R. Cordy. Recognizing Mathematical expressions using tree transformation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(11):1455–1467, 2002.

Curriculum Vitae

Personal Data

Name: Faisal Shafait

Date of Birth: February 15, 1981

Place of Birth: Jhelum, Pakistan

Family Status: Married, one son

Education

School Education	F. G. Public School, Mangla Cantt, *S*CC exam, **1995** Army Public School and College, Jhelum Cantt, *H*SSC exam, **1997**
University Education:	University of Engineering and Technology, Taxila, Pakistan, *Bachelor in Electrical Engineering*, **2002** Hamburg University of Technology, Hamburg, Germany *Masters in Information and Communication Systems*, **2005**

Academic and Professional Experience

Feb. 2003 – Nov. 2004	Student researcher in the Vision Systems and Automation labs at TUHH, Hamburg
Aug 2003 – Oct. 2003	Internship at Filtec Europe GmbH, Hamburg
May 2005 – present	Researcher in the IUPR lab at DFKI GmbH, Kaiserslautern